Pour bénéficier des routines d'interpolation (polynôme, spline, etc) de Scipy nous utiliserons

```
>>> from numpy import *
>>> import matplotlib.pyplot as plt
>>> import scipy.interpolate as inter
```

1. Interpolation polynomiale

La bibliothèque Scipy possède deux routines pour l'interpolation de Lagrange

- `barycentric_interpolate(xi, yi, x)` qui utilise un algorithme stable numériquement. On devine facilement le rôle de `xi` et `yi`, `x` est un tableau et la routine retourne les valeurs du polynôme d'interpolation de Lagrange en les valeurs de `x`.
- `lagrange(x,w)` numériquement instable (c'est la documentation de Scipy qui le dit), visiblement ne rien espérer au delà de 20 points même si ceux-ci sont bien choisis (comme ceux de Tchebychev)! Cette routine retourne un polynôme, c.-à-d. un objet de type `poly1d` et qui accepte d'être évalué, etc.

Voici les deux routines en action.

```
>>> x=array([0,1,2])
>>> w=array([4,-2,4.])
>>> p=inter.lagrange(x,w)
>>> print p
>>> print p(0), p(1), p(2)
>>> xpt=linspace(-1,3,40)
>>> plt.plot(xpt,p(xpt),'k')
>>> w=w+1 # on décale les points yi vers le haut
>>> plt.plot(xpt,inter.barycentric_interpolate(x,w,xpt))
```

Exercice 1. [mise en défaut de interpolate] Soit $f(x) = \cos(x)$ et soit p_n le polynôme d'interpolation de Lagrange aux points équidistants de l'intervalle $[-1,1]$. Comme les dérivées successives de la fonction cos sont bornées, on démontre que la suite des polynômes p_n converge uniformément sur $[-1,1]$ vers la fonction $\cos(x)$. Écrire une routine Python `defaut(n)` qui prend en argument n, calcule via les deux routines `inter.lagrange` et `inter.barycentric_interpolate` le polynôme p_n, trace les deux polynômes (théoriquement semblables) et la fonction $\cos(x)$. Observez ce qui se passe à partir de $n = 30$.

Dorénavant pour toute interpolation de Lagrange nous utiliserons `inter.barycentric_interpolate`

Exercice 2. [Facultatif] Programmer les différences divisées de Newton, la méthode d'Hörner pour l'évaluation du polynôme d'interpolation de Lagrange. Tester cette méthode avec les deux autres.

Exercice 3. [Effet de Runge] Mettre en évidence l'effet de Runge annoncé en cours ($f(x) = 1/(1+x^2)$ sur $[-5,5]$ pour les points équidistants). Vérifier que le phénomène ne se produit pas pour les points de Tchebychev.

Exercice 4. [Une autre fonction] Sur le même modèle que l'exercice précédent pour $f(x) = |x|$ sur l'intervalle $[-1,1]$ tester l'interpolation aux points équidistants et aux points de Tchebychev.

2. Splines

Scipy contient des éléments évolués sur les splines (B-splines, 2D, etc) mais pas les splines naturelles, encastrées, «not-a-knot» ou périodique (c'est possible et même souhaitable mais il faut y consacrer du temps). Nous disposons tout de même de plusieurs solutions «splines» clés en main :

- `UnivariateSpline` qui fait du «fitting» de données ou de l'interpolation (en précisant un paramètre)
- `InterpolatedUnivariateSpline` sous classe de la précédente dédiée à l'interpolation
- `spline` qui possède en théorie une option `natural, not-a-knot, periodic` mais en réalité non implémentée!
- splrep, splev, sproot, spint, spalde - an older wrapping of FITPACK

Nous nous contenterons de `InterpolatedUnivariateSpline` pour faire des tests sur les mêmes exemples que précédemment et en plus calculer une approximation de la dérivée, de l'intégrale. Voici un exemple

```
>>> x=linspace(-3,3,10)
>>> y=exp(-x**2)
>>> s=inter.InterpolatedUnivariateSpline(x,y)
>>> s(x)-y
>>> xpt=linspace(-3,3,100)
>>> plt.plot(xpt,exp(-xpt**2))
>>> plt.plot(xpt,s(exp))
```

Devinez le rôle des fonctions

```
>>> s.derivatives(1)
>>> s.integral(-3,3)
```

Exercice 5. Reprendre les tests de l'interpolation et vérifier qu'il n'y a pas d'effet de Runge.

Exercice 6. Pour la fonction sin vérifier expérimentalement les résultats annoncés en cours sur l'ordre d'approximation des dérivées successives. Pour cela on prendra l'intervalle $[-1,1]$ et $x=0$ pour étudier le comportement de $s'(0)-1$, $s''(0)$ et $s'''(0)+1$ en fonction du pas de temps. Une bonne observation consiste à regarder pour plusieurs valeurs de n (de 10 à 500 ou plus) le logarithme de la quantité à observer divisée par $\log(n)$.

Exercice 7. Pour la fonction cos et pour différentes valeurs de n on créer un spline interpolant sur $[0,1]$ pour les $n+1$ points équidistants i/n $(0 \leq i \leq n)$. Comparer la valeur retournée par `s.integral(0,pi)` et $\int_0^\pi \cos(x)dx$.