

Informatique : DM sur la dynamique gravitationnelle

Gatt Guillaume

I Quelques fonctions utilitaires

1. a) $[1,2,3] + [4,5,6] = [1,2,3,4,5,6]$

b) $2 * [1,2,3] = [1,2,3,1,2,3]$

#Question 1.2

```
def smul(n,L) :  
    T=[]  
    for i in range(len(L)) :  
        T.append(n*L[i])  
    return T
```

#Question 1.3 a)

```
def vsom(L_1,L_2) :  
    T=[]  
    for i in range(len(L_1)) :  
        T.append(L_1[i] +L_2[i])  
    return T
```

#Question 1.3 b)

```
def vdiff(L_1,L_2) :  
    T=[]  
    for i in range(len(L_1)) :  
        T.append(L_1[i] -L_2[i])  
    return T
```

II Etude de schémas numériques

II.1 Mise en forme du problème

1. On peut écrire l'équation (1) sous la forme :

$\forall t \in I, z'(t) = f(y(t))$ car $\forall t \in I, z(t) = y'(t)$

2. On a :

$$y(t_{i+1}) = y(t_i) + (y(t_{i+1}) - y(t_i))$$

$$\text{Or : } (y(t_{i+1}) - y(t_i)) = \int_{t_i}^{t_{i+1}} y'(t) dt$$

$$y(t_{i+1}) = y(t_i) + \int_{t_i}^{t_{i+1}} z(t) dt \text{ car } \forall t \in I, z(t) = y'(t)$$

de même pour $z(t)$:

$$z(t_{i+1}) = z(t_i) + (z(t_{i+1}) - z(t_i))$$

$$\text{Or : } (z(t_{i+1}) - z(t_i)) = \int_{t_i}^{t_{i+1}} z'(t) dt$$

$$z(t_{i+1}) = z(t_i) + \int_{t_i}^{t_{i+1}} f(y(t)) dt \text{ car } \forall t \in I, z'(t) = f(y(t))$$

II.2 Schéma d'Euler explicite

1. On approche les intégrales par la méthode des rectangles à gauche :

$$y_{i+1} = y_i + h z_i = y_0 + h \sum_{k=0}^i z_k$$

$$z_{i+1} = z_i + hf(y_i) = z_0 + h \sum_{k=0}^i f(y_k)$$

#Question 11.2.2

```
def euler(f,t_min,t_max,n,z0,y0) :
    h=(t_max -t_min)/(n-1)
    T=[]
    for i in range(n) :
        T.append(t_min + i*h)
    z=z_0
    y=y_0
    Z=[z_0]
    Y=[y_0]
    for i in range(len(T)) :
        z,y= z + (T[i+1] - T[i])*f(y),y + (T[i+1] - T[i]) * z
        Z.append(z)
        Y.append(y)
    return (Z,Y)
```

La fonction prend en paramètre f, la fonction donnée par l'équation différentielle afin de calculer les (z_i) , t_{min}, t_{max} et n, pour obtenir la liste des (t_i) afin de pouvoir calculer de manière approchée les intégrales, ainsi que z_0 et y_0 afin de pouvoir initialiser la récurrence.

3. a) On a l'équation :

$$y'' + \omega^2 y(t) = 0$$

$$y''(t) \times y'(t) + y'(t) \times \omega^2 y(t) = 0$$

On reconnaît $g'(x) = -f(x) = \omega^2 x$:

$$y''(t) \times y'(t) + y'(t) \times g'(y(t)) = 0$$

On primitive l'équation avec une constante d'intégration $-E$:

$$\frac{1}{2}(y'(t))^2 + g(y(t)) - E = 0$$

$$\frac{1}{2}(y'(t))^2 + g(y(t)) = E$$

b) On a en remplaçant $y(t)$ par y_i et $y'(t)$ par z_i :

$$E_i = \frac{1}{2}(z_i)^2 + g(y_i)$$

On a donc :

$$E_{i+1} - E_i = \left(\frac{1}{2}(z_{i+1})^2 + g(y_{i+1})\right) - \left(\frac{1}{2}(z_i)^2 + g(y_i)\right)$$

$$E_{i+1} - E_i = \frac{1}{2}((z_{i+1})^2 - (z_i)^2) + g(y_{i+1}) - g(y_i)$$

Comme $g' = -f$, $g(x) = \frac{1}{2}\omega^2 x^2$ on a :

