Contexte :

Soit une fonction "régulière" **G** passant par les points définis par les vecteurs ax et ay :
ax = 0:0.1:2
ay = [0.5, 0.6554217416, 0.7881446014, 0.8849303298, 0.9452007083, 0.9772498681];
plot(ax, ay, 'Marker', 'o');
Ces données sont disponibles sous : http://www.juggling.ch/gisin/coursam4os/s01_data.sce

Le but est d'interpoler "au mieux" la fonction complète entre x=0 et x=2.

1. L'interpolation en x = 1

Quelle valeur donneriez-vous à la fonction **G** en x = 1?
Quelle valeur donneriez-vous à la fonction **G** en x = 0.9?
Les réponses ne sont pas intéressantes, mais la manière de les obtenir le sont !

2. L'interpolation linéaire

Écrivez une fonction "z=intlin(x1, x2, y1, y2, x)" définie comme suit.
Pour 0 ≤ x1 ≤ x ≤ x2 ≤ 2, z est tel que le point (x, z) se trouve sur le segment allant de (x1, y1) à (x2, y2).
Testez votre interpolation linéaire sur le segment x1=ax(3); x2=ax(4); y1=ay(3); y2=ay(4); pour vérifier que l'on obtient les mêmes valeurs que sur le graphique.
La définition de cette fonction ne prend qu'une ligne !

3. L'interpolation linéaire, suite

Écrivez une fonction "fx=lineaire_interpol(ax, ay, x)" définie comme suit.
fx est l'image de x par l'interpolation linéaire sur le segment [ax(i), ax(i+1)] contenant x.
Testez votre interpolation linéaire pour x=0:0.5:2; pour vérifier que l'on obtient les mêmes valeurs que sur le graphique.

Sachant que les points viennent de la fonction :

```
function y=fct(x)
//===============
y = 0.5 + erf(x /sqrt(2))/2;
endfunction
```

Tracez la courbe de la différence entre le polynôme d'interpolation et la fonction sur [0, 2].

4. L'interpolation polynomiale

La fonction d'interpolation linéaire a le défaut de ne pas être dérivable aux points d'interpolation.
De plus, il semble évident que l'on peut faire mieux.
Écrivez une fonction "z=interpol_poly(axn, ayn, x)" définie comme suit.
Notons n la dimension des vecteurs axn et ayn .
z est l'évaluation en x du polynôme de degré n-1 passant par les points (axn(i), ayn(i)), i=1:n;

5. L'interpolation polynomiale

Déterminez le polynôme de degré length(ax)-1 qui interpole tous les points (ax, ay).
Tracez la courbe de la différence entre le polynôme d'interpolation et la fonction sur [0, 2].

6. L'interpolation quadratique

Écrivez une fonction "fx=quadratique_interpol(ax, ay, x)" définie comme suit.
Cherche l'indice j tel que ax(j) <= x <= ax(j+1).
Pose j = 2 si x < ax(2)
Pose j = length(ax)-1 si x > ax($-1) // ax($-1) est l'avant dernier élément de ax.
z est l'évaluation en x du polynôme de degré 2 passant par les points (ax(i), ay(i)), i=j-1:j+1;
z = interpol_poly(ax(j-1:j+1), ay(j-1:j+1), x);
Tracez la courbe de la différence entre le polynôme d'interpolation et la fonction sur [0, 2].

7. **L'interpolation cubique**
   Écrivez une fonction "fx=cubique_interpol(ax, ay, x)" définie comme suit.
   Cherche l'indice j tel que ax(j) <= x <= ax(j+1).
   Pose j = 2 si x < ax(2)
   Pose j = length(ax)-2 si x > ax($-2) // ax($-2) est l'avant dernier élément de ax.
   z est l'évaluation en x du polynôme de degré 3 passant par les points (ax(i), ay(i)), i=j-1:j+2;
   z = interpol_poly(ax(j-1:j+2), ay(j-1:j+2), x);
   Tracez la courbe de la différence entre le polynôme d'interpolation et la fonction sur [0, 2].

8. **L'interpolation spline,** juste pour information
   Une autre interpolation souvent utile, qui est très stable est l'interpolation spline.
   La théorie est hors niveau de ce cours et la pratique assez compliquée, lorsqu'un logiciel comme SciLab ne résout pas les problèmes pour nous.
   Une **courbe spline** est une courbe passant par les points d'interpolations, qui est partout dérivable, sa dérivée seconde est discontinue aux points d'interpolations. Entre chaque paire de points consécutifs de l'interpolation, la courbe spline correspond à un polynôme de degré 3.
   ```
   derivees = splin(ax, ay, "natural");  // c.f. help splin
   ```
   Fait tout le travail compliqué des calculs. Ensuite, l'évaluation se fait à l'aide de :
   ```
   images = interp(x, ax, ay, derivees);  //  x = vecteur des valeurs à évaluer
   plot(x, images);  // trace le graphique
   ```

Un peu de théorie sur le calcule d'un polynôme d'interpolation.
On a vu en fin de série 2 comment déterminer un polynôme d'interpolation. Cette méthode nécessite de résoudre un système d'équations, ce qui est simple avec SciLab, mais pas forcément simple si on a pas les outils nécessaires.
Le corrigé : "s03c_interpolation.sce" utilise cette méthode

Une autre méthode utilise les **différences divisées**, dont voici des éléments d'informations pratiques.
On cherche le polynôme passant par les points ax, ay.
On définit :
$d(j,j)$ = ay(j); // pour j=1:length(ax)

$$d(i,j)=\frac{d(i+1,j)-d(i,j-1)}{ax(j)-ax(i)}$$ // pour $1 \le i < j \le$ length(ax)

Que représente d(i, i+1) ?
Quelle devrait être sa valeur si ax(i) ≈ ax(i+1) ?
Que représente d(i-1, i+1) ?
Quelle devrait être sa valeur si ax(i-1) ≈ ax(i) ≈ ax(i+1) ?

Le polynôme d'interpolation se calcule comme suit pour length(ax) = 5. (polynôme de degré 4)
P(x) = *d*(1, 1) + (x - ax(1)) * (d(1,2) + (x - ax(2)) * (d(1,3) + (x - ax(3)) * (d(1,4) + (x - ax(4)) * d(1, 5))));
Une écriture plus efficace est :
P(x) = (((*d*(1, 5) * (x - ax(4)) + d(1,4)) * (x - ax(3)) + d(1,3)) * (x - ax(2)) + d(1,2)) * (x - ax(1)) + d(1, 1);

Le corrigé : "s03c_interpolation_difference_divisee.sce" utilise cette méthode