Système solaires : équations différentielles

1. Introduction

Ce document montre la mise en équation du mouvement des planètes dans le système solaire. L'objectif est de faire une intégration numérique de ces équations, à partir de conditions initiales fournies par des éphémérides.

2. Équations du mouvement

Dans tout ce qui suit, on entend par planète tout corps du système solaire autre que le Soleil, y compris les satellites comme la Lune, les comètes, les astéroïdes.

L'application des lois de la mécanique classique au système solaire suppose l'existence d'un référentiel inertiel à l'échelle du système solaire. Le système solaire (Soleil+planètes) étant soumis à l'influence gravitationnelle des autres corps de la galaxie, le centre de masse C du système solaire est nécessairement fixe dans ce référentiel. On appelle *référentiel barycentrique* (RB) le référentiel inertiel dans lequel le centre de masse C est fixe. En pratique, ce référentiel est réalisé avec le centre C et deux étoiles assez lointaines pour que leur mouvement soit négligeable sur l'échelle de temps considérée (de l'ordre du siècle). Sachant que l'étoile la plus proche du système solaire est à une distance environ 6000 fois plus grande que celui-ci, le référentiel RB peut être considéré en mouvement de chute libre dans un champ uniforme. Autrement dit, les forces de marée sont négligeables dans ce référentiel et l'on peut ignorer complètement l'influence gravitationnelle des autres corps de la galaxie.

Le référentiel RB est peu pratique car le centre C n'est pas directement observable. On introduit donc le référentiel héliocentrique RH, en mouvement de translation par rapport à RB. RH n'est pas tout à fait galiléen, puisqu'il est accéléré par rapport à RB. Il faudra donc prendre en compte les forces d'inertie d'entraînement dans ce référentiel. Ce réferentiel RH est réalisé par le centre du Soleil et deux étoiles assez lointaines (les trois non alignés pour définir un système rigide de référence).

On introduit aussi le référentiel Terre-Lune (RTL) en translation par rapport à RB mais centré sur le centre de masse du système Terre-Lune, et le référentiel géocentrique (RG) en translation par rapport à RB mais centré sur le centre de masse de la Terre.

Soit M_s la masse du Soleil et m_i la masse des planètes, l'indice i variant de 1 à N, où N est le nombre de planètes considérées. La constante de gravitation est notée G. La position du centre du Soleil est S, celle du centre de la planète i est P_i. Considérons tout d'abord l'accélération du centre du Soleil dans le référentiel inertiel RB. Le Soleil est soumis à l'influence gravitationnelle des planètes donc :

$$\overrightarrow{a}_{S/RB} = \sum_{j=1}^{N} -Gm_j \frac{\overrightarrow{P_jS}}{P_jS^3} \tag{1}$$

Dans le référentiel non inertiel RH (en translation par rapport à RB), il y a une force d'inertie par unité de masse égale à l'opposée de cette accélération.

En tenant compte de cette force d'inertie et des forces gravitationnelles directes, l'accélération d'une planète dans le référentiel RH s'écrit :

$$\overrightarrow{a_i} = -GM_s\frac{\overrightarrow{SP_i}}{SP_i^3} - \sum_{j\neq i} Gm_j\frac{\overrightarrow{P_jP_i}}{P_jP_i^3} - \sum_j Gm_j\frac{\overrightarrow{SP_j}}{SP_j^3} \tag{2}$$

En sortant le terme de la planète considérée dans la force d'inertie, on obtient :

$$\overrightarrow{a_i} = -G(M_s+m_i)\frac{\overrightarrow{SP_i}}{SP_i^3} - \sum_{j\neq i} Gm_j\left(\frac{\overrightarrow{P_jP_i}}{P_jP_i^3} + \frac{\overrightarrow{SP_j}}{SP_j^3}\right) \tag{3}$$

3. Unités astronomiques internationales

Des unités astronomiques ont été définies :

- ▷ L'unité de temps est le jour (j), soit 86400 s.
- ▷ L'unité de masse est la masse du Soleil M_s.
- ▷ L'unité de longueur astronomique (UA) est le rayon de l'orbite circulaire d'une planète de masse négligeable et non perturbée, dont la vitesse angulaire autour du Soleil est de k radian par jour, où k est la constante de Gauss.

La constante de Gauss a été fixée en 1938 pour que l'UA soit égale au demi-grand axe de l'orbite terrestre à cette date. Sa valeur est :

$$k = 0{,}01720209895\,\text{rad}\cdot\text{j}^{-1} \tag{4}$$

Sur une échelle de temps de plusieurs siècles, le demi-grand axe de l'orbite terrestre évolue très lentement, mais la masse du Soleil reste constante, et donc l'unité astronomique de change pas. La troisième loi de Kepler s'écrit :

$$GM_s = a^3k^2 \tag{5}$$

Si l'on utilise les unités astronomiques, on a donc :

$$GM_s = k^2 = 2{,}95912208286\cdot10^{-4}\,\text{UA}^3\cdot\text{j}^{-2} \tag{6}$$

Pour écrire les équations différentielles, il faut les masses des planètes exprimées en unité de masse solaire. Voici les rapports de la masse du Soleil sur la masse des planètes principales du système Solaire ([1]). Pour les planètes comportant des satellites, c'est la masse totale du sytème planète-satellite qui est donnée (à l'exception de la Terre).

Planete	Ms/mi
Mercure	6023600.0
Vénus	408523.5
Terre	332946.0
Terre-Lune	328900.5
Lune	27068620.9
Mars	3098710.0
Jupiter	1047.355
Saturne	3498.5
Uranus	22869.0
Neptune	19314.0

L'inverse de ces valeurs donne la masse m_i de chaque planète exprimée en masse solaire.

Voici finalement l'expression de l'accélération d'une planète dans le système d'unités astronomiques :

$$\overrightarrow{a_i} = -k^2(1+m_i)\frac{\overrightarrow{SP_i}}{SP_i^3} - k^2\sum_{j\neq i} m_j\left(\frac{\overrightarrow{P_jP_i}}{P_jP_i^3} + \frac{\overrightarrow{SP_j}}{SP_j^3}\right) \tag{7}$$

4. Équations différentielles

On note (x_i, y_i, z_i) les coordonnées de la planète dans un repère lié au référentiel héliocentrique RH. Les composantes de sa vitesse sont notées (u_i, v_i, w_i). Le système étudié comporte N planètes. Pour chaque planète, voici les 6 équations différentielles :

$$\frac{dx_i}{dt} = u_i \tag{8}$$

$$\frac{dy_i}{dt} = v_i \tag{9}$$

$$\frac{dz_i}{dt} = w_i \tag{10}$$

$$\frac{du_i}{dt} = -k^2\left((1+m_i)\frac{x_i}{r_i^3} + \sum_{j\neq i} m_j\left(\frac{x_i - x_j}{r_{ij}^3} + \frac{x_j}{r_j^3}\right)\right) \tag{11}$$

$$\frac{dv_i}{dt} = -k^2\left((1+m_i)\frac{y_i}{r_i^3} + \sum_{j\neq i} m_j\left(\frac{y_i - y_j}{r_{ij}^3} + \frac{y_j}{r_j^3}\right)\right) \tag{12}$$

$$\frac{dw_i}{dt} = -k^2\left((1+m_i)\frac{z_i}{r_i^3} + \sum_{j\neq i} m_j\left(\frac{z_i - z_j}{r_{ij}^3} + \frac{z_j}{r_j^3}\right)\right) \tag{13}$$

où les distances au cube sont :

$$r_i^3 = (x_i^2 + y_i^2 + z_i^2)^{3/2} \tag{14}$$

$$r_{ij}^3 = ((x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2)^{3/2} \tag{15}$$

5. Conditions initiales

L'institut de mécanique céleste et de calcul des éphémérides (IMCCE) fournit les éphémérides des corps du système solaire. On utilise ce service pour obtenir les positions et vitesses initiales des corps.

