Trajectoire orbitale de la Lune et du satellite Capstone : étude numérique

Yuguang XIAO et Noé DESPORTES DE LA FOSSE

29 mars 2023

1 Introduction

Les missions spatiales sont devenues une priorité pour de nombreux pays et organisations. Parmi ces missions, l'exploration de la Lune est l'un des principaux objectifs. Cependant, cette exploration nécessite une compréhension précise des mouvements de la Lune et de ses satellites. Dans ce rapport, nous étudions le mouvement de CAPSTONE, un CubeSat qui sera le premier engin spatial à fonctionner sur une orbite de halo quasi rectiligne autour de la Lune. Cette orbite unique implique des mouvements complexes, qui peuvent être modélisés par un problème à N-corps en mécanique. Notre objectif est de construire un modèle physique pour simuler le mouvement du satellite et de la Lune, en utilisant des équations différentielles [3] pour décrire le système. Nous analyserons les données fournies par le site Web https ://ssd.jpl.nasa.gov/horizons/ [2], qui sont essentielles pour notre étude.

2 Matériel et Méthode

2.1 En problème à trois-corps [5, 4]

Considérons un système mécanique S formé de trois points matériels M_1, M_2 et M_3 de masse respective m_1, m_2 et m_3. Nous étudions la dynamique de ce système dans un référentiel R galiléen et l'on note $\vec{u_1} = \overrightarrow{GM_1}$, $\vec{u_2} = \overrightarrow{GM_2}$ et $\vec{u_3} = \overrightarrow{GM_3}$,les vecteurs positions avec le point G, le centre de masse de M_1, M_2 et M_3. Nous allons montrer que lorsque le système est isolé, le problème se découple en trois mouvements indépendants.

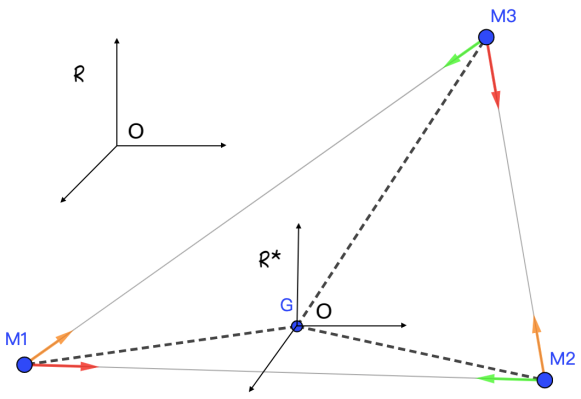


FIGURE 1 – Système à trois corps

Supposons donc que les trois corps soient en interaction mutuelle mais isolés de l'extérieur. Nous conservons la notation habituelle : $\overrightarrow{f_{12}}$ désigne la force qu'exerce le point M_1 sur M_2, $\overrightarrow{f_{32}}$ celle produite par M_3 sur M_2, $\overrightarrow{f_{13}}$ celle produite par M_1 sur M_3 et $\overrightarrow{f_{23}}$ celle produite par M_2 sur M_3. Par ailleurs, en vertu du théorème du centre d'inertie, nous avons :

$$(m_1 + m_2 + m_3)\frac{d\vec{v_G}}{dt} = \overrightarrow{F_{ext}} = \vec{0}$$

Ainsi, le centre d'inertie G décrit une trajectoire rectiligne uniforme. Le référentiel barycentrique R^* est donc en translation rectiligne uniforme par rapport à R ce qui lui confère un caractère galiléen. Analysons donc le mouvement dans le référentiel barycentrique R^*, par la formule gravitationnelle :

$$\overrightarrow{F_i} = G\sum_{k\neq i}\frac{m_k m_i \vec{r_{ik}}}{r_{ik}^3}$$

et par la deuxième loi de Newton :

$$\overrightarrow{F_i} = m_i\frac{d^2\vec{u_i}}{dt^2}$$

nous avons donc :

$$\overrightarrow{F_i} = m_i\frac{d^2\vec{u_i}}{dt^2} = G\sum_{k\neq i}\frac{m_k m_i \vec{r_{ik}}}{r_{ik}^3}$$

nous divisons les deux côtés par m_i :

$$\frac{d^2\vec{u_i}}{dt^2} = G\sum_{k\neq i}\frac{m_k \vec{r_{ik}}}{r_{ik}^3}$$

finalement, nous posons i =1, 2 et 3, et nous notons $\vec{r_3} = \overrightarrow{M_1M_3} = \vec{u_3} - \vec{u_1}, \vec{r_2} = \overrightarrow{M_1M_2} = \vec{u_2} - \vec{u_1}$, donc $\overrightarrow{M_2M_3} = \vec{r_3} - \vec{r_2} = \vec{u_3} - \vec{u_2}$ et $\overrightarrow{M_3M_2} = \vec{r_2} - \vec{r_3} = \vec{u_2} - \vec{u_3}$

$$\frac{d^2\vec{u_1}}{dt^2} = Gm_2\frac{\vec{r_2}}{|\vec{r_2}|^3} + Gm_3\frac{\vec{r_3}}{|\vec{r_3}|^3}$$

$$\frac{d^2\vec{u_2}}{dt^2} = Gm_1\frac{-\vec{r_2}}{|-\vec{r_2}|^3} + Gm_3\frac{\vec{r_3}-\vec{r_2}}{|\vec{r_3}-\vec{r_2}|^3}$$

$$\frac{d^2\vec{u_3}}{dt^2} = Gm_1\frac{-\vec{r_3}}{|-\vec{r_3}|^3} + Gm_2\frac{\vec{r_2}-\vec{r_3}}{|\vec{r_2}-\vec{r_3}|^3}$$

dans ce projet, nous devons étudier $\ddot{\vec{r_3}}$ et $\ddot{\vec{r_2}}$, car les conditions initiales sont données par rapport à la Terre (ici, M_1).

$$\ddot{\vec{r_3}} = \ddot{\vec{u_3}} - \ddot{\vec{u_1}} = Gm_1\frac{-\vec{r_3}}{|-\vec{r_3}|^3} + Gm_2\frac{\vec{r_2}-\vec{r_3}}{|\vec{r_2}-\vec{r_3}|^3} - Gm_2\frac{\vec{r_2}}{|\vec{r_2}|^3} - Gm_3\frac{\vec{r_3}}{|\vec{r_3}|^3}$$

nous avons donc

$$\ddot{\vec{r_3}} = -G(m_1+m_3)\frac{\vec{r_3}}{|\vec{r_3}|^3} + Gm_2\left(\frac{\vec{r_2}-\vec{r_3}}{|\vec{r_2}-\vec{r_3}|^3} - \frac{\vec{r_2}}{|\vec{r_2}|^3}\right) \quad (1)$$

de la même façon, nous pouvons obtenir :

OK

$$\ddot{\vec{r_2}} = -G(m_1+m_2)\frac{\vec{r_2}}{|\vec{r_2}|^3} + Gm_3\left(\frac{\vec{r_3}-\vec{r_2}}{|\vec{r_3}-\vec{r_2}|^3} - \frac{\vec{r_3}}{|\vec{r_3}|^3}\right) \quad (2)$$

Nous établissons un système de coordonnées cartésiennes (3D) dont l'origine est la Terre (Figure 2). Nous décomposons $\vec{r_3}$ en vecteurs dans trois directions, c'est-à-dire $\vec{r_3} = x_s\vec{e_x} + y_s\vec{e_y} + z_s\vec{e_z}$,

idem pour $\vec{r_2}$. De plus, nous pouvons décomposer (1) en trois équations différentielles (selon x, y et z), idem pour (2). À la fin, nous avons un système de six équations différentieles.

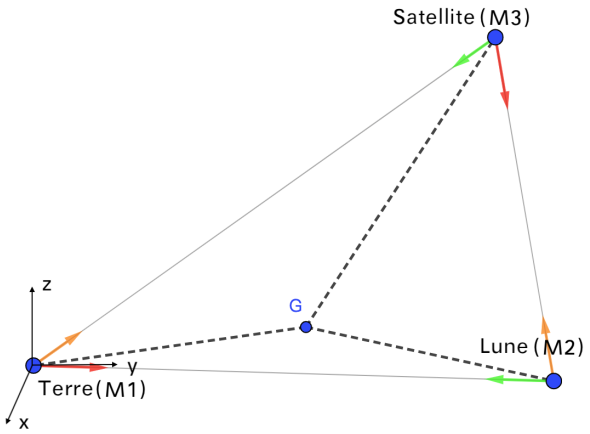


FIGURE 2 – Système de coordonnées cartésiennes établis

$$\begin{cases} \ddot{x_s} &= -G(m_t+m_s)\dfrac{x_s}{{r_s}^3} + Gm_l(\dfrac{x_l-x_s}{r_{ls}^3} - \dfrac{x_l}{r_l^3}) \\ \ddot{y_s} &= -G(m_t+m_s)\dfrac{y_s}{{r_s}^3} + Gm_l(\dfrac{x_l-x_s}{r_{ls}^3} - \dfrac{y_l}{r_l^3}) \\ \ddot{z_s} &= -G(m_t+m_s)\dfrac{z_s}{{r_s}^3} + Gm_l(\dfrac{z_l-z_s}{r_{ls}^3} - \dfrac{z_l}{r_l^3}) \\ \ddot{x_l} &= -G(m_t+m_l)\dfrac{x_l}{{r_l}^3} + Gm_s(\dfrac{x_s-x_l}{r_{sl}^3} - \dfrac{x_s}{r_s^3}) \\ \ddot{y_l} &= -G(m_t+m_l)\dfrac{y_l}{{r_l}^3} + Gm_s(\dfrac{x_s-x_l}{r_{sl}^3} - \dfrac{y_s}{r_s^3}) \\ \ddot{z_l} &= -G(m_t+m_l)\dfrac{z_l}{{r_l}^3} + Gm_s(\dfrac{z_s-z_l}{r_{sl}^3} - \dfrac{z_s}{r_s^3}) \end{cases} \tag{3}$$

avec m_t = la masse de la Terre, m_s = la masse du satellite, m_l = la masse de la Lune, $r_s = \sqrt{x_s^2+y_s^2+z_s^2}$, $r_l = \sqrt{x_l^2+y_l^2+z_l^2}$ et $r_{ls} = r_{sl} = \sqrt{(x_s-x_l)^2+(y_s-y_l)^2+(z_s-z_l)^2}$.