$$E_{i+1} - E_i = h^2 \omega^2 \left(\frac{\omega^2 y_i^2}{2} + \frac{z_i^2}{2}\right)$$

$$E_{i+1} - E_i = h^2 \omega^2 \left(\frac{1}{2}(z_i)^2 + g(y_i)\right)$$

$$E_{i+1} - E_i = h^2 \omega^2 E_i$$

c) On aurait :

$$\forall i, \frac{1}{2}(z_i)^2 + g(y_i) = E$$

d) On a si conservation de E :

$$2E = (z_i)^2 + \omega^2 y_i^2$$

Si $E \neq 0$:

$$\left(\frac{z_i}{\sqrt{2E}}\right)^2 + \left(\frac{\omega y_i}{\sqrt{2E}}\right)^2 = 1$$

On a une ellipse

Si $E = 0$:

$z_i^2 = -\omega^2 y_i^2$ Or $(z_i)^2 \geq 0$ et $\omega^2 y_i^2 \geq 0$ donc $z_i = 0$ et $y_i = 0$ on obtient un point de coordonnées $(0; 0)$

e) Si le schéma numérique satisfaisait la conservation de E on aurait une ellipse ou un point or on a une spirale. Comme les y_i et z_i ont une valeur plus grande que leurs valeurs réelles $y(t_i)$ et $z(t_i)$ l'écart entre la solution et le calcul approché s'agrandit à chaque calcul ce qui donne une spirale qui s'éloigne du centre.

II.3 Schéma de Verlet

#Question 11.3.1

```

def verlet(f,t_min,t_max,n,z0,y0):
    h=(t_max-t_min)/(n-1)
    T=[]
    for i in range(n):
        T.append(t_min+i*h)
    z=z_0
    y=y_0
    Z=[z_0]
    Y=[y_0]
    for i in range(len(T)):
        y= y + h*z + (h**2)/2*f(y_i)
        z= z+ h/2*(f(Y[-1])+f(y))
        Z.append(z)
        Y.append(y)
    return (Z,Y)

```

2. a) On a :

$$\begin{aligned}
 E_{i+1} - E_i &= \frac{1}{2}((z_{i+1} - z_i)(z_{i+1} + z_i)) + \frac{\omega^2}{2}((y_{i+1} - y_i)(y_{i+1} + y_i)) \\
 E_{i+1} - E_i &= \frac{h}{4}(f_i + f_{i+1})(2z_i + \frac{h}{2}(f_i + f_{i+1})) + \frac{\omega^2}{2}((hz_i + \frac{h^2}{2}f_i)(2y_i + hz_i + \frac{h^2}{2}f_i)) \\
 E_{i+1} - E_i &= \frac{h}{4}(f_i + f_{i+1})(2z_i + \frac{h}{2}(f_i + f_{i+1})) + \frac{\omega^2}{2}((hz_i + \frac{h^2}{2}f_i)(2y_i + hz_i + \frac{h^2}{2}f_i)) \\
 E_{i+1} - E_i &= \frac{h}{4}(f_i + f_{i+1})(2z_i + \frac{h}{2}(f_i + f_{i+1})) + \frac{\omega^2}{2}(h^2z_i^2 + f_i h^2 y_i + 2h y_i z_i + O(h^3)) \\
 \text{Or } f_{i+1} &= -\omega^2(y_i + hz_i + \frac{h^2}{2}f_i) \\
 E_{i+1} - E_i &= \frac{h}{4}(f_i - \omega^2(y_i + hz_i + \frac{h^2}{2}f_i))(2z_i + \frac{h}{2}(f_i - \omega^2(y_i + hz_i + \frac{h^2}{2}f_i))) + \frac{\omega^2}{2}(h^2z_i^2 + f_i h^2 y_i + 2h y_i z_i + O(h^3)) \\
 E_{i+1} - E_i &= \frac{1}{4}(hf_i - \omega^2(hy_i + h^2z_i + O(h^3)))(2z_i + \frac{1}{2}(hf_i - \omega^2(hy_i + h^2z_i + O(h^3)))) + \frac{\omega^2}{2}(h^2z_i^2 + f_i h^2 y_i + 2h y_i z_i + O(h^3)) \\
 E_{i+1} - E_i &= \frac{1}{8}(hf_i - \omega^2(hy_i + h^2z_i + O(h^3)))(hf_i - \omega^2(hy_i + h^2z_i + O(h^3))) + \frac{\omega^2}{2}(f_i h^2 y_i + O(h^3)) \text{ car } f(y_i) = -\omega^2 y_i \\
 E_{i+1} - E_i &= \frac{f_i}{8}(h^2 f_i - \omega^2(h^2 y_i + O(h^3))) + \frac{\omega^2}{2}(\frac{2f_i h^2 y_i}{4} + O(h^3)) \\
 E_{i+1} - E_i &= O(h^3) \text{ car } f(y_i) = -\omega^2 y_i
 \end{aligned}$$

b) L'allure du graphique est une ellipse ce qui est en accord avec une conservation de E.