Les éphémérides de l'IMCCE sont calculées avec des modèles de perturbation, qui utilisent des développements en série pour modéliser les différentes perturbations que les planètes exercent les unes sur les autres. Une méthode d'intégration numérique, similaire à celle que nous mettons en place ici, est utilisée par le Jet Propulsion Laboratory. Ces deux approches (théorie des perturbations et intégration numériques) sont complémentaires et contribuent toutes les deux à une connaissance précise des mouvements dans le système solaire. Le JPL a aussi un service d'éphéméride en ligne (Horizon).

Pour interroger le serveur d'éphémérides de l'IMCCE (Myriade), on peut utiliser le formulaire en ligne. Voici les paramètres à choisir :

- ▷ Target : choix d'une planète ou d'un satellite.
- ▷ Epoch : choix de la date initiale, du nombre de dates, et de l'intervalle de temps.
- ▷ Reference center : heliocenter.
- ▷ Advanced parameters : choisir le plan de référence écliptique et les coordonnées rectangulaires.

Obtenir les données sous forme d'un fichier texte. Voici un exemple, avec 5 dates espacées de 1 jour (réglage par défaut) :

```
# Miriade.ephemcc.results
# Request: targetType: Planet, targetName: Earth, Theory: INPOP13C, Coordinates: Ecli
# Nb rows:
Target, Date, X (au), Y (au), Z (au), Heliocentric distance (au), Xp (au/d), Yp (au/d
Earth,   2016-08-20T00:00:00.00,   0.8503047799995,  -0.5483797111848,   0.0000166509
Earth,   2016-08-21T00:00:00.00,   0.8592264122864,  -0.5339150182548,   0.0000167353
Earth,   2016-08-22T00:00:00.00,   0.8679042411478,  -0.5192976272552,   0.0000167577
Earth,   2016-08-23T00:00:00.00,   0.8763355651493,  -0.5045311829495,   0.0000166809
Earth,   2016-08-24T00:00:00.00,   0.8845176315919,  -0.4896194124351,   0.0000164742
```

Le tableau comporte la date, les coordonnées héliocentriques en unité astronomique (AU), la distance au Soleil, et les composantes héliocentriques de la vitesse, en AU par jour. Voici comment extraire les positions et les vitesses du fichier :

```
import numpy
data = numpy.loadtxt("miriade-earth.txt",skiprows=4,delimiter=",",usecols=(2,3,4,6,7,8
Y_terre = data[0] # condition initiale pour la Terre au 20/8/2016  0h00


print(Y_terre)
--> array([  8.50304780e-01,  -5.48379711e-01,   1.66509015e-05,
         9.04264811e-03,   1.43871423e-02,   1.02031100e-07])
```

Dans un premier temps, nous allons intégrer les équations pour le système Soleil-Terre-Lune. Il faut donc les conditions initiales pour la Lune :

```
# Miriade.ephemcc.results
# Request: targetType: Satellite, targetName: Moon, Theory: INPOP13C, Coordinates: Ec
# Nb rows:
Target, Date, X (au), Y (au), Z (au), Heliocentric distance (au), Xp (au/d), Yp (au/d
Moon,   2016-08-20T00:00:00.00,   0.8527213361762,  -0.5488817445380,  -0.00000626780
Moon,   2016-08-21T00:00:00.00,   0.8616800284951,  -0.5338045999284,  -0.00006112800
Moon,   2016-08-22T00:00:00.00,   0.8702477738636,  -0.5185824053193,  -0.00011132965
Moon,   2016-08-23T00:00:00.00,   0.8784276203895,  -0.5032553942750,  -0.00015383683
Moon,   2016-08-24T00:00:00.00,   0.8862321652082,  -0.4878609016503,  -0.00018613411
```

```
data = numpy.loadtxt("miriade-moon.txt",skiprows=4,delimiter=",",usecols=(2,3,4,6,7,8
Y_lune = data[0] # condition initiale pour la Lune au 20/8/2016  0h00


print(Y_lune)
--> array([  8.52721336e-01,  -5.48881745e-01,  -6.26780840e-06,
         9.15266395e-03,   1.49910873e-02,  -5.60971550e-05])
```

On choisit par convention de ranger les variables dans l'ordre suivant : coordonnées de la Terre, vitesses de la Terre, coordonnées de la Lune, vitesse de la Lune. La condition initiale complète est donc obtenue en concaténant les deux précédentes :

```
Yi = numpy.append(Y_terre,Y_lune)


print(Yi)
--> array([  8.50304780e-01,  -5.48379711e-01,   1.66509015e-05,
         9.04264811e-03,   1.43871423e-02,   1.02031100e-07,
         8.52721336e-01,  -5.48881745e-01,  -6.26780840e-06,
         9.15266395e-03,   1.49910873e-02,  -5.60971550e-05])
```

6. Intégration numérique

6.a. Définition du système différentiel

On commence par introduire les constantes en variables globales :

```
k = 0.01720209895
k2 = k*k
mT = 1.0/332946
mL = 1.0/27068620.9
```

Le système différentiel doit être défini dans une fonction de format `systeme(Y,t)`, où `Y` est un tableau comportant les N=12 variables. La fonction doit renvoyer un tableau comportant les N dérivées. Voici la correspondance entre les noms des variables et les indices du tableau Y :

- ▷ xT,yT,zT : 0,1,2
- ▷ uT,vT,wT : 3,4,5
- ▷ xL,yL,zL : 6,7,8
- ▷ uL,vL,wL : 9,10,11

```
def systeme(Y,t):
    rT3 = numpy.power(Y[0]*Y[0]+Y[1]*Y[1]+Y[2]*Y[2],1.5)
    rL3 = numpy.power(Y[6]*Y[6]+Y[7]*Y[7]+Y[8]*Y[8],1.5)
    xTL = Y[6]-Y[0]
    yTL = Y[7]-Y[1]
    zTL = Y[8]-Y[2]
```

```
    rTL3 = numpy.power(xTL*xTL+yTL*yTL+zTL*zTL,1.5)
    aT = -k2*((1.0+mT)*Y[0]/rT3+mL*(-xTL/rTL3+Y[6]/rL3))
    bT = -k2*((1.0+mT)*Y[1]/rT3+mL*(-yTL/rTL3+Y[7]/rL3))
    cT = -k2*((1.0+mT)*Y[2]/rT3+mL*(-zTL/rTL3+Y[8]/rL3))
    aL = -k2*((1.0+mL)*Y[6]/rL3+mT*(xTL/rTL3+Y[0]/rT3))
    bL = -k2*((1.0+mL)*Y[7]/rL3+mT*(yTL/rTL3+Y[1]/rT3))
    cL = -k2*((1.0+mL)*Y[8]/rL3+mT*(zTL/rTL3+Y[2]/rT3))
    return numpy.array([Y[3],Y[4],Y[5],aT,bT,cT,Y[9],Y[10],Y[11],aL,bL,cL])
```

6.b. Méthode de Bulirsch-Stoer

Pour faire l'intégration numérique, on utilise la méthode de Bulirsch-Stoer, qui est une des meilleures méthodes lorsqu'une grande précision est recherchée et lorsque l'évaluation des dérivées demande beaucoup de calculs. C'est la méthode utilisée par le Bureau des longitudes (([1]) pour le calcul des mouvements des comètes.

