TP N°17 : Le module incontournable du calcul scientifique : Numpy
Les fonctions universelles

On importera la bibliothèque Numpy au moyen de la commande : `import numpy as np`

Introduction

Une *fonction universelle* (le terme exact est « ufunc », abréviation de *universal function*) est une fonction qui peut s'appliquer terme à terme aux éléments d'un tableau.
Si f est une *ufunc* et si $a = [a_0, a_1, ..., a_{n-1}]$ est un tableau, alors $f(a)$ renvoie le tableau $[f(a_0), f(a_1), ..., f(a_{n-1})]$.
Les a_k peuvent être des tableaux, par exemple les lignes d'une matrice m. La fonction f s'applique alors récursivement aux éléments de m.
Un grand nombre de fonctions usuelles sont directement « universalisées » dans numpy. Les fonctions mathématiques gardent le même nom (préfixé par np si on a importé numpy par `import numpy as np`). Exemple : `np.exp(t)`
Si on effectue une opération arithmétique entre un tableau a et un scalaire x, tout se passe comme si l'opération arithmétique est effectuée entre x et chaque élément de tableau a.
Pour la documentation officielle en anglais, voir : http ://docs.scipy.org/doc/numpy/reference/ufuncs.html
Le mieux est de commencer par quelques opérations arithmétiques simples sur un vecteur ou une matrice.

Les opérations arithmétiques

Les opérations $+, *, /, ==$, etc. s'appliquent aux tableaux numpy, mais TERME à TERME.
L'expression `a * b` renvoie le tableau des produits terme à terme des éléments de a et b : elle ne désigne donc pas un produit matriciel au sens où on l'entend habituellement.

```
a = np.random.randint(0, 10, (2, 5))
-a ; negative(a) #tableaux des opposés
b = np.random.randint(1, 10, (2, 5)); print(b)
a + b ; np.add(a,b)  # additionne terme à terme les éléments de a et b
a - b ; np.subtract(a,b) #  soustrait terme à terme les éléments de b à ceux de a
a * b ; np.multiply(a,b) # multiplie terme à terme les éléments de a et b
a // b ; np.floor_divide(a,b) #  quotients (entiers) des divisions des éléments de a par ceux de b
a % b  ; np.mod(a,b) # donne les restes dans les divisions des éléments de a par ceux de b
a / b ; np.divide(a,b) # quotients (flottants) terme à terme des éléments de a par ceux de b
a ** b ; np.power(a,b) # élever les éléments de a à la puissance les éléments de b
a == b
```

Fonctions mathématiques usuelles

Toutes les fonctions mathématiques usuelles sont présentes sous forme universelle dans le module numpy (il faut bien penser à les préfixer par np).

- `log log10 log2` : logarithme népérien (resp. de base 10, de base 2).
- `degrees(t) (ou encore rad2deg)` : convertit les angles t de radians en degrés.
- `radians(t) (ou encore deg2rad)` : convertit les angles t de degrés en radians.
- fonctions trigonométriques : `sin, cos, tan, arcsin, arccos, arctan, sinh, cosh, tanh, arcsinh, arccosh, arctanh`

Exemple :

```
rad = np.array([pi/6, pi/4, pi/3, pi/2, pi])
deg = np.degrees(rad); # deg = [  30.,   45.,   60.,   90.,  180.]
```

Vectorisation d'une fonction

Pour qu'une fonction f puisse s'appliquer, terme à terme, à tous les éléments d'un ou de plusieurs tableaux (selon que f accepte un ou plusieurs arguments), il faut la *« vectoriser »*.
Exemple 1 : Considérons la fonction $f : x \longmapsto \dfrac{x + sin(x)}{x + cos(x)}$

```
def f(x): return (x+sin(x))/(x+cos(x))
a = np.arange(1,7)
f(a) # Tenter d'appliquer la fonction f à un tableau numpy conduit à une erreur
```

Il faut donc utiliser la fonction `vectorize` pour rendre f universelle.

```
vf = np.vectorize(f) # vf estt la version vectorisée de f
vf(a) # on peut applique f terme à terme aux éléments de a
(a+np.sin(a))/(a+np.cos(a)) # même résultat avec des opérations usuelles (toutes déjà vectorisées)
```

Exemple 2 :

```
def f(x,y):
        return 0 if x < y else y
vf = np.vectorize(f)
a = np.arange(1,7); a # array([1, 2, 3, 4, 5, 6])
b = np.array([4,1,8,7,2,9]); b # array([4, 1, 8, 7, 2, 9])
vf(a,b) # array([0, 1, 0, 0, 2, 0])
vf(b,a) # array([1, 0, 3, 4, 0, 6])
```

En continuant sur cet exemple, on voit bien que la version vectorisée de f agit comme une « *ufunc* » (fonction universelle).

```
vf(a,3) #array([0, 0, 3, 3, 3, 3])
vf(3,b) # array([0, 1, 0, 0, 2, 0])
vf(5,[[1,6,2],[3,4,8],[6,3,2]]) # array([[1, 0, 2],[3, 4, 0],[0, 3, 2]])
```

Exercice 1

1. Soit la fonction $h(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}$. Créer des tableaux xt et ht qui contiennent respectivement 41 valeurs reparties uniformement pour $x \in [-4,4]$ et l'évaluation de la fonction h sur ces valeurs.