2.2 Méthode de Runge-Kutta (explicite) d'ordre 4 (méthode RK4) [1]

La méthode de Runge-Kutta explicite d'ordre 4 (méthode RK4) est une méthode numérique d'intégration d'équations différentielles. Elle permet de calculer une approximation numérique de la solution d'une équation différentielle ordinaire (EDO) du premier ordre de la forme $y' = f(t,y)$, où y est la fonction inconnue dépendant de la variable t. Dans ce projet, nous posons y = [xs,ys,zs,xs1,ys1,zs1,xl,yl,zl,xl1,yl1,zl1] où xs1,ys1,zs1 sont les vitesses du satellite selon x,y,z et de même pour la Lune avec xl1, yl1 et zl1. Nous posons donc dy[0] = y[3], dy[1] = y[4], dy[2] = y[5] et par la formule (3), nous avons donc dy[3] = -G * (mt + ms) * u[0] /rs**3 + G*ml*((u[6]-u[0])/(rls)**3 -(u[6]/rl**3)) et ainsi de suite pour les autres coordonnées. Voici le code où *deriv* représente la fonction f ci-dessus et u représente y.

```
1 def deriv(t,u):
2     '''
3         Soit u = (xs,ys,zs,xs1,ys1,zs1,xl,yl,zl,xl1,yl1,zl1),
```

```
        Équation d'évolution de l'oscillateur harmonique :  du/dt =
        d(xs,ys,zs,xs1,ys1,zs1)/dt = (xs1,ys1,zs1,-G * (mt+ms) * xs/|rs|^3 + G * ml
        ((xl-xs)/(|rl-rs|)^3 - xl/|rl|^3),...,...)

    '''
    du = np.empty(u.size)

    # Dérivée de la vitesse
    du[0] = u[3]
    ......
    du[3] = -G * (mt + ms) * u[0] /rs**3  + G*ml*((u[6]-u[0])/(rls)**3 -(u[6]/rl**3))
    ......
    du[6] = u[9]
    ......
    du[9] = -G * (mt + ml) * u[6] /rl**3  + G*ms*((u[0]-u[6])/(rsl)**3 -(u[0]/rs**3))
    ......
    return du
```

La méthode RK4 fonctionne en utilisant une combinaison de quatre approximations de la pente de la fonction $f(t, y)$ à différents points du domaine de la variable indépendante t. Les valeurs de la fonction y sont ensuite mises à jour à chaque itération en utilisant ces approximations de pente.

Plus précisément, la méthode RK4 calcule une approximation de la solution à l'instant t_{n+1} à partir de la solution à l'instant t_n en utilisant la formule suivante :

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

où
y_n est l'approximation de la solution à l'instant t_n
$k_1 = hf(t_n, y_n)$
$k_2 = hf(t_n + \frac{h}{2}, y_n + \frac{k_1}{2})$
$k_3 = hf(t_n + \frac{h}{2}, y_n + \frac{k_2}{2})$
$k_4 = hf(t_n + h, y_n + k_3)$
h est le pas d'intégration, c'est-à-dire la distance entre deux instants de temps consécutifs.

La méthode RK4 est une méthode d'ordre 4, ce qui signifie que l'erreur commise à chaque itération est proportionnelle à h^5. Elle est donc plus précise que les méthodes d'ordre inférieur comme la méthode d'Euler explicite.

Nous avons utilisé la méthode RK4 pour résoudre les équations différentielles du mouvement des corps célestes dans notre étude. Nous avons implémenté cette méthode dans notre programme Python en utilisant les données de position et de vitesse des corps célestes que nous avons obtenues à partir des fichiers CSV téléchargés à partir du site Web. (Voir le code en annexe)

Cependant, cela ne permet de calculer qu'une seule valeur à un moment donné. Par conséquent, nous avons besoin d'une boucle pour répéter le calcul n fois, jusqu'à ce que t_initial + dt*n ≈ t_final où dt correspond au pas de temps choisi.

OK

```python
def integrationEDO(start,end,step, v_ini, deriv):
    n = v_ini.size
    # nombre d'equa-diff du 1er ordre

    # Création du tableau temps
    interval = end - start                      # Intervalle
    num_points = int(interval / step) + 1       # Nombre d'éléments
    t = np.linspace(start, end, num_points)     # Tableau temps t

     # Initialisation du tableau v
    v = np.empty((n, num_points))
    # tableau y qui contient les valeurs actuelles de v au temps t[i]
    y = np.empty(n)

     # Condition initiale
    v[:, 0] = v_ini
    y[:] = v_ini # important de mettre [:] pour y,
                 # sinon y serait juste un alias de v_ini et non une copie
                 # ce qui ferait qu'on modifie v_ini quand on modifie y et
                 # donc aussi le u_ini donné à la fonction lors de son appel.

    # Boucle for
    for i in range(num_points - 1):
        y = rk4(t[i], step, y, deriv) # modifie le tableau y
        v[:, i + 1] = y

    # Argument de sortie
    return t, v
```

Finalement, il suffit de définir une fonction *resultat(tmax,dt)* en prenant comme argument initial u_ini = [xs,ys,zs,vxs,vys,vzs,xl,tl,zl,vxl,vyl,vzl], et en l'utilisant dans la fonction précédente pour obtenir notre résultat final u.

```python
def resultat(tmax, dt):
    # Conditions initiales
    u_ini = np.array([xs,ys,zs,vxs,vys,vzs,xl,yl,zl,vxl,vyl,vzl])

    # Méthode d'Euler
    t, u = integrationEDO(0, tmax, dt, u_ini, deriv)
    return u
```

3 Résultats

3.1 La trajectoire de mouvement réelle du satellite et de la lune

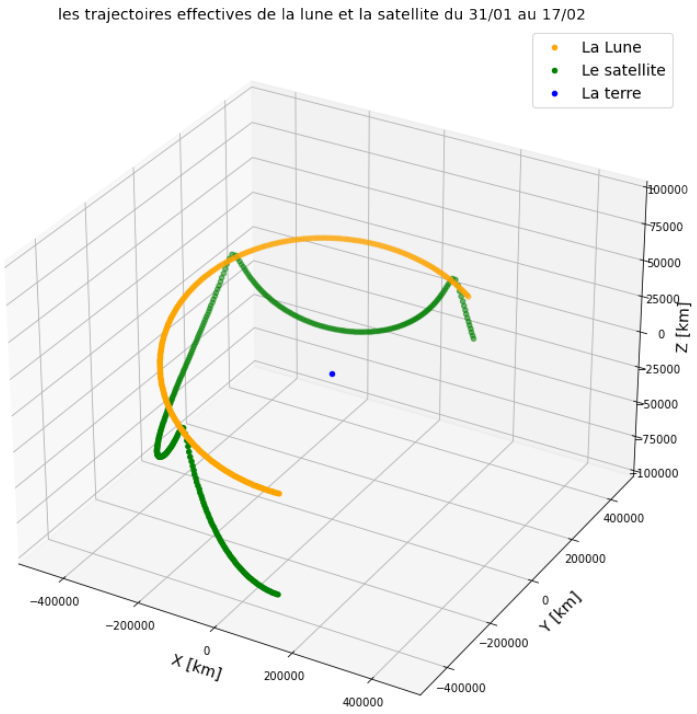


FIGURE 3 – Les trajectoires réelles de la lune et du satellite

Dans un premier temps, nous avons tracé la trajectoire réelle du satellite et de la Lune dans le referentiel terrestre à partir de données collectées sur le site Horizons. Voir le résultat ci-dessus. Selon la méthode de la section précédente, nous devons définir un pas de temps (dt) et utiliser le modèle réel pour ajuster la trajectoire calculée du mouvement de la lune et du satellite. Nous commençons par régler le pas sur 0,0001 jour, soit environ 9 secondes.

3.2 Cas : dt = 0,0001 jours (environ 9 secondes)

3.2.1 la trajectoire du mouvement réel et simulé du satellite et de la Lune

En analysant la FIGURE 4, nous avons remarqué que l'ajustement était initialement très bon, mais que les erreurs s'accumulent à partir d'un certain moment. Pour faciliter l'observation, nous avons sélectionné la coordonnée x du satellite et l'avons comparée à la valeur réelle.

3.2.2 Les trajectoires réelles et simulées du satellite selon l'axe x

En regardant la FIGURE 5, nous avons remarqué qu'il y a des incertitudes à partir du 7^e jour environ, mais qu'elles sont acceptables car il y a beaucoups de facteurs non considerés qui influent sur ce système. Par exemple : la force de gravité exercée par le Soleil. De plus, la valeur du pas de temps n'est pas optimale. Pour la suite, nous avons donc choisi un nouveau dt en cherchant pour quelle valeur de pas l'écart-type de l'énergie mécanique du système est minimale.