On reprend tout d'abord l'implémentation définie dans le document Méthode de Bulirsch-Stoer, qui est constituée de deux fonctions :

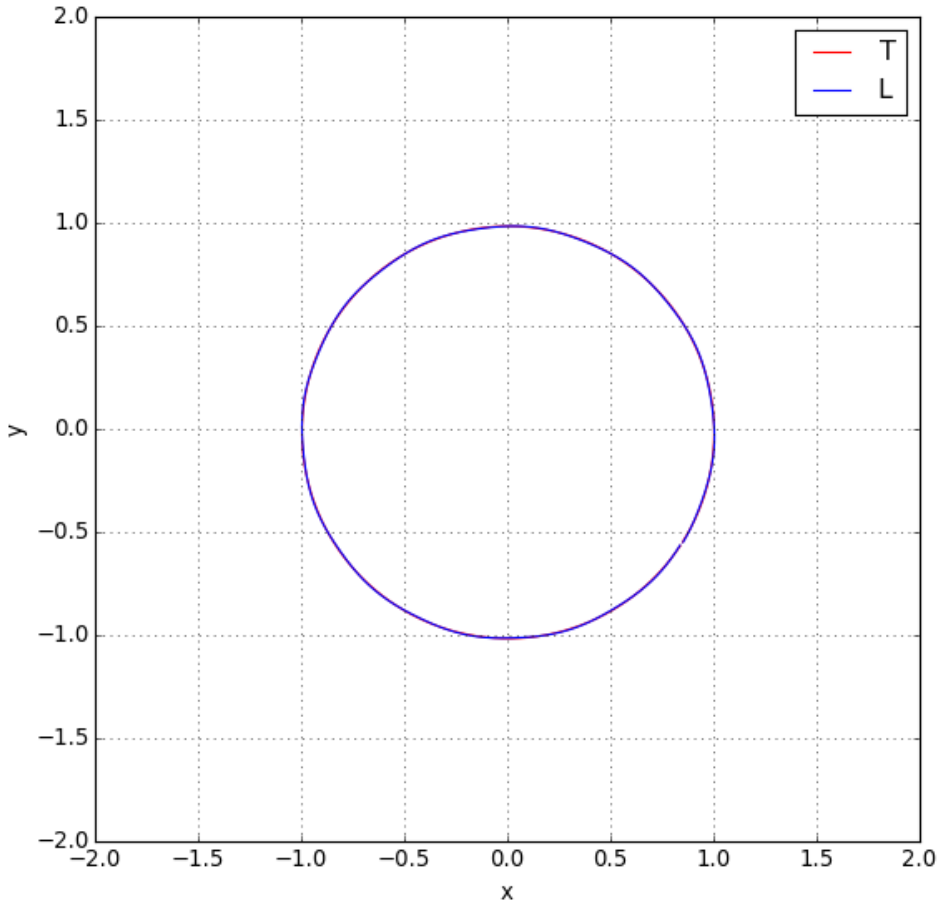
```
def pas_pointmilieu_modifie(systeme,H,t,Yn,m):
    h = H/m
    u = Yn
    v = u+h*systeme(u,t)
    for k in range(1,m):
        w = u+2*h*systeme(v,t+k*h)
        u = v
        v = w
    return 0.5*(v+u+h*systeme(v,t+H))

def bulirsch(systeme,jmax,Yi,T,H,atol=1e-6,rtol=1e-6,S=0.9):
    N = len(Yi)
    A = numpy.zeros((jmax+1,jmax+1,N))
    m = 2*(numpy.arange(jmax+1)+1)
    Y = Yi
    t = 0.0
    liste_y = [Y]
    liste_t = [t]
    liste_e = [0]
    liste_j = [0]
    while t<T:
        for j in range(jmax+1):
            A[j][0] = pas_pointmilieu_modifie(systeme,H,t,Y,m[j])
            for i in range(1,j+1):
                correction = (A[j][i-1]-A[j-1][i-1])/((m[j]*1.0/m[j-i])**2-1)
                A[j][i] = A[j][i-1] + correction
            e=0.0
            for k in range(N):
                e += (abs(A[j][j][k]-A[j][j-1][k])/(atol+rtol*abs(A[j][j][k])))**2
            e = numpy.sqrt(e/N)
```

```
            if e<1.0:
                break
        liste_j.append(j)
        t += H
        Y = numpy.array(A[j][j])
        #H = H*S*(S/e)**(1.0/(2*j+1)) # pas d'adaptation du pas principal
        liste_y.append(Y)
        liste_t.append(t)
        liste_e.append(e)
    return (numpy.array(liste_t),numpy.array(liste_y),numpy.array(liste_e),numpy.arra
```

On a désactivé l'adaptation du pas principal, car on souhaite des résultats avec un pas de temps constant, de manière à faciliter la comparaison avec les éphémérides de l'IMCCE. Il faudra donc choisir un pas de temps H assez petit et contrôler l'estimation de l'erreur. Le nombre de points utilisés pour l'extrapolation polynomiale est variable.

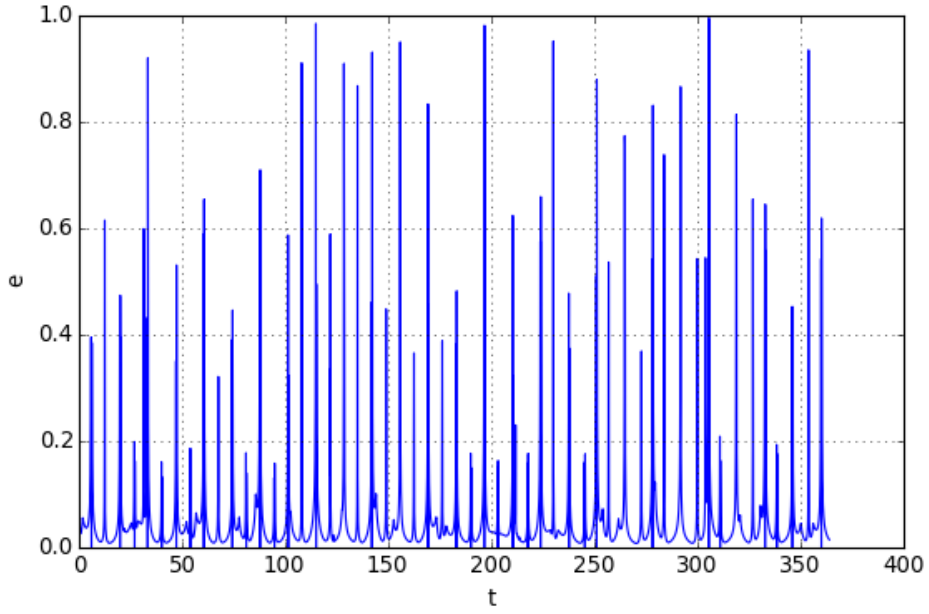
Voici un calcul fait sur une durée de 364 jours, avec un pas de temps H de 0.1 jour. On choisit une tolérance relative, avec une tolérance absolue de sécurité (si une variable s'annule). La trajectoire est tracée en projection dans le plan XY.

```
from matplotlib.pyplot import *
T = 364.0
H = 0.1
jmax = 10
atol=1e-12
rtol=1e-5
(t,tab_y,e,J) = bulirsch(systeme,jmax,Yi,T,H,atol,rtol)
Y = tab_y.transpose()
figure(figsize=(8,8))
plot(Y[0],Y[1],'r',label="T")
plot(Y[6],Y[7],'b',label="L")
xlabel("x")
ylabel("y")
axes().set_aspect('equal')
axis([-2,2,-2,2])
legend(loc="upper right")
grid()
```