c) Le schéma de Verlet conserve E

III Problème à N corps

III.1 Position du problème

$$1. \vec{F}_j = \sum_{k \neq j} \vec{F}_{k/j}$$

#Question III.1.2

```

def force2(m1,p1,m2,p2):
    G=6.67e-11
    p1p2= vdiff(p2,p1)
    r=sqrt(p1p2[0]**2+p1p2[1]**2+p1p2[2]**2)
    return (smul(G*m1*m2/(r**3),p1p2))

```

#Question III.1.3

```

def forceN(j,m,pos):
    m1=m[j]
    p1=pos[j]
    F=[0,0,0]
    for i in range(len(m)):
        if i!=j:
            vsom(F,force2(m1,p1,m[i],pos[i]))
    return F

```

III.2 Approche numérique

1. **position[i]** donne la position des objets au temps t_i , **vitesse[i]** donne la vitesse des objets au temps t_i
2. On a par le principe fondamentale de la dynamique appliqué à un corps j avec O l'origine du repère :

$$m_j(\overrightarrow{OP_j})'' = \vec{F}_j$$

$$(\overrightarrow{OP_j})'' = \frac{\vec{F}_j}{m_j}$$

On a donc comme l'équation (1) avec $f = \frac{\vec{F}_j}{m_j}$

#Question III.2.2

```
def euler_3d(f, t_min, t_max, n, z0, y0) :  
    h=(t_max -t_min)/(n-1)  
    T=[]  
    for i in range(n) :  
        T.append(t_min + i*h)  
    z=z_0  
    y=y_0  
    Z=[z_0]  
    Y=[y_0]  
    for i in range(len(T)) :  
        z,y= vsom(z, smul((T[i+1] - T[i]), f(y))), vsom(y, smul((T[i+1] - T[i]), z))  
        Z.append(z)  
        Y.append(y)  
    return (Z,Y)  
  
def pos_suiv(m, pos, vit, h) :  
    pos2=[]  
    vit2=[]  
    def f(y) :  
        return forceN(y,m,pos)  
    for i in range(len(pos)) :  
        solution=euler_3d(f,0,h,2, vit[i], pos[i])  
        pos2.append( solution[1][1])  
        vit2.append( solution[0][1])  
    return (vit2, pos2)
```

#Question III.2.3

```
def verlet_3d(f, t_min, t_max, n, z0, y0) :  
    h=(t_max -t_min)/(n-1)  
    T=[]  
    for i in range(n) :  
        T.append(t_min + i*h)  
    z=z_0  
    y=y_0  
    Z=[z_0]  
    Y=[y_0]  
    for i in range(len(T)) :  
        y= vsom(vsom(y, smul(h, z)), smul((h**2)/2, f(y)))  
        z= vsom(z, smul(h/2, vsom(f(Y[-1]), f(y))))  
        Z.append(z)  
        Y.append(y)  
    return (Z,Y)  
  
def etat_suiv(m, pos, vit, h) :  
    pos2=[]  
    vit2=[]  
    def f(j) :  
        return (forceN(j, m, pos)/m[j])
```

```

for i in range(len(pos)) :
    solution=verlet_3d(f,0,h,2,vit[i],pos[i])
    vit2.append(solution[0][1])
    pos2.append(solution[1][1])
return (vit2 ,pos2)

```

4.a) $\ln(\tau_N) = K \ln(N)$ avec $K = 2$

b) on a donc une complexité en $O(N^2)$.

5. a) état suiv a une complexité en $O(N^3)$

b) La complexité est plus grande que celle obtenue en III.2.4

#Question III.2.6

```

def simulation_verlet(deltat,n) :
    pos=p0
    vit=v0
    position=[p0]
    for i in range(n) :
        solutions=etat_suiv(m,pos,vit,deltat)
        vit=solutions[0]
        pos=solutions[1]
        position.append(pos)
    return position

```