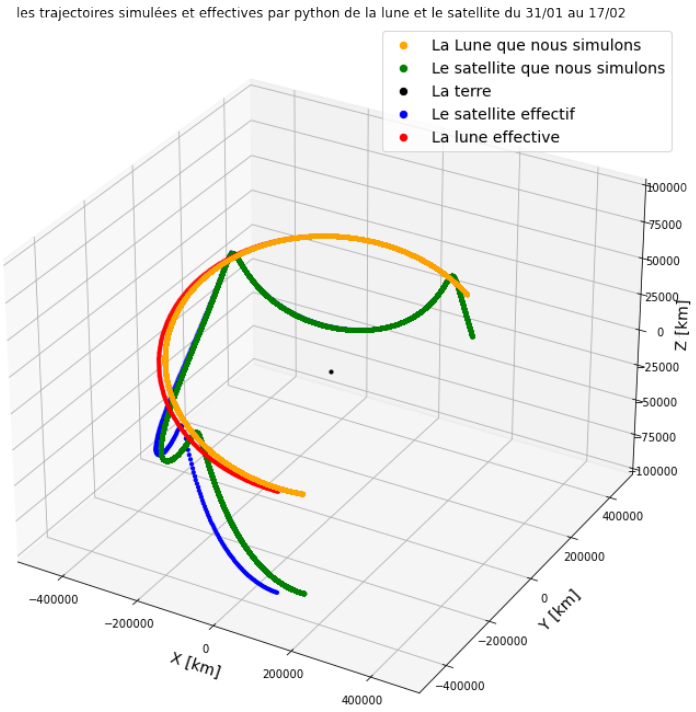


FIGURE 4 – Les trajectoires réelles et simulées de la Lune et du satellite

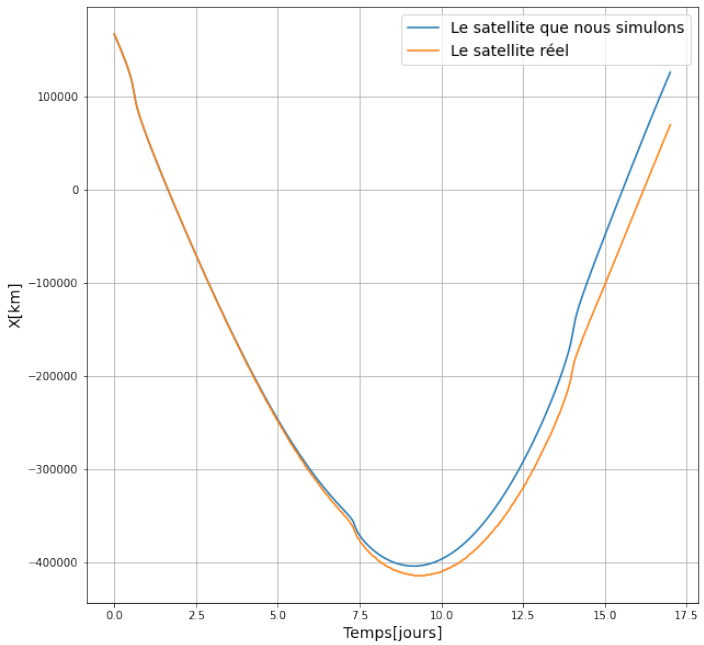


FIGURE 5 – Les trajectoires réelles et simulées de la Lune et du satellite

3.3 L'écart-type de la liste (la somme de l'énergie du satellite et de la Lune) par les différents *dt*

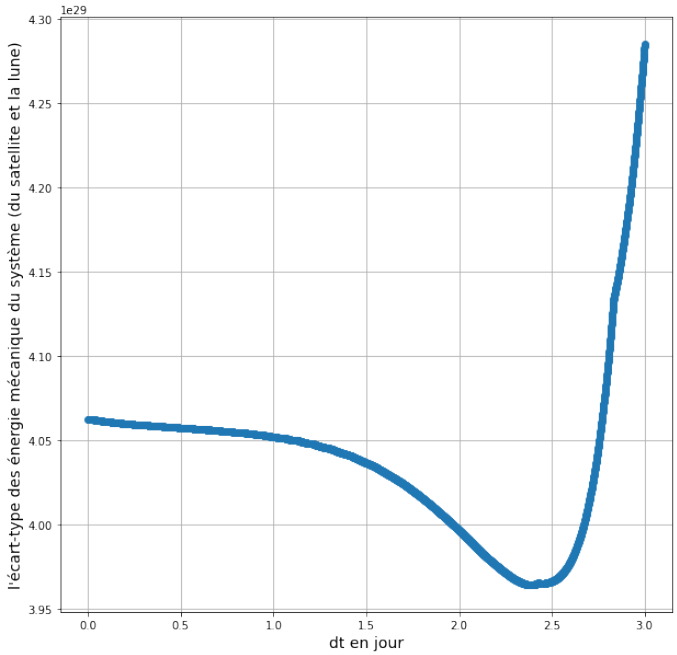


FIGURE 6 – L'écart-type (Lune et satellite) par les différents *dt*

En regardant la FIGURE 6, le pas de temps idéal est de 2,387 jours. Nous avons donc répeté les étapes du cas précedent avec ce nouveau pas de temps pour obtenir une meilleure simulation.

3.4 Cas : $dt = 2{,}387$ jours

3.4.1 la trajectoire de mouvement simulée du satellite et de la lune

Ok explication ensuite

En regardant la FIGURE 7, nous avons remarqué que malgré la modification de la valeur de *dt* pour minimiser l'écart-type global, le résultat n'était pas parfait. Pour vérifier davantage nos résultats, nous avons encore une fois extrait les valeurs de la coordonnée x du satellite et les avons comparées aux valeurs réelles.

3.4.2 Les trajectoires réelle et simulée du satellite selon l'axe x

En regardant la FIGURE 8, nous avons constaté qu'à partir du premier jour, il y avait une déviation très importante.

3.5 L'écart-type de l'énergie mécanique du satellite par rapport aux différents pas de temps

Cette fois-ci, nous avons calculé l'écart-type de l'énergie mécanique du satellite uniquement en fonction de *dt*.

En regardant la FIGURE 9, le pas de temps idéal obtenu par python est de 0,0072 jour. Nous avons répété les étapes des cas précedents avec ce nouveau pas de temps.

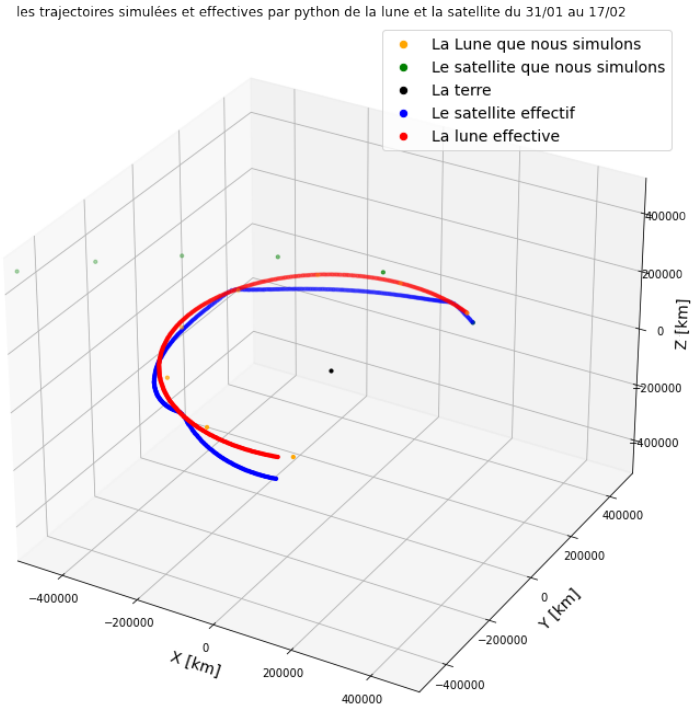


FIGURE 7 – les trajectoires du mouvement réel et simulé du satellite et de la Lune, si dt = 2,387 jours

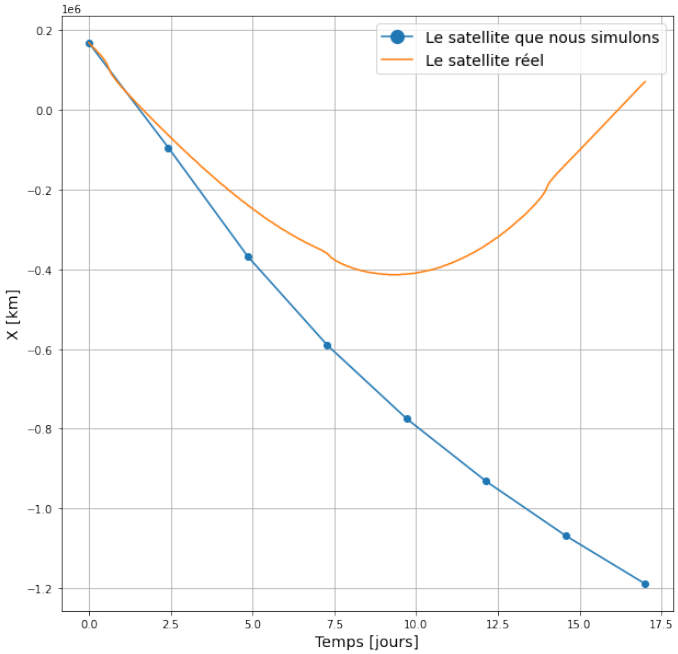


FIGURE 8 – la trajectoire de mouvement réelle et simulée du satellite et de la Lune selon x, si dt = 2,387 jours

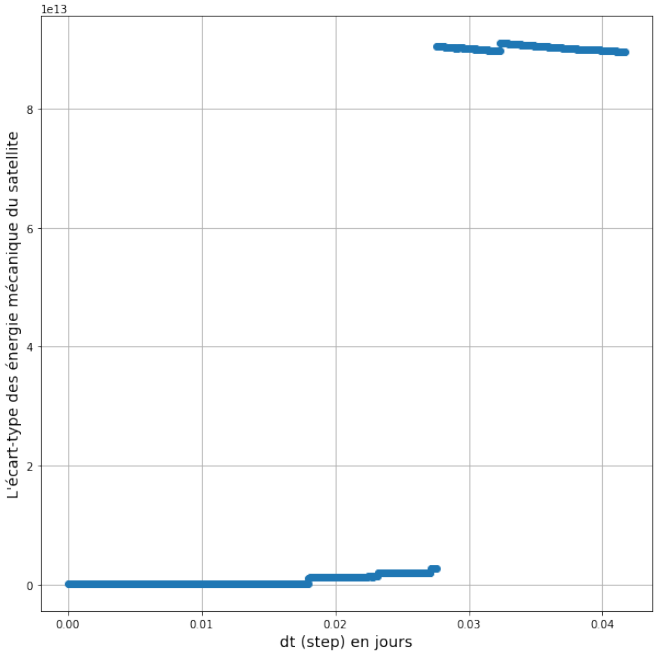


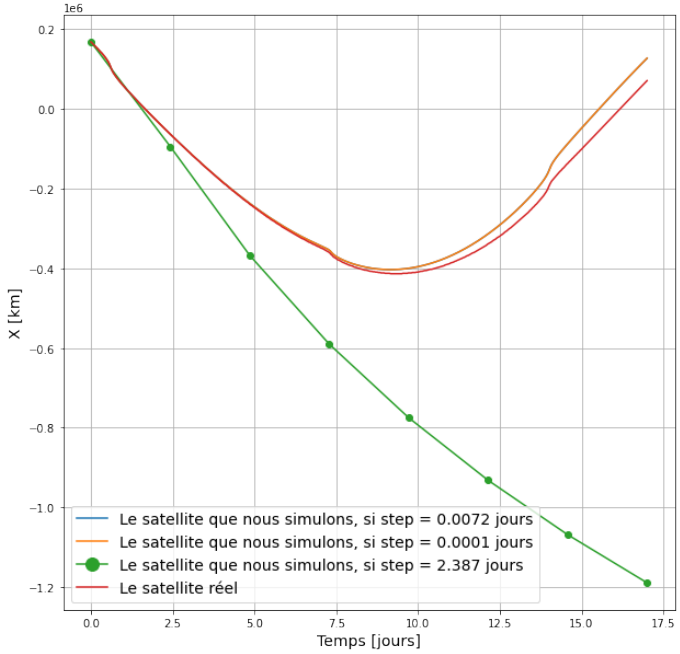
FIGURE 9 – L'écart-type (satellite) par les différents dt

3.6 Cas : dt = 0,0072 jours (environ 10 mins)

3.6.1 la trajectoire de mouvement simulée du satellite et de la lune

En regardant la FIGURE 10, cela semble similaire au *cas : dt = 0,0001 jours*. Pour une vérification supplémentaire, nous allons encore tracer le graphique selon x pour comparer.

3.6.2 Les trajectoires réelles et simulées du satellite selon l'axe x, si dt = 0,001 ; 0,0072 et 2,387 en jours



Indiquez où est la courbe bleue

FIGURE 11 – la trajectoire du mouvement réel et simulé du satellite et de la Lune selon x, si dt = 0,001 ; 0,0072 et 2,387 en jours

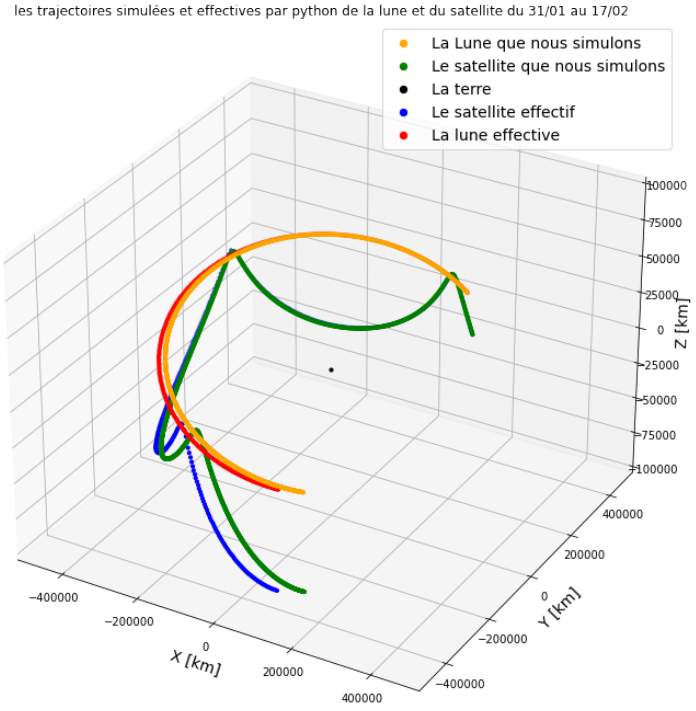


FIGURE 10 – la trajectoire du mouvement réel et simulé du satellite et de la Lune, si dt = 0,0072 jours

La courbe bleue et la courbe orange se superposent presque complètement. En regardant la FIGURE 9, nous pouvons également constater que pour les valeurs de step = 0,0001 jours et 0,0072 jours, l'écart-type global est très proche de zéro.