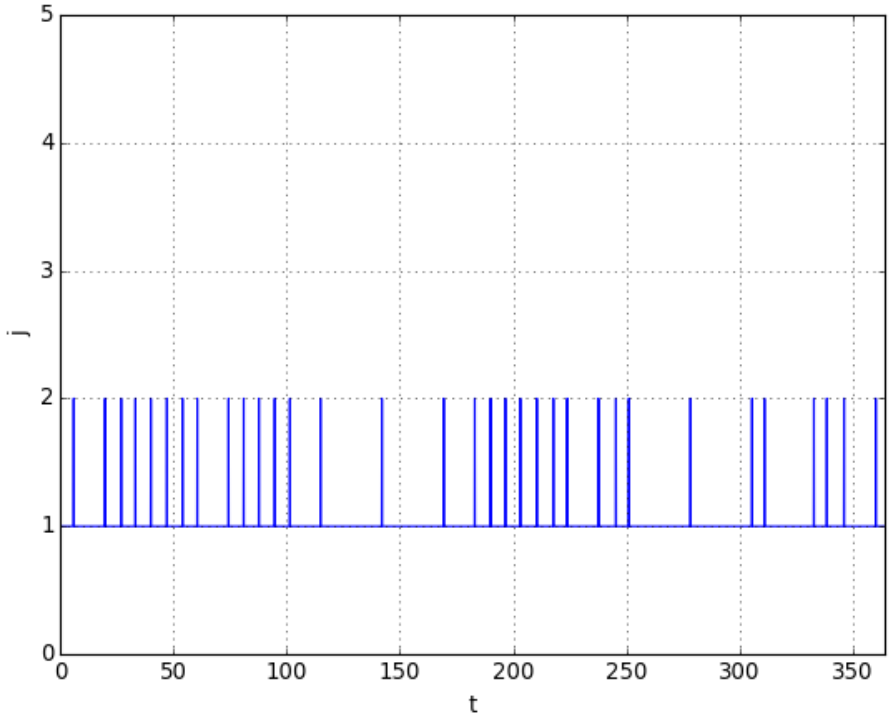
La distance Terre-Lune étant d'environ 2 millième de la distance Terre-Soleil, la Terre et la Lune sont pratiquement confondus à cette échelle. On vérifie que le calcul s'est bien déroulé en traçant l'estimation de l'erreur locale (qui doit rester inférieure à 1) :

```
figure(figsize=(8,5))
plot(t,e)
xlabel('t')
ylabel('e')
grid()
```



et le degré du polynôme utilisé pour l'extrapolation :

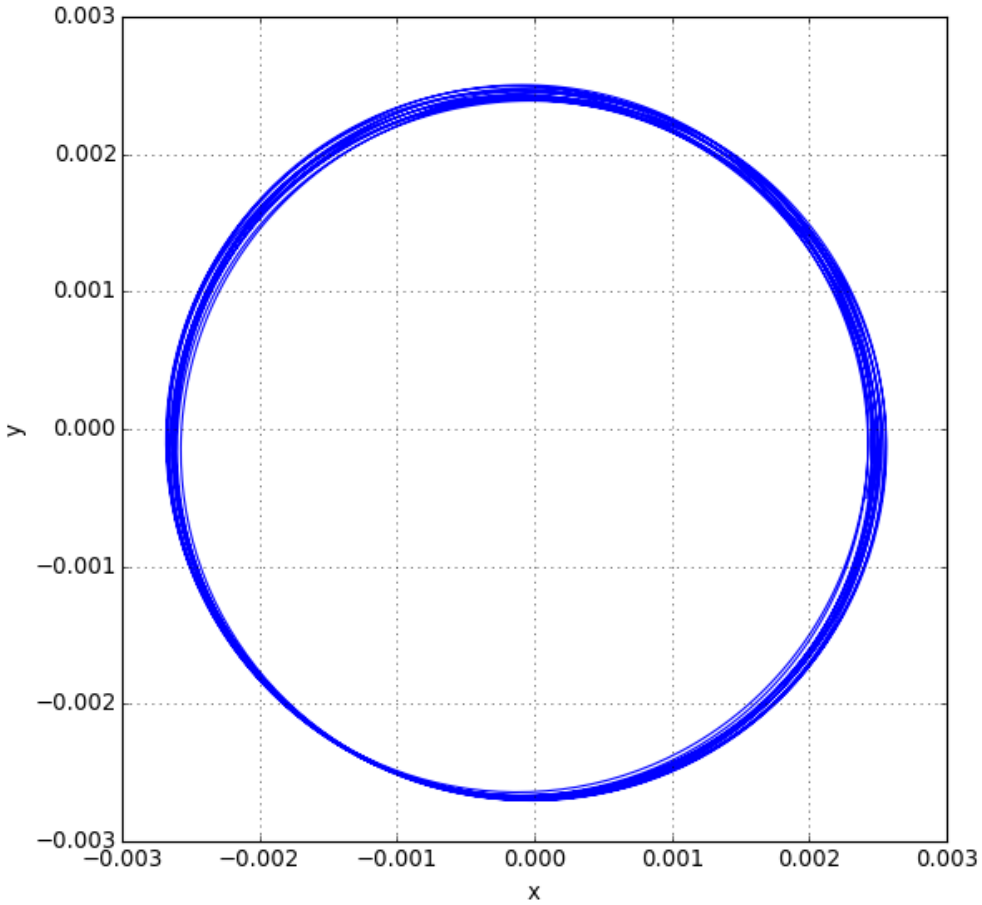
```
figure()
plot(t,J)
xlabel("t")
ylabel("j")
grid()
axis([0,t.max(),0,5])
```



Le degré utilisé est variable, afin de satisfaire la tolérance demandée. Bien que le pas H soit constant, le pas effectif h est réduit à chaque fois que le degré augmente. Cependant, c'est l'extrapolation polynomiale qui fait vraiment l'efficacité de la méthode de Bulirsch-Stoer.

Pour voir le mouvement de la Lune en projection dans le plan XY, on peut le tracer dans le référentiel géocentrique :

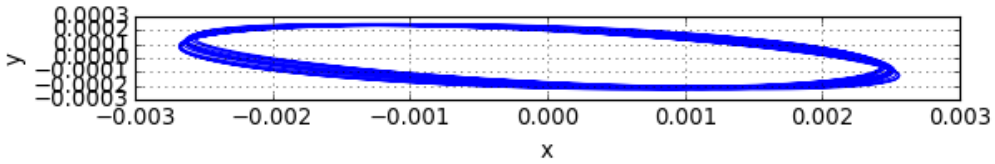
```
figure(figsize=(8,8))
plot(Y[6]-Y[0],Y[7]-Y[1],'b')
xlabel("x")
ylabel("y")
axes().set_aspect('equal')
grid()
```



voici aussi la projection dans le plan XZ :

```
figure(figsize=(8,8))
plot(Y[6]-Y[0],Y[8]-Y[2],'b')
xlabel("x")
```

```
ylabel("y")
axes().set_aspect('equal')
grid()
```



En voyant les changement d'orbite de la Lune d'une période à l'autre, on comprend pourquoi le problème du mouvement lunaire est si difficile.

6.c. Comparaison avec les éphémérides de l'IMCCE

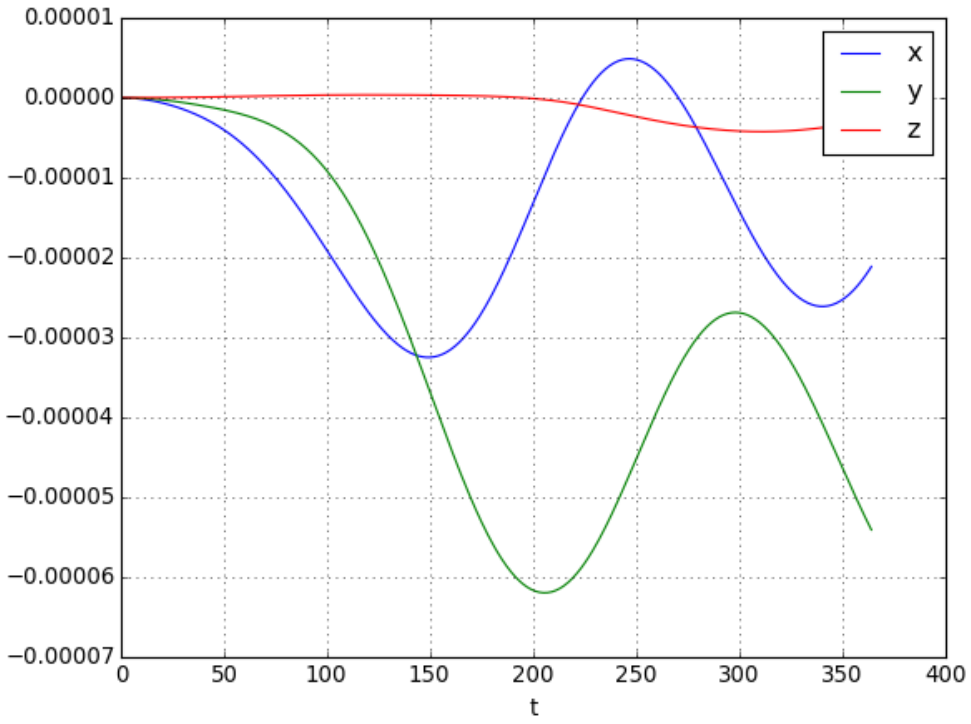
On fait tout d'abord une extraction des éphémérides pour la Terre sur une durée de 365 jours, avec un intervalle de 1 jour.

```
terre = numpy.loadtxt("miriade-earth.txt",skiprows=4,delimiter=",",usecols=(2,3,4,6,7
```

