Reprendre les trois exercices traités en cours et écrire les programmes correspondants. Les tester sur des exemples. Pour l'exercice trois utiliser la fonction proposée.

Exercice 1

1) Ecrire une fonction **norme(X) :** qui étant donné un vecteur **X** (numpy) retourne un vecteur unitaire **Y** colinéaire à **X**.

2) Ecrire une fonction **gramschmidt(A) :**, **A :** étant une matrice $n \times m$ dont les n lignes représente des vecteurs de $\mathbb{R}^m$, avec $n \leq m$, qui retourne une matrice **B** dont les lignes représentent une famille orthonormale de n vecteurs.

Exercice 2

La méthode d'Euler de résolution approchée des équations différentielles repose sur une approximation d'intégrale par la méthode des rectangles, la méthode d'heun repose sur une approximation d'intégrale par la méthode des trapèzes, bien plus précise.
En exercice nous allons étudier la méthode de Runge Kutta d'ordre quatre, qui repose sur une approximation d'intégrale par la méthode de Simpson :

$$\int_{t_k}^{t_{k+1}} g(t)dt \approx \frac{1}{6}g(t_k) + \frac{2}{6}g\left(\frac{t_k + t_{k+1}}{2}\right) + \frac{2}{6}g\left(\frac{t_k + t_{k+1}}{2}\right) + \frac{1}{6}g(t_{k+1}).$$

Le schéma de cette méthode est le suivant pour donner une solution approchée de l'équation (E) : $y' = f(t, y)$.

$p_1 = f(t_n, y_n),\ p_2 = f(t_n + \frac{h}{2}, y_n + \frac{h}{2}p_1),\ p_3 = f(t_n + \frac{h}{2}, y_n + \frac{h}{2}p_2),\ p_4 = f(t_{n+1}, y_n + hp_3)$ où $t_{n+1} = t + h$, puis pour terminer $y_{n+1} = y_n + \frac{1}{6}(p_1 + 2p_2 + 2p_3 + p_4)$.

Ecrire une fonction **def rungekutta(f,t_0,y_0, T,N)**, les notations étant celles de la méthode d'Euler, qui renvoie deux tableaux X et Y où $Y[t_k]$ est la valeur approchée de y en $t_k = X[k] = t_0 + kh$, h étant le pas de la subdivision.

Exercice 3 :système proie prédateurs

Lotka et Volterra ont proposé au début du vingtième siècle un modèle simple de relations entre proies et prédateurs. $x_1(t)$ représente la population à l'instant t des proies et $x_2(t)$ celle des prédateurs. L'évolution est décrite de la manière suivante :

$x_1'(t) = r * x_1(t) - p * x_1(t)x_2(t),\ \ x_2'(t) = -dx_2(t) + qx_1(t)x_2(t)$.
r étant le taux d'accroissement intrinsèque des proies sans prédation et d indique qu'en l'absence de proies la populaton des prédateurs périclite, ces lois décrivent donc un équilibre naturel entre ces populations. On veut étudier le système suivant :

$$\begin{cases} x_1'(t) &= 2.5\, x_1(t) - 0.05\, x_1(t)x_2(t) \\ x_2'(t) &= -0.8\, x_1(t) + 0.05\, x_1(t)x_2(t) \\ x_1(t_0) &= a \\ x_2(t_0) &= b \end{cases}$$

où $r = 2.5,\ p = q = 0.05,\ d = 0.8$
On définit la fonction f par :

def $f(t,y)$:
return array($[2.5*y[0]-0.05*y[1]*y[0], -0.8*y[0]+0.05*y[1]*y[0]]$)

où y est donné par une matrice colonne de deux lignes.

Ecire une fonction **euler(f,t0,y0,T,N)** qui renvoie une matrice de type $2 \times (N+1)$ contenant les valeurs de $x_1(t_k)$ en première ligne et les valeurs de $x_2(t_k)$ en deuxième ligne. la k-ième étape on écrira $Y[:,k+1] = Y[:k] + h * F(X[k], Y[:,k])$ où $Y[:,t_k]$ est la colonne d'ordre k.

Exercice 4

Ecrire les fonctions correspondant aux méthodes d'Euler, de Heun, du point milieu et de Runge Kutta. appliquer chacune des méthodes à la fonction définie par $f(t,y) = y$ avec t0=0,T=1,y0=0et N=100. Comparer $Y[101]$ à la valeur de e dans chaque méthode.

Exercice 5

Tracer les courbes correspondantes sur une même représentation : importer mathplotlib puis pylab. Les abscisses sont obtenues, dans la méthode d'e'euler, par exemple, par **X=euler(f,t0,y0,T,N)**[0] où les paramètres prennent les valeurs précédentes, les ordonnées sont obtenues par **Y=euler(f,t0,y0,T,N)**[0], le tracé de la courbe est obtenu par **plot(X,Y,color='blue'** blue étant donné pour l'exemple.

Exercice 6

On écrit la fonction d'Euler pour obtenir des ordonnées dans une liste :

```
def euler2(f,t0,y0,T,N):
    h=T/N
    X=t0
    t=t0
    u=y0
    Y=[y0]
    for k in range(N):
        v=u+h*f(t,u)
        Y.append(v)
        t=t+h
        u=v
    return Y
```

Sous cette forme il est possible de résoudre des équations différentielles du second ordre, par exemple $y'' = -\sin y$:

```
def fpendule(t, [y,yp]):
return [yp, -sin(y)]
```

où yp représente y'.

$Y = euler(fpendule, 0, np.array([0,3]), 10, 100)$ Y est alors une liste de couples (y, y') où y est solution de $y" = -\sin y$ avec les conditions initiales $y(0) = 0,\ y'(0) = 3$.
On peut représenter la solution approchée par :

```
t = np.linspace(0, 10, 101)
plot(t, Y[:, 0]) # on prend la première composante
```

Représenter les solutions de cette équation sur $[0, 30]$ pour différentes vitesses et différentes valeurs de n.