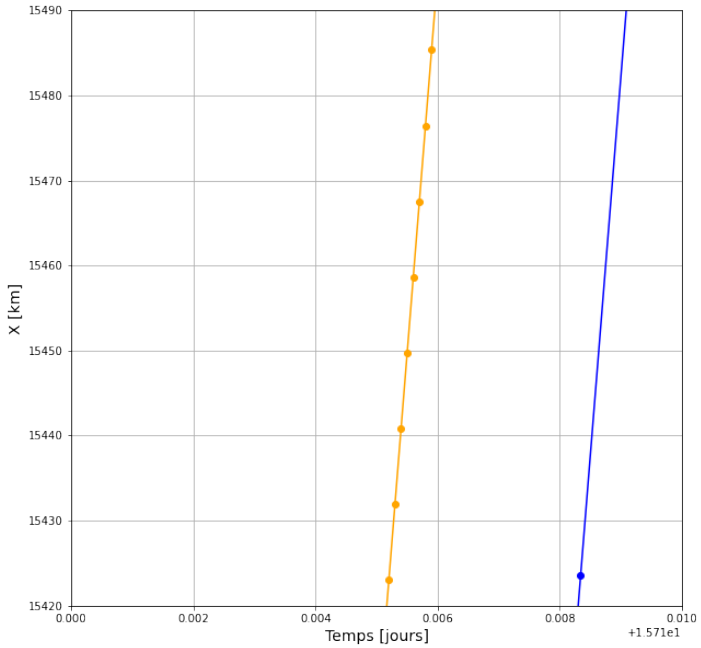


FIGURE 12 – la trajectoire du mouvement réel et simulé du satellite et de la Lune selon x, si dt = 0,001 ; 0,0072(Version agrandie)

En observant la FIGURE 12, nous remarquons que les points bleus sont en dessous des points oranges au même moment, ce qui signifie que la courbe bleue est plus proche de la situation

réelle car en observant la FIGURE 11, les courbes simulées sont supérieures aux courbes réelles.

4 Analyse des résultats et discussion

4.1 Analyse des résultats

Calculer la période orbitale du satellite pour connaître les périodes caractéristiques du problème et choisir un pas de temps.

Au début, nous avons défini dt = 0,0001 jour, sans savoir s'il s'agissait de la meilleure taille de pas. Nous l'avons simplement choisie aussi petite que possible en pensant que cela minimiserait les erreurs. En observant la FIGURE 5, nous avons remarqué une erreur significative qui a commencé à s'amplifier vers le septième jour. Afin d'ajuster le mieux possible les résultats, nous avons modifié la taille du pas de temps afin de minimiser l'écart-type de l'énergie mécanique du système car étant pseudo-isolé dans l'espace, cette énergie est supposée constante. Alors, la valeur du pas de temps qui minimise l'erreur sur l'énergie mécanique du système doit être celle qui nous rapproche le plus du cas réel. Nous avons utilisé la formule suivante pour ce calcul :

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(E_i - E_0)^2} \tag{4}$$

où E_0 est l'énergie mécanique initiale du système.

Nous avons donc exploité la valeur obtenue mais en examinant le graphique de la trajectoire simulée par l'équation différentielle (FIGURE 7), nous avons constaté que l'ajustement était très mauvais et que la trajectoire simulée du satellite différait considérablement de la valeur réelle. Comme la masse du satellite (25kg) que nous avons considéré est très faible par rapport à celle de la lune (7,36e22 kg) ; lorsque la valeur de l'écart-type est minimale pour l'ensemble (Lune et satellite), cela signifie que l'énergie mécanique de la lune est plus proche de la réalité, car l'impact mécanique du satellite est négligeable par rapport à celui de la lune. Par exemple, pour un objet tombant librement sur la Terre, en négligeant bien sûr la résistance de l'air, nous considérons que son énergie mécanique est conservée. Bien qu'il subisse l'attraction gravitationnelle de la Lune et que la lune subisse également l'attraction gravitationnelle de cet objet, nous ne considérons pas que l'énergie mécanique du système Objet-Lune est conservée mais que l'énergie mécanique de l'objet est conservée. Ainsi, dans le code suivant, nous ne tiendrons pas compte de l'énergie mécanique de la Lune, afin de mieux ajuster la trajectoire de mouvement aux conditions réelles. B

Finalement, nous avons amélioré la méthode en ne considérant que l'énergie mécanique du satellite, et avons obtenu un pas parfait (0,0072 jours) avec lequel nous avons constaté que la trajectoire simulée correspond étroitement à celle obtenue avec dt = 0,0001 jours. Néanmoins, est-ce que ce pas de temps est vraiment parfait ? Comme l'erreur est cumulative lors de l'utilisation de la méthode RK4, nous avons observé dans la FIGURE 11 que nous n'avons besoin que de calculer la valeur finale de x (Il suffit de zoomer sur la fin des courbes pour voir quelle courbe est la plus proche de la courbe réelle.) pour voir si elle est plus proche de la vraie valeur. En observant la FIGURE 12, nous remarquons que les points bleus sont en dessous des points oranges au même moment, ce qui signifie que la courbe bleue est plus proche de la situation réelle.

et au bout de 7 jours, que se passe t'il ?

4.2 Discussion

Les résultats obtenus confirment l'hypothèse de départ selon laquelle la simulation peut reproduire avec une précision satisfaisante la trajectoire de la lune et du satellite. Cependant, les résultats d'observation montrent une erreur à partir d'un certain moment dans les simulations avec deux pas de temps différents.

L'analyse de l'écart type montre que les valeurs pour les pas de temps de 0,0001 jour et 0,0072 jour sont stables, avec un écart type proche de zéro. Cependant, entre ces deux pas de temps, le pas de 0,0072 jour est plus proche de la valeur réelle. D'autre part, des résultats instables ont été observés pour les pas de temps différents, ce qui confirme l'importance de choisir un pas de temps approprié pour la simulation. Par exemple, une méthode de contrôle de pas adaptatif peut être utilisée pour ajuster automatiquement le pas de chaque étape en fonction de l'erreur, afin de maintenir l'erreur dans des limites acceptables.

Enfin, il convient de souligner que les résultats de simulation dépendent des paramètres sélectionnés pour le modèle. D'autres tests peuvent être effectués pour améliorer la précision des résultats, tels que le choix de méthodes numériques plus optimisées pour résoudre les équations différentielles ou la prise en compte de l'influence d'autres forces gravitationnelles, telles que le soleil.

En résumé, ces résultats montrent que la modélisation numérique de la trajectoire de la Lune et du satellite peut être effectuée avec précision en choisissant un pas de temps approprié et en prenant en compte les limites du modèle sélectionné.

5 Conclusion

Notre étude visait à simuler la trajectoire de mouvement du satellite et de la Lune du 31 janvier au 17 février 2023 en utilisant un modèle physique et des données réelles. Nous avons déterminé le meilleur intervalle de temps de 0,0072 jours pour une simulation précise mais l'erreur augmente avec le temps. Nous recommandons donc une méthode de contrôle adaptative de l'intervalle de temps. Nous pouvons également tester et optimiser d'autres paramètres (Considérons l'attraction gravitationnelle du Soleil) pour améliorer la précision des résultats. En résumé, notre étude a démontré la précision de notre modèle et proposé des améliorations pour une simulation plus précise.

6 Annexes

6.1 Lecture des données

Nous avons téléchargé les données de position (x : km, y : km et z :km) et de vitesse (km par jours) toutes les heures (km par jours) de la Lune et du satellite à partir de 0 :00 le 31 janvier jusqu'à 0 :00 le 17 février sur le site Web du «Horizons System» et les avons enregistrées dans deux fichiers en format csv noté «Les données du satellite.csv» et «Les données de la lune.csv».

```
import pandas as pd  # Exportation du fichier au format CSV de traitement de pandas
data_lune= pd.read_csv('Les données de la lune.csv',comment = '#', encoding = 'utf-8')
#Read Les donnees de la lune.csv
x_lune = data_lune["X"] # en km
y_lune = data_lune["Y"] # en km
z_lune = data_lune["Z"]  # en km
vx_lune = data_lune['VX']  # en km/jours
vy_lune = data_lune['VY']  # en km/jours
vz_lune = data_lune['VZ']  # en km/jours
# conditions initiales
```

```
xl = x_lune[0] # le premier élement, c'est-à-dire si t = 0, la position de la lune selon x
yl = y_lune[0] # ..... si t = 0, la postion de la lune selon y
zl = z_lune[0] # ..... si t = 0, la postion de la lune selon z
vxl = vx_lune[0] # ..... si t = 0, la vitesse de la lune selon x
vyl = vy_lune[0] # ..... si t = 0, la vitesse de la lune selon y
vzl = vz_lune[0] # ..... si t = 0, la vitesse de la lune selon z
data_satellite = pd.read_csv('Les données du satellite.csv',comment = '#' , encoding = "utf-8")
#Read Les données de la satellite.csv
x_satellite = data_satellite['X'] # en km
y_satellite = data_satellite['Y'] # en km
z_satellite = data_satellite['Z'] # en km
vx_satellite = data_satellite['VX'] # en km/jours
vy_satellite = data_satellite['VY'] # en km/jours
vz_satellite = data_satellite['VZ'] # en km/jours
# conditions initiales
xs = x_satellite[0] # le premier élement, c'est-à-dire si t = 0, la position du satellite selon x
ys = y_satellite[0] # ..... si t = 0, la postion du satellite selon y
zs = z_satellite[0] # ..... si t = 0, la postion du satellite selon z
vxs = vx_satellite[0] # ..... si t = 0, la vitesse du satellite selon x
vys = vy_satellite[0] # ..... si t = 0, la vitesse du satellite selon y
vzs = vz_satellite[0] # ..... si t = 0, la vitesse du satellite selon z
# L'unité de base du SI
"""
import pandas as pd  # Exportation du fichier au format CSV de traitement de pandas
data_lune= pd.read_csv('Les données de la lune.csv',comment = '#', encoding = 'utf-8')
#Read Les donnees de la lune.csv
x_lune = data_lune["X"] * 1000 # en mètre
y_lune = data_lune["Y"] * 1000 # en mètre
z_lune = data_lune["Z"] * 1000 # en mètre
vx_lune = data_lune['VX'] *1000 /(24*3600) # en m/s
vy_lune = data_lune['VY'] *1000 /(24*3600) # en m/s
vz_lune = data_lune['VZ'] *1000 /(24*3600) # en m/s
# conditions initiales
xl = x_lune[0] # le premier élement, c'est-à-dire si t = 0, la position de la lune selon x
yl = y_lune[0] # ..... si t = 0, la postion de la lune selon y
zl = z_lune[0] # ..... si t = 0, la postion de la lune selon z
vxl = vx_lune[0] # ..... si t = 0, la vitesse de la lune selon x
vyl = vy_lune[0] # ..... si t = 0, la vitesse de la lune selon y
vzl = vz_lune[0] # ..... si t = 0, la vitesse de la lune selon z
data_satellite = pd.read_csv('Les données du satellite.csv',comment = '#' , encoding = "utf-8")
```

```
#Read Les données de la satellite.csv
x_satellite = data_satellite['X'] *1000 # en mètre
y_satellite = data_satellite['Y'] *1000 # en mètre
z_satellite = data_satellite['Z'] *1000 # en mètre
vx_satellite = data_satellite['VX'] *1000 /(24*3600) # en m/s
vy_satellite = data_satellite['VY'] *1000 /(24*3600) # en m/s
vz_satellite = data_satellite['VZ'] *1000 /(24*3600) # en m/s
# conditions initiales
xs = x_satellite[0] # le premier élement, c'est-à-dire si t = 0, la position du satellite selon x
ys = y_satellite[0] # ..... si t = 0, la postion du satellite selon y
zs = z_satellite[0] # ..... si t = 0, la postion du satellite selon z
vxs = vx_satellite[0] # ..... si t = 0, la vitesse du satellite selon x
vys = vy_satellite[0] # ..... si t = 0, la vitesse du satellite selon y
vzs = vz_satellite[0] # ..... si t = 0, la vitesse du satellite selon z
"""
```