L'intégration numérique a été faite avec un intevalle de temps H=0.1 jours. On doit donc prélever un point sur 10 pour faire la comparaison avec les éphémérides :

```
xT = Y[0][::10]
yT = Y[1][::10]
```

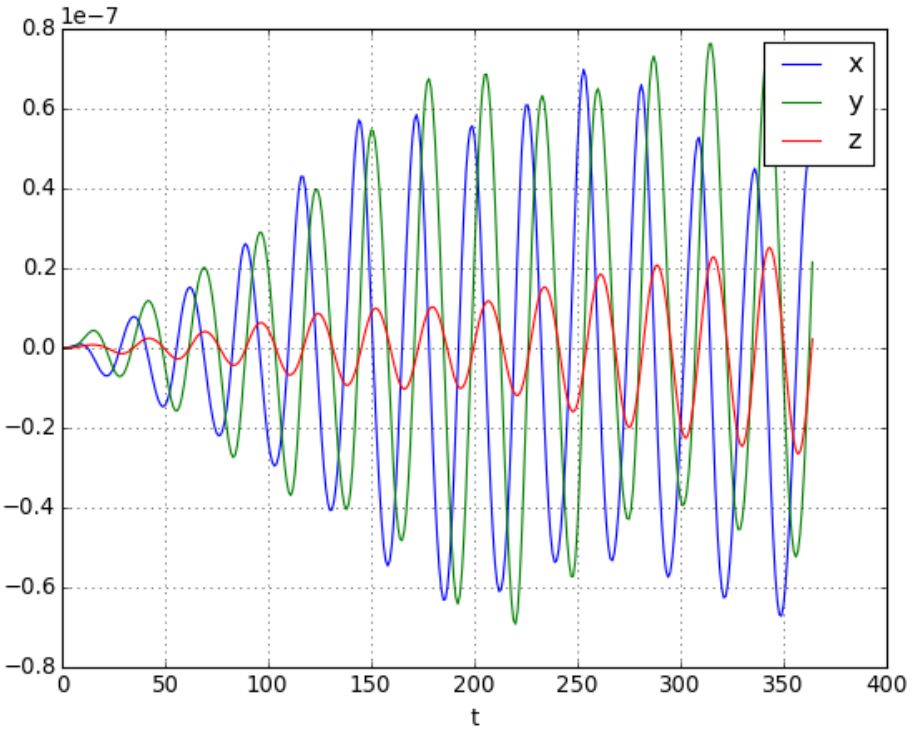
```
zT = Y[2][::10]
t2 = t[::10]

figure()
plot(t2,xT-terre[0],label="x")
plot(t2,yT-terre[1],label="y")
plot(t2,zT-terre[2],label="z")
xlabel("t")
grid()
legend(loc="upper right")
```



Le plus grand écart (sur la variable y) est de 0.00006 UA, soit environ 9000 km.

Voici la même comparaison pour la Lune, dans le référentiel géocentrique :

```
lune = numpy.loadtxt("miriade-moon.txt",skiprows=4,delimiter=",",usecols=(2,3,4,6,7,8
xL = Y[6][::10]
yL = Y[7][::10]
zL = Y[8][::10]
figure()
plot(t2,xL-xT-(lune[0]-terre[0]),label="x")
plot(t2,yL-yT-(lune[1]-terre[1]),label="y")
plot(t2,zL-zT-(lune[2]-terre[2]),label="z")
xlabel("t")
grid()
legend(loc="upper right")
```



Le plus grand écart est d'environ 12 km.

Les écarts absolus observés sur ces courbes ne sont pas causés par le calcul numérique lui-même, car changer la tolérance ne change pas ces courbes.

7. Système Soleil-Terre-Lune-Jupiter

L'écart entre notre calcul numérique et les éphémérides est peut-être en partie dû aux perturbations de Jupiter. Pour le vérifier, nous allons ajouter Jupiter au système :

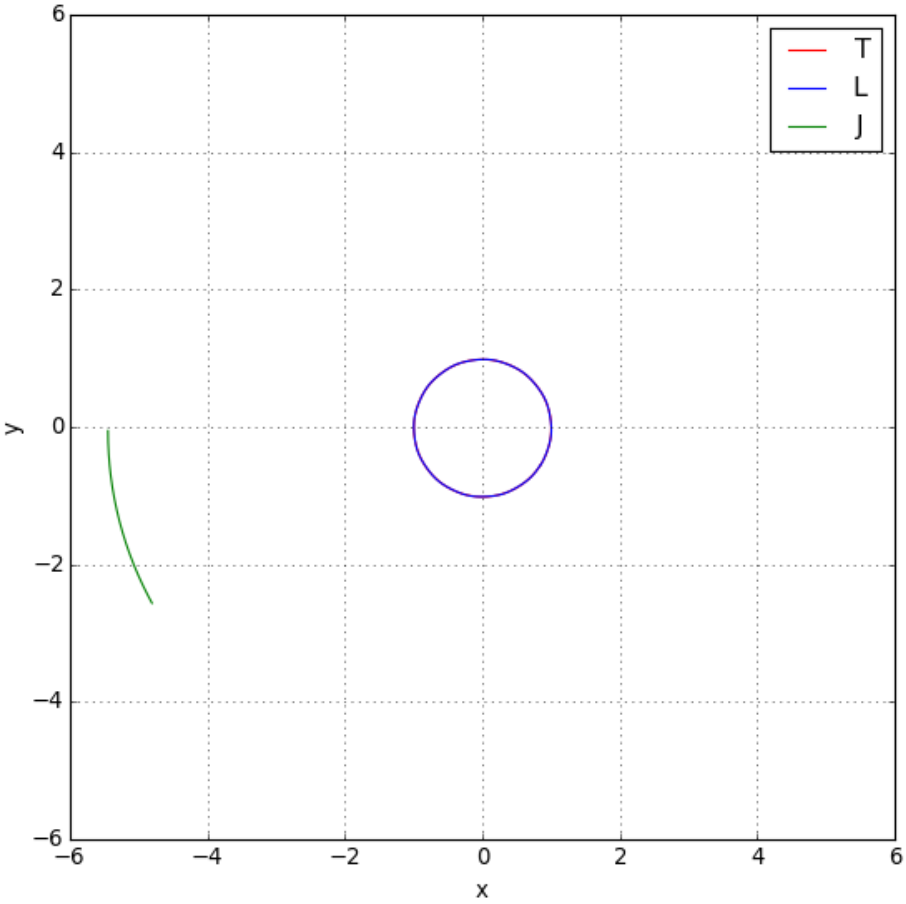
```
mJ = 1.0/1047.355
```

- xT,yT,zT : 0,1,2
- uT,vT,wT : 3,4,5
- xL,yL,zL : 6,7,8
- uL,vL,wL : 9,10,11
- xJ,yJ,zJ : 12,13,14
- uJ,vJ,zJ : 15,16,17

```
def systeme(Y,t):
    rT3 = numpy.power(Y[0]*Y[0]+Y[1]*Y[1]+Y[2]*Y[2],1.5)
    rL3 = numpy.power(Y[6]*Y[6]+Y[7]*Y[7]+Y[8]*Y[8],1.5)
    rJ3 = numpy.power(Y[12]*Y[12]+Y[13]*Y[13]+Y[14]*Y[14],1.5)
    xTL = Y[6]-Y[0]
    yTL = Y[7]-Y[1]
```