Les coordonnées cartésiennes :

Plan X-Y : plan orbital terrestre adopté à l'époque de référence. Note : Obliquité UAI76 de 84381.448 secondes d'arc par rapport au plan X-Y ICRF.

Axe X : ICRF

Axe Z : perpendiculaire au plan X-Y dans le sens directionnel (+ ou -) du pôle nord de la Terre à l'époque de référence.

6.2 Les trajectoires réelles

Ensuite, nous sommes prêts à dessiner les trajectoires réelles de la Lune, du satellite et de la Terre (O).

```
from mpl_toolkits import mplot3d
import numpy as np
import matplotlib.pyplot as plt

fig = plt.figure(figsize = (12, 12)) #la taille de la figure
ax = plt.axes(xlim=(-5e5,5e5),ylim=(-5e5,5e5),zlim = (-1e5,1e5),projection ="3d")
ax.scatter3D(x_lune, y_lune, z_lune, color = "orange",label = "La Lune")
ax.scatter3D(x_satellite, y_satellite, z_satellite, color = "green",label = "Le satellite")
ax.scatter3D([0],[0],[0],color = "blue", label = "La terre")
ax.set_xlabel("X [km]",fontsize = 14)
ax.set_ylabel("Y [km]",fontsize = 14)
ax.set_zlabel("Z [km]",fontsize = 14)
plt.legend(fontsize=14, markerscale=1., scatterpoints=1)
plt.title("les trajectoires effectives de la lune et la satellite du 31/01 au 17/02",fontsize = 14)
```

```
15 plt.show()
```

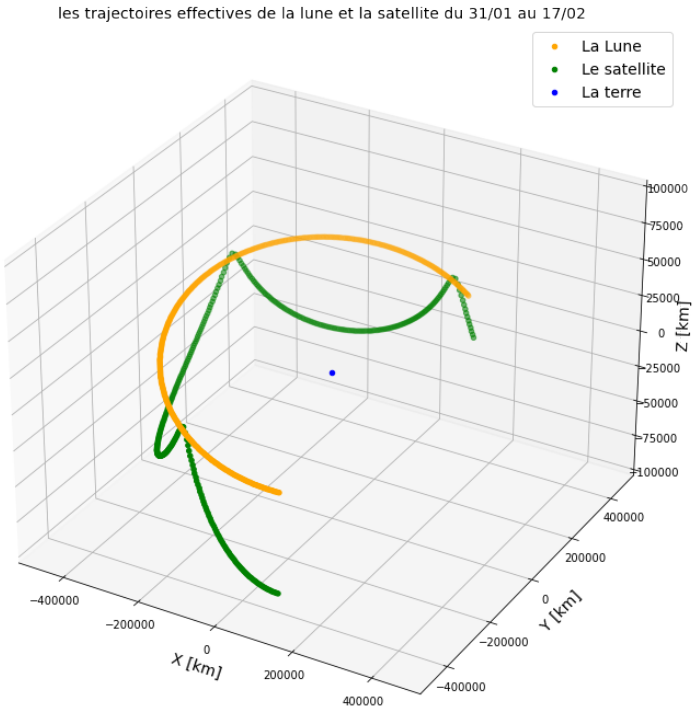


FIGURE 13 – Les trajectoires réelles de la lune et du satellite

6.3 Les trajectoires simulées

```
1 G = 6.67e-11* (24*3600)**2 / (1000)**3# en km3 kg-1 jours-2
2 """
3 G = 6.67e-11 # en m3 kg-1 s-2
4 """
5 mt = 5.972e24 #en kg
6 ms = 25 #en kg
7 ml = 7.36e22 # en kg
```

Nous ajoutons la constante de Newton G (en faisant attention aux unités), la masse de la Terre, la masse de la Lune et la masse du satellite. Par le systèmes de six équations différentielles (3),

on code :

$$\begin{cases} \ddot{x_s} &= -G(m_t+m_s)\dfrac{x_s}{{r_s}^3} + Gm_l(\dfrac{x_l-x_s}{r_{ls}^3} - \dfrac{x_l}{r_l^3}) \\ \ddot{y_s} &= -G(m_t+m_s)\dfrac{y_s}{{r_s}^3} + Gm_l(\dfrac{x_l-x_s}{r_{ls}^3} - \dfrac{y_l}{r_l^3}) \\ \ddot{z_s} &= -G(m_t+m_s)\dfrac{z_s}{{r_s}^3} + Gm_l(\dfrac{z_l-z_s}{r_{ls}^3} - \dfrac{z_l}{r_l^3}) \\ \ddot{x_l} &= -G(m_t+m_l)\dfrac{x_l}{{r_l}^3} + Gm_s(\dfrac{x_s-x_l}{r_{sl}^3} - \dfrac{x_s}{r_s^3}) \\ \ddot{y_l} &= -G(m_t+m_l)\dfrac{y_l}{{r_l}^3} + Gm_s(\dfrac{x_s-x_l}{r_{sl}^3} - \dfrac{y_s}{r_s^3}) \\ \ddot{z_l} &= -G(m_t+m_l)\dfrac{z_l}{{r_l}^3} + Gm_s(\dfrac{z_s-z_l}{r_{sl}^3} - \dfrac{z_s}{r_s^3}) \end{cases} \tag{5}$$

```
def deriv(t,u):
    '''
        Soit u = (xs,ys,zs,xs1,ys1,zs1,xl,yl,zl,xl1,yl1,zl1),
        Équation d'évolution de l'oscillateur harmonique :  du/dt =
        d(xs,ys,zs,xs1,ys1,zs1)/dt = (xs1,ys1,zs1,-G * (mt+ms) * xs/|rs|^3 + G * ml
        ((xl-xs)/(|rl-rs|)^3 - xl/|rl|^3),...,...)

    '''
    du = np.empty(u.size)

    # Dérivée de la vitesse
    du[0] = u[3]
    du[1] = u[4]
    du[2] = u[5]
    rs = np.sqrt(u[0]**2 +u[1]**2 + u[2]**2)
    rls = np.sqrt((u[6]-u[0])**2+(u[7] - u[1])**2+(u[8] - u[2])**2)
    rsl = rls
    rl = np.sqrt(u[6]**2 + u[7]**2 + u[8]**2)
    du[3] = -G * (mt + ms) * u[0] /rs**3  + G*ml*((u[6]-u[0])/(rls)**3 -(u[6]/rl**3))
    du[4] = -G * (mt + ms) * u[1] /rs**3  + G*ml*((u[7]-u[1])/(rls)**3 -(u[7]/rl**3))
    du[5] = -G * (mt + ms) * u[2] /rs**3  + G*ml*((u[8]-u[2])/(rls)**3 -(u[8]/rl**3))
    du[6] = u[9]
    du[7] = u[10]
    du[8] = u[11]
    du[9] = -G * (mt + ml) * u[6] /rl**3  + G*ms*((u[0]-u[6])/(rsl)**3 -(u[0]/rs**3))
    du[10] = -G * (mt + ml) * u[7] /rl**3  + G*ms*((u[1]-u[7])/(rsl)**3 -(u[1]/rs**3))
    du[11] = -G * (mt + ml) * u[8] /rl**3  + G*ms*((u[2]-u[8])/(rsl)**3 -(u[2]/rs**3))
    return du
```

Ensuite, nous définissons la fonction rk4.

```
def rk4(x, dx, y, deriv):
    """
      /*----------------------------------------
      sous programme de resolution d'équations
      différentielles du premier ordre par
      la methode de Runge-Kutta d'ordre 4
      x = abscisse, une valeur scalaire, par exemple le temps
      dx = pas, par exemple le pas de temps
      y = valeurs des fonctions au temps t(i), c'est un tableau numpy de taille n
      avec n le nombre d'équations différentielles du 1er ordre

      rk4 renvoie les nouvelles valeurs de y pour t(i+1)

      deriv = variable contenant le nom du
      sous-programme qui calcule les derivees
      deriv doit avoir deux arguments: deriv(x,y) et renvoyer
      un tableau numpy dy de taille n
      ----------------------------------------*/
    """
    #  /* d1, d2, d3, d4 = estimations des derivees
    #    yp = estimations intermediaires des fonctions */
    ddx = dx/2.        #         /* demi-pas */
    d1 = deriv(x,y)    #       /* 1ere estimation */
    yp = y + d1*ddx
    #    for  i in range(n):
    #        yp[i] = y[i] + d1[i]*ddx
    d2 = deriv(x+ddx,yp)      #/* 2eme estimat. (1/2 pas) */
    yp = y + d2*ddx
    d3 = deriv(x+ddx,yp)  #/* 3eme estimat. (1/2 pas) */
    yp = y + d3*dx
    d4 = deriv(x+dx,yp)      #  /* 4eme estimat. (1 pas) */
    #/* estimation de y pour le pas suivant en utilisant
    #  une moyenne ponderee des derivees en remarquant
    #  que : 1/6 + 1/3 + 1/3 + 1/6 = 1 */
    return y + dx*( d1 + 2*d2 + 2*d3 + d4 )/6
```