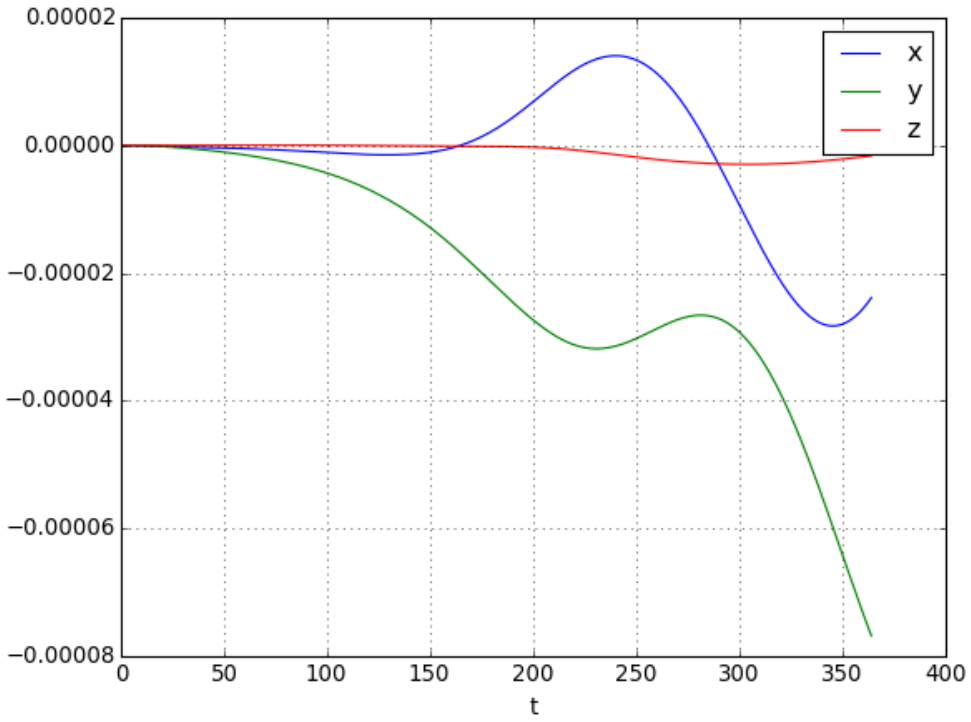
```
    zTL = Y[8]-Y[2]
    xTJ = Y[12]-Y[0]
    yTJ = Y[13]-Y[1]
    zTJ = Y[14]-Y[2]
    xJL = Y[6]-Y[12]
    yJL = Y[7]-Y[13]
    zJL = Y[8]-Y[14]
    rTL3 = numpy.power(xTL*xTL+yTL*yTL+zTL*zTL,1.5)
    rTJ3 = numpy.power(xTJ*xTJ+yTJ*yTJ+zTJ*zTJ,1.5)
    rJL3 = numpy.power(xJL*xJL+yJL*yJL+zJL*zJL,1.5)
    xrT = Y[0]/rT3
    yrT = Y[1]/rT3
    zrT = Y[2]/rT3
    xrL = Y[6]/rL3
    yrL = Y[7]/rL3
    zrL = Y[8]/rL3
    xrJ = Y[12]/rJ3
    yrJ = Y[13]/rJ3
    zrJ = Y[14]/rJ3
    aT = -k2*((1.0+mT)*xrT+mL*(-xTL/rTL3+xrL)+mJ*(-xTJ/rTJ3+xrJ))
    bT = -k2*((1.0+mT)*yrT+mL*(-yTL/rTL3+yrL)+mJ*(-yTJ/rTJ3+yrJ))
    cT = -k2*((1.0+mT)*zrT+mL*(-zTL/rTL3+zrL)+mJ*(-zTJ/rTJ3+zrJ))
    aL = -k2*((1.0+mL)*xrL+mT*(xTL/rTL3+xrT)+mJ*(xJL/rJL3+xrJ))
    bL = -k2*((1.0+mL)*yrL+mT*(yTL/rTL3+yrT)+mJ*(yJL/rJL3+yrJ))
    cL = -k2*((1.0+mL)*zrL+mT*(zTL/rTL3+zrT)+mJ*(zJL/rJL3+zrJ))
    aJ = -k2*((1.0+mJ)*xrJ+mT*(xTJ/rTJ3+xrT)+mL*(-xJL/rJL3+xrL))
    bJ = -k2*((1.0+mJ)*yrJ+mT*(yTJ/rTJ3+yrT)+mL*(-yJL/rJL3+yrL))
    cJ = -k2*((1.0+mJ)*zrJ+mT*(zTJ/rTJ3+zrT)+mL*(-zJL/rJL3+zrL))
    return numpy.array([Y[3],Y[4],Y[5],aT,bT,cT,Y[9],Y[10],Y[11],aL,bL,cL,Y[15],Y[16]


data = numpy.loadtxt("miriade-jupiter.txt",skiprows=4,delimiter=",",usecols=(2,3,4,6,
Y_jupiter = data[0] # condition initiale pour Jupiter au 20/8/2016  0h00
Yi = numpy.append(Yi,Y_jupiter)


T = 364.0
H = 0.1
(t,tab_y,e,J) = bulirsch(systeme,jmax,Yi,T,H,atol,rtol)
Y = tab_y.transpose()
figure(figsize=(8,8))
plot(Y[0],Y[1],'r',label="T")
plot(Y[6],Y[7],'b',label="L")
plot(Y[12],Y[13],'g',label="J")
xlabel("x")
ylabel("y")
axes().set_aspect('equal')
axis([-6,6,-6,6])
legend(loc="upper right")
```

```
grid()
```



Comparaison avec les éphémérides pour la Terre :

```
terre = numpy.loadtxt("miriade-earth.txt",skiprows=4,delimiter=",",usecols=(2,3,4,6,7
xT = Y[0][::10]
yT = Y[1][::10]
zT = Y[2][::10]
t2 = t[::10]

figure()
plot(t2,xT-terre[0],label="x")
plot(t2,yT-terre[1],label="y")
plot(t2,zT-terre[2],label="z")
xlabel("t")
grid()
legend(loc="upper right")
```



Il y a toujours un écart mais sa forme est différente, ce qui montre que l'influence de Jupiter sur la Terre au cours d'une année n'est pas négligeable à ce niveau de précision (moins de un dixmillième d'UA).

8. Système solaire à N planètes

Pour un système à plus de 3 planètes, il devient trop fastidieux de développer les équations une par une. Il faut écrire une boucle sur l'indice des planètes. On a tout d'abord besoin d'un tableau pour les masses. Dans l'exemple ci-dessous, on ajoute Mars, Vénus, Mercure et Saturne.

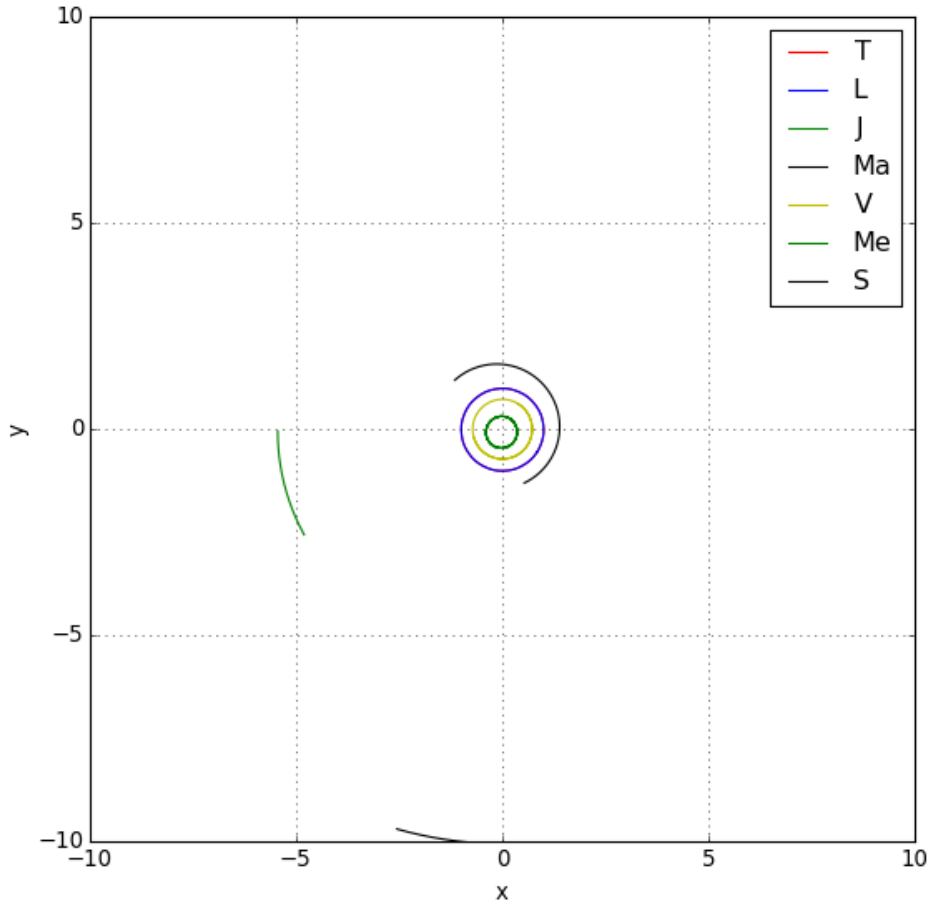
```
N = 7
mMa = 1.0/3098710.0
mV = 1.0/408523.5
mMe = 1.0/6023600.0
mS = 1.0/3498.5
m = [mT,mL,mJ,mMa,mV,mMe,mS]
data = numpy.loadtxt("miriade-mars.txt",skiprows=4,delimiter=",",usecols=(2,3,4,6,7,8
Y_mars = data[0] # condition initiale pour Mars au 20/8/2016  0h00
Yi = numpy.append(Yi,Y_mars)
data = numpy.loadtxt("miriade-venus.txt",skiprows=4,delimiter=",",usecols=(2,3,4,6,7,
Y_venus = data[0] # condition initiale pour Venus au 20/8/2016  0h00
Yi = numpy.append(Yi,Y_venus)
data = numpy.loadtxt("miriade-mercury.txt",skiprows=4,delimiter=",",usecols=(2,3,4,6,
Y_mercure = data[0] # condition initiale pour Mercure au 20/8/2016  0h00
Yi = numpy.append(Yi,Y_mercure)
```

```
data = numpy.loadtxt("miriade-saturn.txt",skiprows=4,delimiter=",",usecols=(2,3,4,6,7
Y_saturne = data[0] # condition initiale pour Saturne au 20/8/2016  0h00
Yi = numpy.append(Yi,Y_saturne)
```

Pour faciliter l'implémentation, on prévoit un tableau à deux indices pour stocker les distances (au cube) entre deux planètes. On prévoit aussi des tableaux pour stocker les rapports des coordonnées par la distance au Soleil au cube, et un tableau pour stocker les dérivées :

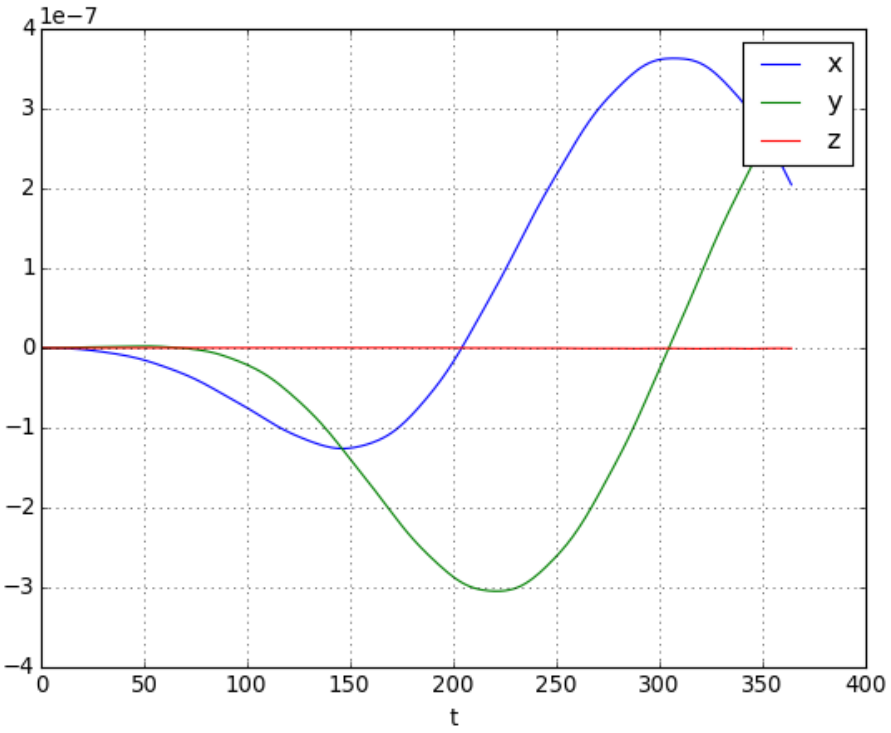
```
r3 = numpy.zeros((N,N))
xr = numpy.zeros(N)
yr = numpy.zeros(N)
zr = numpy.zeros(N)
Yp = numpy.zeros(N*6)


def systeme(Y,t):
    for i in range(N):
        k = i*6
        r = numpy.power(Y[k]*Y[k]+Y[k+1]*Y[k+1]+Y[k+2]*Y[k+2],1.5)
        xr[i] = Y[k]/r
        yr[i] = Y[k+1]/r
        zr[i] = Y[k+2]/r
        for j in range(i):
            l = j*6
            x = Y[l]-Y[k]
            y = Y[l+1]-Y[k+1]
            z = Y[l+2]-Y[k+2]
            r3[i][j] = r3[j][i] = numpy.power(x*x+y*y+z*z,1.5)
        for j in range(i+1,N):
            l = j*6
            x = Y[l]-Y[k]
            y = Y[l+1]-Y[k+1]
            z = Y[l+2]-Y[k+2]
            r3[i][j] = r3[j][i] = numpy.power(x*x+y*y+z*z,1.5)
        Yp[k] = Y[k+3]
        Yp[k+1] = Y[k+4]
        Yp[k+2] = Y[k+5]
    for i in range(N):
        k = i*6
        g = k2*(1.0+m[i])
        Yp[k+3] = -g*xr[i]
        Yp[k+4] = -g*yr[i]
        Yp[k+5] = -g*zr[i]
        for j in range(i):
            l = j*6
            g = k2*m[j]
```

```
            r = r3[i][j]
            Yp[k+3] += -g*((Y[k]-Y[l])/r+xr[j])
            Yp[k+4] += -g*((Y[k+1]-Y[l+1])/r+yr[j])
            Yp[k+5] += -g*((Y[k+2]-Y[l+2])/r+zr[j])
        for j in range(i+1,N):
            l = j*6
            g = k2*m[j]
            r = r3[i][j]
            Yp[k+3] += -g*((Y[k]-Y[l])/r+xr[j])
            Yp[k+4] += -g*((Y[k+1]-Y[l+1])/r+yr[j])
            Yp[k+5] += -g*((Y[k+2]-Y[l+2])/r+zr[j])
    return Yp


T = 364.0
H = 0.1
(t,tab_y,e,J) = bulirsch(systeme,jmax,Yi,T,H,atol,rtol)
Y = tab_y.transpose()
figure(figsize=(8,8))
plot(Y[0],Y[1],'r',label="T")
plot(Y[6],Y[7],'b',label="L")
plot(Y[12],Y[13],'g',label="J")
plot(Y[18],Y[19],'k',label="Ma")
plot(Y[24],Y[25],'y',label="V")
plot(Y[30],Y[31],'g',label="Me")
plot(Y[36],Y[37],'k',label="S")
xlabel("x")
ylabel("y")
axes().set_aspect('equal')
axis([-10,10,-10,10])
legend(loc="upper right")
grid()
```