Ensuite, nous définissons la fonction intergrationEDO

```
def integrationEDO(start,end,step, v_ini, deriv):
    n = v_ini.size
```

```
    # nombre d'equa-diff du 1er ordre

    # Création du tableau temps
    interval = end - start                      # Intervalle
    num_points = int(interval / step) + 1       # Nombre d'éléments
    t = np.linspace(start, end, num_points)     # Tableau temps t

     # Initialisation du tableau v
    v = np.empty((n, num_points))
    # tableau y qui contient les valeurs actuelles de v au temps t[i]
    y = np.empty(n)

     # Condition initiale
    v[:, 0] = v_ini
    y[:] = v_ini # important de mettre [:] pour y,
                 # sinon y serait juste un alias de v_ini et non une copie
                 # ce qui ferait qu'on modifie v_ini quand on modifie y et
                 # donc aussi le u_ini donné à la fonction lors de son appel.

    # Boucle for
    for i in range(num_points - 1):
        y = rk4(t[i], step, y, deriv) # modifie le tableau y
        v[:, i + 1] = y

    # Argument de sortie
    return t, v
```

À la fin, nous définissons la fonction résultat qui renvoie u.

```
def resultat(tmax, dt):
    # Conditions initiales
    u_ini = np.array([xs,ys,zs,vxs,vys,vzs,xl,yl,zl,vxl,vyl,vzl])

    # Méthode d'Euler
    t, u = integrationEDO(0, tmax, dt, u_ini, deriv)
    return u
# Représentation de la solution numérique
u = resultat(17,0.0001)
"""
on pose step = 0.0001 joursenviron 9 secondes, assez petit, 17 * 24 * 3600 signifie qu'il y a
totalement 17 jours (du 00h00,31/01 au 00h00,17/02)
```

```
"""
```

Pour faciliter l'observation, nous traçons la figure avec la fonction *scatter3D*.

```
# Ensuite, nous allons tracer la figure avec la solution numérique
fig = plt.figure(figsize = (12, 12)) #la taille de la figure
ax = plt.axes(xlim=(-5e5,5e5),ylim=(-5e5,5e5),zlim = (-1e5,1e5),projection ="3d")
ax.scatter3D(u[6,:],u[7,:],u[8,:], s = 10, color = "orange",label = "La Lune que nous simulons")
ax.scatter3D(u[0,:], u[1,:], u[2,:], s =10 ,color = "green",label = "Le satellite que nous simulons")
ax.scatter3D([0],[0],[0],color = "black", s=10, label = "La terre")
ax.scatter3D(x_satellite,y_satellite,z_satellite,s = 10, color = "blue", label ="Le satellite effectif"
ax.scatter3D(x_lune,y_lune,z_lune,s=10, color = "red", label ="La lune effective")
ax.set_xlabel("X [km]",fontsize = 14)
ax.set_ylabel("Y [km]",fontsize = 14)
ax.set_zlabel("Z [km]",fontsize = 14)
plt.legend(fontsize=14, markerscale=2., scatterpoints=1)
plt.title("les trajectoires simulées et effectives par python de la lune et le satellite du 31/01 au 17
plt.show()
```

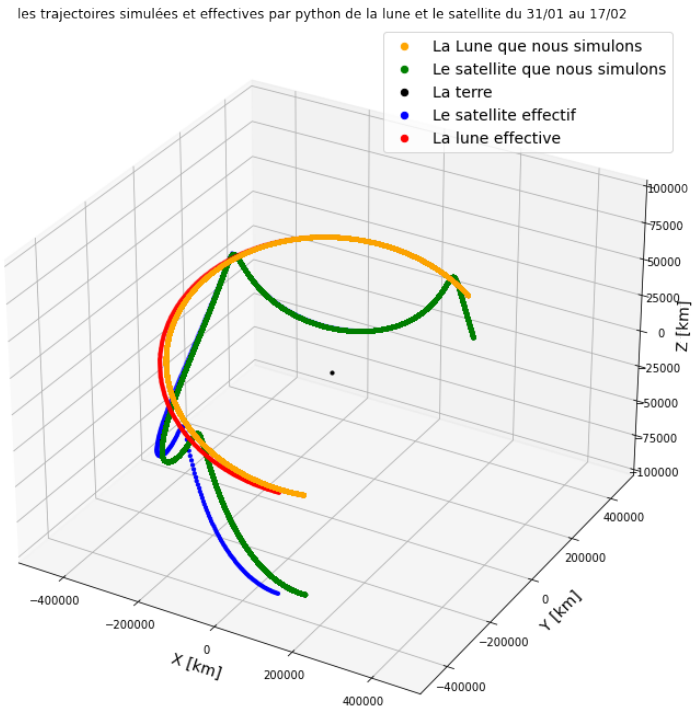


FIGURE 14 – Les trajectoires simulées et effectives par python de la lune et le satellite du 31/01 au 17/02

En regardant la figure 13, la courbe orange est celle de la trajectoire calculée de la Lune , la courbe verte est celle de la trajectoire calculée du satellite, la courbe rouge est celle de la trajectoire réelle de la Lune et la courbe bleue est celle de la trajectoire réelle du satellite. Nous

pouvons constater que cela semble bien simuler la réalité. Pour obtenir une image plus claire de notre simulation, nous avons dessiné un graphique comparant la coordonnée x réelle et calculée en fonction du temps.

```
start = 0 # temps initials
end = 17 # totale 17 jours
step1 = 0.0001 # le step que nous posons
step2 = 1/24 # step de .csv
interval = end - start                    # Intervalle
num_points1 = int(interval / step1) + 1      # Nombre d'éléments
num_points2 = int(interval / step2) + 1
t1 = np.linspace(start, end, num_points1)    # Tableau temps t
t2 = np.linspace(start, end, num_points2)    # Tableau temps t
plt.figure(figsize = (10, 10))
plt.plot(t1,u[0,:],label ="Le satellite que nous simule")
plt.plot(t2,x_satellite,label = "Le satellite réel")
plt.xlabel ("Temps[jours]",fontsize=14)
plt.ylabel ("X[km]",fontsize=14)
plt.legend(fontsize=14, markerscale=2., scatterpoints=1)
plt.grid()
plt.show()
```

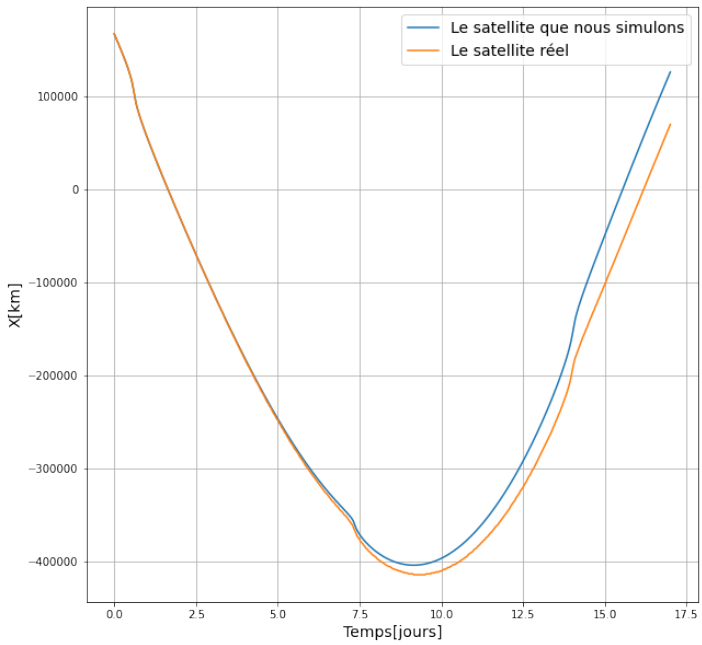


FIGURE 15 – Les trajectoires simulées par python et réelles de la Lune et du satellite du 31/01/23 au 17/02/23

En regardant la figure 14, nous avons remarqué qu'il y a des incertitudes à partir du 7^e jour environ, mais qu'elles sont acceptables car il y a beaucoups de facteurs non considerés qui influent sur ce système. Par exemple : la force de gravité exercée par le Soleil. De plus, la

valeur du pas de temps n'est pas optimale. Pour la suite, nous avons donc choisi un nouveau dt en cherchant pour quelle valeur de pas l'écart-type de l'énergie mécanique du système est minimale.

6.4 Trouver le *step* parfait

Nous définissons la fonction *Et* pour calculer la variance de la liste avec la moyenne (ici, nous prenons E_0 comme moyenne) et par la formule de l'écart-type :

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(E_i - E_0)^2} \tag{6}$$

```
def Et(liste,E0): # pour obtenir l'écart-type dans une liste, E0 est l'énergie quand t = 0
    l = 0 # pour la somme de (E_i - E_0)^2
    for i in range(len(liste)): # len(liste) signifie le nombre de l'élements
        var2 = (liste[i] - E0)**2
        l = l + var2
    return np.sqrt(l/len(liste))
```

Nous définissons la fonction *Ec* pour obtenir l'énergie cinétique avec leur vitesses et par la formule :

$$E_c = \frac{1}{2}m(v_x^2 + v_y^2 + v_z^2) \tag{7}$$

```
def Ec(m,vx,vy,vz):
    return m*(vx**2+vy**2+vz**2)/2
```

Ensuite, nous définissons les deux fonctions *dis* et *Ep*,*dis* représentant la distance entre deux points dans le système de coordonnées cartésiennes (formule 8) et *Ep* représentant l'énergie potentielle totale de la Lune et du satellite dans le vide

$$D = \sqrt{(x - x_1)^2 + (y - y_1)^2 + (z - z_1)^2} \tag{8}$$

Pour le problème à N+1 corps, la formule de l'énergie potentielle est :

$$E_p = -\sum_{i=1}^{N}\frac{GM_i m}{r_i} \tag{9}$$

```
def Ep(x,y,z,x1,y1,z1):
    pot = - G*mt*ml/dis(x1,y1,z1,0,0,0)-2*G*ml*ms/dis(x,y,z,x1,y1,z1)-G*mt*ms/dis(x,y,z,0,0,0)
    return pot
```

Nous avons donc E_0 Pourquoi ?

```
E0 = Ep(xs,ys,zs,xl,yl,zl) + Ec(ml,vxl,vyl,vzl) + Ec(ms,vxs,vys,vzs)
```

Pour xs,ys,zs ...., regardez la section *Lecture des données*, ce sont les conditions initiales.