Comparaison avec les éphémérides pour la Terre :

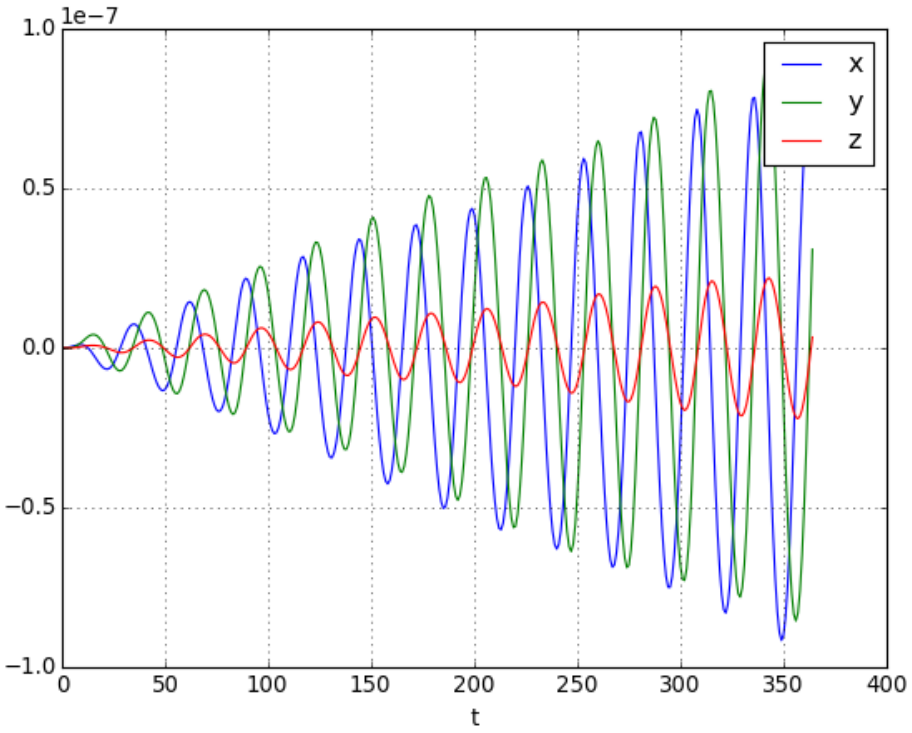
```
terre = numpy.loadtxt("miriade-earth.txt",skiprows=4,delimiter=",",usecols=(2,3,4,6,7
xT = Y[0][::10]
yT = Y[1][::10]
zT = Y[2][::10]
t2 = t[::10]

figure()
plot(t2,xT-terre[0],label="x")
plot(t2,yT-terre[1],label="y")
plot(t2,zT-terre[2],label="z")
xlabel("t")
grid()
legend(loc="upper right")
```



L'erreur de position sur la Terre ne dépasse pas 60 km (contre 9000 km pour le système Soleil-Terre-Lune). La prise en compte de Jupiter et de Saturne est nécessaire pour atteindre cette précision.

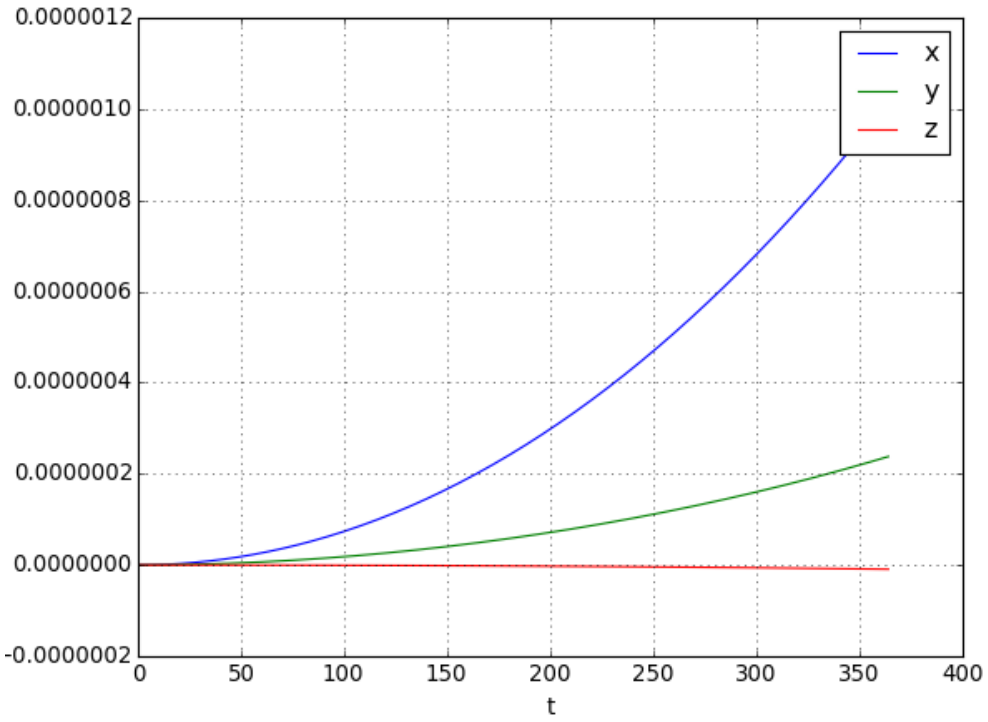
Voici la même comparaison pour la Lune, dans le référentiel géocentrique :

```
lune = numpy.loadtxt("miriade-moon.txt",skiprows=4,delimiter=",",usecols=(2,3,4,6,7,8
xL = Y[6][::10]
yL = Y[7][::10]
zL = Y[8][::10]
figure()
plot(t2,xL-xT-(lune[0]-terre[0]),label="x")
plot(t2,yL-yT-(lune[1]-terre[1]),label="y")
plot(t2,zL-zT-(lune[2]-terre[2]),label="z")
xlabel("t")
grid()
legend(loc="upper right")
```



et pour Jupiter :

```
jupiter = numpy.loadtxt("miriade-jupiter.txt",skiprows=4,delimiter=",",usecols=(2,3,4
xJ = Y[12][::10]
yJ = Y[13][::10]
zJ = Y[14][::10]
figure()
plot(t2,xJ-jupiter[0],label="x")
plot(t2,yJ-jupiter[1],label="y")
plot(t2,zJ-jupiter[2],label="z")
xlabel("t")
grid()
legend(loc="upper right")
```



L'écart sur la position de Jupiter ne dépasse pas 150 km sur une année, ce qui est infime comparé à sa distance au Soleil. Pour réduire cet écart, il faut probablement prendre en compte Uranus et Neptune. De plus, une meilleure précision sur la position de Jupiter aura un effet sur la Terre, car Jupiter a une influence notable sur la Terre. A ce niveau de précision, il y a l'influence directe (Jupiter influence la Terre) et indirecte, par exemple Saturne influence Jupiter dont la position modifie son influence sur la Terre.

Références

[1] Bureau des longitudes, *Introduction aux éphémérides astronomiques*, (EDP Sciences, 1998)