A éviter, écrire : la section... contient les conditions initiales

```
liste = []
liste1 = []
i = 0.001 # jours
v1 = 0
v2 = 0
v3 = 0 # Nous posons 3 variables pour obtenir le step si l'écart-type est min
while i <= 3:
    u = resultat(17,i)
    for j in range(len(u[0,:])):
        E_1 = Ep(u[0,:][j],u[1,:][j],u[2,:][j],u[6,:][j],u[7,:][j],u[8,:][j])
        E_2 = Ec(ms,u[3,:][j],u[4,:][j],u[5,:][j]) + Ec(ml,u[9,:][j],u[10,:][j],u[11,:][j])
        E_t = E_1 + E_2
        liste.append(E_t)
    v3 = Et(liste,E0)
    liste1.append(v3)
    if i == 0.001:
        v1 = v3 # v1 = liste1[0]
    if v3 <= v1: # pour trouver v3 min
        v1 = v3
        v2 = i # ici, v2 est le step "parfait"
    i = i + 0.001
res = liste1
print("Le step 'parfait' est",'%.3f'% v2,"en jours")
```

le *step* 'parfait' est 2.387 en jours

```
t1 = np.linspace(0.001,3,len(res)) # tableau du temps
plt.figure(figsize = (10, 10)) # la taille de l'image
plt.plot(t1,res,'o') # nous la tracons
plt.xlabel("dt en jour",fontsize = 14)
plt.ylabel("l'écart-type des énergie mécanique du système (du satellite et la lune)", fontsize = 14)
plt.grid()
plt.show()
```

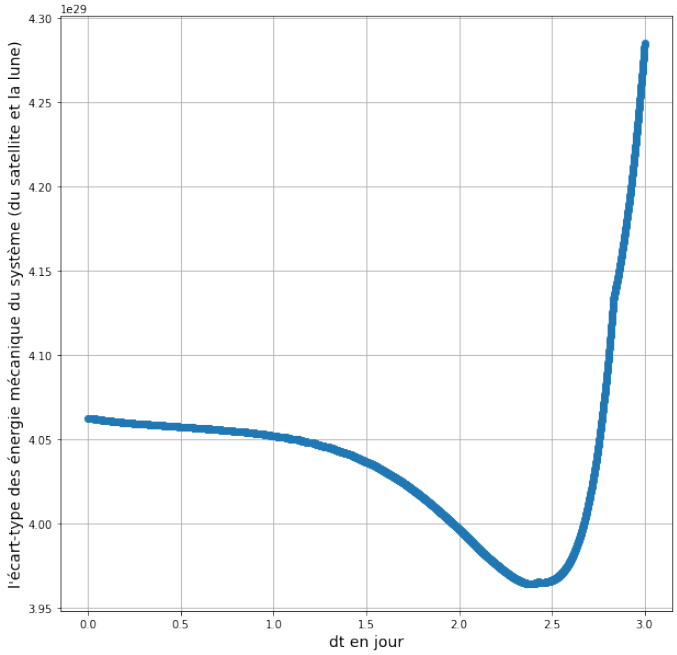


FIGURE 16 – L'écart-type (Lune et satellite) par les différents *dt*

Le pas idéal obtenu par python est de 2,387 en jours. Ensuite, nous avons répété les étapes de *cas : dt = 0,0001 jours* pour obtenir la trajectoire simulée.

```
step = v2# en jours
u = resultat(17,step)
# Représentation de la solution numérique
fig = plt.figure(figsize = (12, 12)) #la taille de la figure
ax = plt.axes(xlim=(-5e5,5e5),ylim=(-5e5,5e5),zlim = (-5e5,5e5),projection ="3d")
ax.scatter3D(u[6,:],u[7,:],u[8,:], s = 10, color = "orange",label = "La Lune que nous simulons")
ax.scatter3D(u[0,:], u[1,:], u[2,:], s =10 ,color = "green",label = "Le satellite que nous simulons")
ax.scatter3D([0],[0],[0],color = "black", s=10, label = "La terre")
ax.scatter3D(x_satellite,y_satellite,z_satellite,s = 10, color = "blue", label ="Le satellite effectif"
ax.scatter3D(x_lune,y_lune,z_lune,s=10, color = "red", label ="La lune effective")
ax.set_xlabel("X [km]",fontsize = 14)
ax.set_ylabel("Y [km]",fontsize = 14)
ax.set_zlabel("Z [km]",fontsize = 14)
plt.legend(fontsize=14, markerscale=2., scatterpoints=1)
plt.title("les trajectoires simulées et effectives par python de la lune et la satellite du 31/01 au 17
plt.show()
```

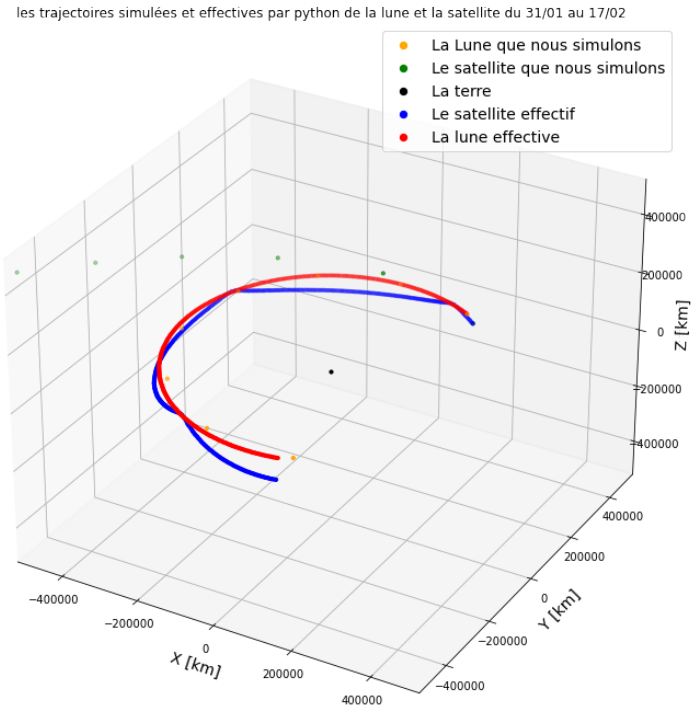


FIGURE 17 – la trajectoire du mouvement réel et simulé du satellite et de la Lune, si dt = 2,387 jours

Nous avons remarqué que même si nous avons modifié la valeur de dt pour minimiser l'écart-type global, le résultat n'était pas parfait. Pour vérifier davantage nos résultats, nous avons encore une fois extrait les valeurs de x selon x pour le satellite simulé et réel.

```
start = 0
end = 17 # totale 17 jours
step1 = v2 # le step qu'on pose
step2 = 1/24 # step de .csv
interval = end - start                      # Intervalle
num_points1 = int(interval / step1) + 1       # Nombre d'éléments
num_points2 = int(interval / step2) + 1
t1 = np.linspace(start, end, num_points1)     # Tableau temps t
t2 = np.linspace(start, end, num_points2)     # Tableau temps t
plt.figure(figsize = (10, 10))
plt.plot(t1,u[0,:],'o-',label ="Le satellite que nous simulons")
plt.plot(t2,x_satellite,label = "Le satellite réel")
plt.xlabel ("Temps [jours]",fontsize=14)
plt.ylabel ("X [km]",fontsize=14)
plt.legend(fontsize=14, markerscale=2., scatterpoints=1)
plt.grid()
plt.show()
```

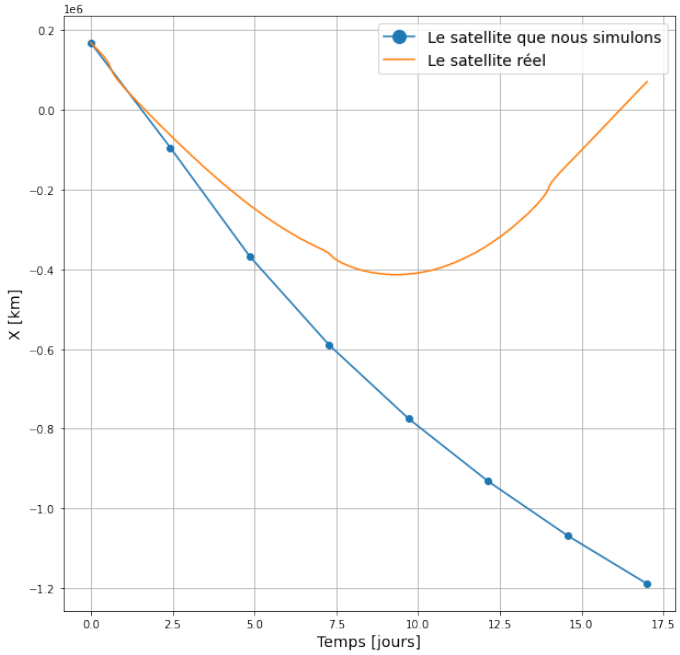


FIGURE 18 – la trajectoire de mouvement réelle et simulée du satellite et de la lune selon x, si dt = 2,387 jours

Nous avons constaté qu'à partir du premier jour, il y avait une déviation très importante. Comme la masse du satellite (25kg) que nous avons considéré est très faible par rapport à celle de la lune (7,36e22 kg) ; lorsque la valeur de l'écart-type est minimale pour l'ensemble (Lune et satellite), cela signifie que l'énergie mécanique de la lune est plus proche de la réalité, car l'impact mécanique du satellite est négligeable par rapport à celui de la lune. Par exemple, pour un objet tombant librement sur la Terre, en négligeant bien sûr la résistance de l'air, nous considérons que son énergie mécanique est conservée. Bien qu'il subisse l'attraction gravitationnelle de la Lune et que la lune subisse également l'attraction gravitationnelle de cet objet, nous ne considérons pas que l'énergie mécanique du système Objet-Lune est conservée mais que l'énergie mécanique de l'objet est conservée. Donc, nous redéfinissons la fonction *Ep*,l'énergie potentielle du satellite.

```
def Ep(x,y,z,xl,yl,zl): #la redéfinition de la fonction Ep
    pot = - G*mt*ms/dis(x,y,z,0,0,0) - G*ml*ms/dis(xs,ys,zs,xl,yl,zl)
    #-G*ml*ms/dis(xs,ys,zs,xl,yl,zl) - - G*mt*ml/dis(xl,yl,zl,0,0,0)
    return pot
E0 = Ep(xs,ys,zs,xl,yl,zl) + Ec(ms,vxs,vys,vzs)
```

Puis de la même manière qu'avant, nous trouvons un *dt*.

```
liste = []
liste1 = []
i = 0.00005 # jours
v1 = 0
v2 = 0
v3 = 0 # Nous posons 3 variables pour obtenir le step si l'écart-type est min
```

```
while i <= 1/24: # t_max = 1 heure
    u = resultat(17,i)
    for j in range(len(u[0,:])):
        E_1 = Ep(u[0,:][j],u[1,:][j],u[2,:][j],u[3,:][j],u[4,:][j],u[5,:][j])
        E_2 = Ec(ms,u[3,:][j],u[4,:][j],u[5,:][j])
        E_t = E_1+E_2
        liste.append(E_t)
    v3 = Et(liste,E0)
    liste1.append(v3)
    if i == 0.00005:
        v1 = v3 # v1 = liste1[0]
    if v3 <= v1: # pour trouver v3 min
        v1 = v3
        v2 = i # ici, v2 est le step "parfait"
    i = i + 0.00005  # 0.00005 est assez petit, en effet plus petit, plus mieux
res = liste1
```

Et nous affichons ce *dt* réel.

```
print("Le step 'parfait' est",'%.4f'% v2,"en jours")
t1 = np.linspace(0.00005,1/24,len(res)) # tableau du temps
plt.figure(figsize = (10, 10)) # la taille de l'image
plt.plot(t1,res,'o') # Tracer
plt.xlabel("dt (step) en jours",fontsize = 14)
plt.ylabel("L'écart-type des énergie mécanique du satellite", fontsize = 14)
plt.grid()
plt.show()
```

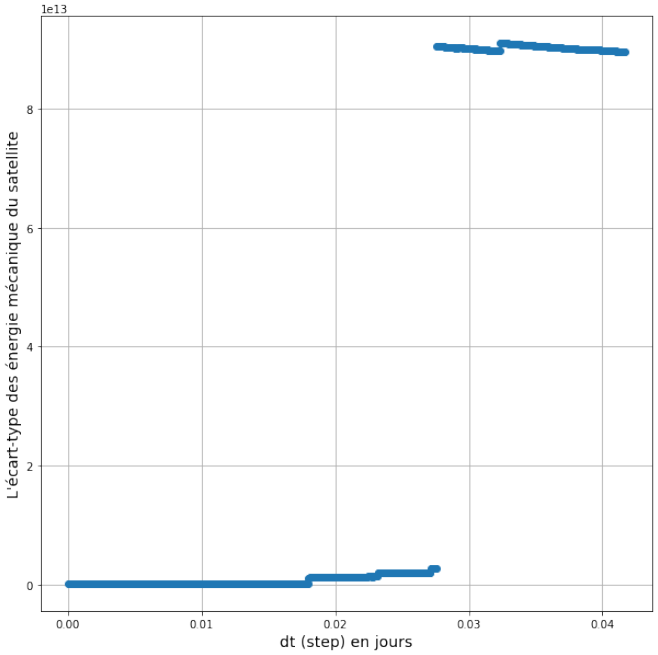


FIGURE 19 – L'écart-type (satellite) par les différents *dt*

Le pas idéal obtenu est 0,0072 jour. Ensuite, nous avons répété les étapes de *cas : dt = 0,0001 secondes* pour obtenir la trajectoire simulée.

```
# Représentation de la solution numérique
u = resultat(17,v2) # Le resultat du cas: step = 0.0072 jours
fig = plt.figure(figsize = (12, 12)) #la taille de la figure
ax = plt.axes(xlim=(-5e5,5e5),ylim=(-5e5,5e5),zlim = (-1e5,1e5),projection ="3d")
ax.scatter3D(u[6,:],u[7,:],u[8,:], s = 10, color = "orange",label = "La Lune que nous simulons")
ax.scatter3D(u[0,:], u[1,:], u[2,:], s =10 ,color = "green",label = "Le satellite que nous simulons")
ax.scatter3D([0],[0],[0],color = "black", s=10, label = "La terre")
ax.scatter3D(x_satellite,y_satellite,z_satellite,s = 10, color = "blue", label ="Le satellite effectif"
ax.scatter3D(x_lune,y_lune,z_lune,s=10, color = "red", label ="La lune effective")
ax.set_xlabel("X [km]",fontsize = 14)
ax.set_ylabel("Y [km]",fontsize = 14)
ax.set_zlabel("Z [km]",fontsize = 14)
plt.legend(fontsize=14, markerscale=2., scatterpoints=1)
plt.title("les trajectoires simulées et effectives par python de la lune et du satellite du 31/01 au 17
plt.show()
```

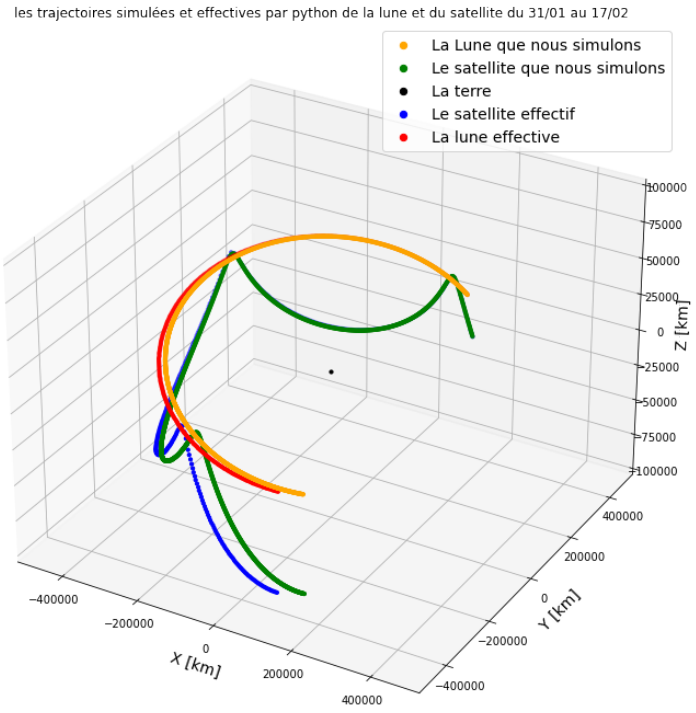


FIGURE 20 – la trajectoire du mouvement réel et simulé du satellite et de la Lune, si dt = 0,0072 jours

Cela semble similaire à *cas : dt = 0,0001 jours*. Pour une vérification supplémentaire, nous avons encore tracé le graphique selon x pour comparer.

```
u1 = resultat(17,0.0001) # resultat du cas : step = 0,0001 jours
u2 = resultat(17,2.387) #  resultat du cas : step = 2.387 jours
start = 0
end = 17 # totale 17 jours
step1 = v2 # le step qu'on pose en jours
step2 = 1/24 # step de .csv en jours
step3 = 0.0001 # le step initial en jours
step4 = 2.387 # le step dernier en jours
interval = end - start                      # Intervalle
num_points1 = int(interval / step1) + 1       # Nombre d'éléments
num_points2 = int(interval / step2) + 1
num_points3 = int(interval / step3) + 1
num_points4 = int(interval / step4) + 1
t1 = np.linspace(start, end, num_points1)     # Tableau temps t
t2 = np.linspace(start, end, num_points2)     # Tableau temps t
t3 = np.linspace(start, end, num_points3)     # Tableau temps t
t4 = np.linspace(start, end, num_points4)     # Tableau temps t
plt.figure(figsize = (10, 10))
plt.plot(t1,u[0,:],label ="Le satellite que nous simulons, si step = 0.0072 jours")
```

```
plt.plot(t3,u1[0,:],label ="Le satellite que nous simulons, si step = 0.0001 jours")
plt.plot(t4,u2[0,:],"o-",label ="Le satellite que nous simulons, si step = 2.387 jours")
plt.plot(t2,x_satellite,label = "Le satellite réel")
plt.xlabel ("Temps [jours]",fontsize=14)
plt.ylabel ("X [km]",fontsize=14)
plt.legend(fontsize=14, markerscale=2., scatterpoints=1)
plt.grid()
plt.show()
```

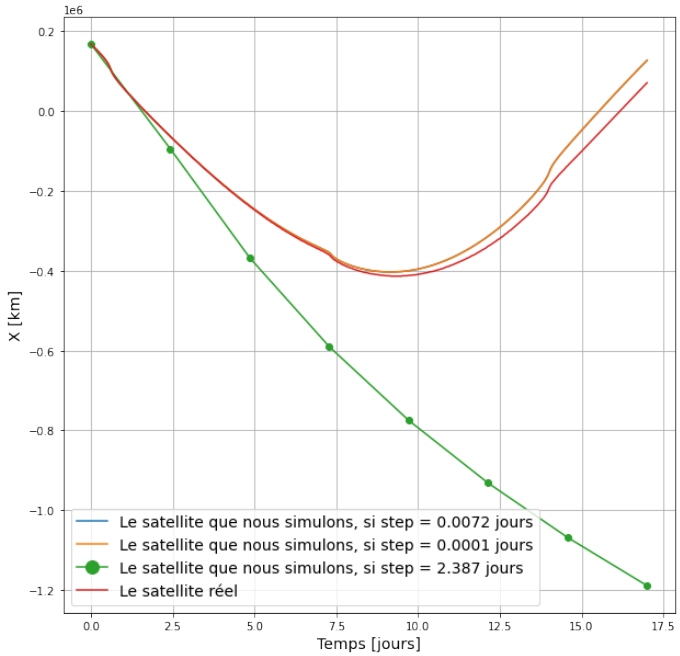


FIGURE 21 – la trajectoire du mouvement réel et simulé du satellite et de la Lune selon x, si dt = 0,001 ; 0,0072 et 2,387 en jours

En regardant FIGURE 21,la courbe bleue et la courbe orange se superposent presque complètement. En regardant la figure 18, nous pouvons également constater que pour les valeurs du pas de temps = 0,0001 jour et 0,0072 jour, l'écart-type global est très proche de zéro.

```
plt.figure(figsize = (10, 10)) # la taille de l'image
plt.plot(t1,u[0,:],'o-',color = "blue",label ="le satellite que nous simulons, si step = 0.0072 jours")
plt.plot(t3,u1[0,:],'o-',color = "orange",label ="le satellite que nous simulons, si step = 0.0001 jour
plt.xlabel ("Temps [jours]",fontsize=14)
plt.ylabel ("X [km]",fontsize=14)
plt.xlim(15.71,15.72) #  Les limites de "x" que nous fixons
plt.ylim(15420,15490) # Les limites de "y" que nous fixons
plt.grid()
plt.show()
```

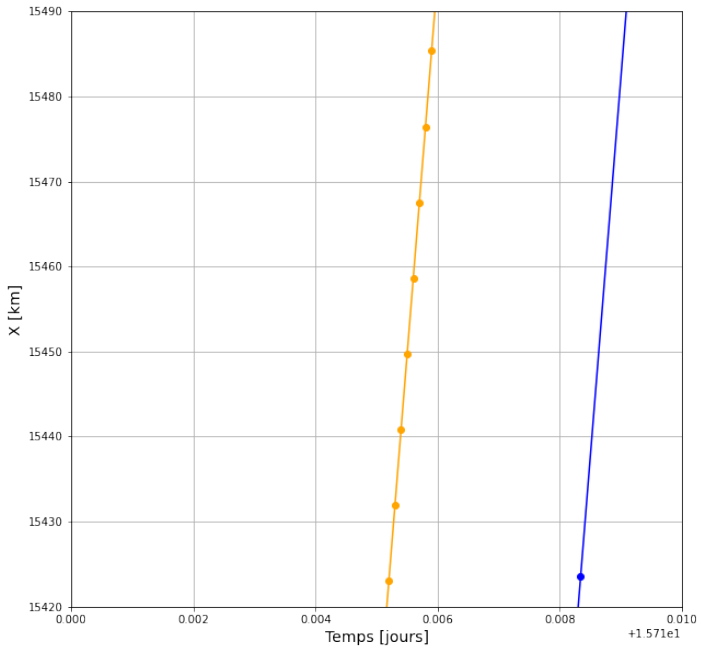


FIGURE 22 – la trajectoire du mouvement réel et simulé du satellite et de la Lune selon x, si dt = 0,001 ; 0,0072(Version agrandie)

En observant la FIGURE 22, nous remarquons que les points bleus sont en dessous des points oranges au même moment, ce qui signifie que la courbe bleue est plus proche de la situation réelle.

7 Références

Références

[1] D. G. BETTIS. "A Special Perturbation Method : m-Fold Runge-Kutta". en. In : *Dynamics of Planets and Satellites and Theories of Their Motion*. Sous la dir. de Victor SZEBEHELY. Astrophysics and Space Science Library. Dordrecht : Springer Netherlands, 1978, p. 157-157. ISBN : 9789400998094. DOI : 10.1007/978-94-009-9809-4_18.

[2] *Horizons System*. English. URL : https://ssd.jpl.nasa.gov/horizons/app.html#/ (visité le 14/03/2023).

[3] Svein LINGE et Hans Petter LANGTANGEN. "Solving Ordinary Differential Equations". en. In : *Programming for Computations - Python : A Gentle Introduction to Numerical Simulations with Python 3.6*. Sous la dir. de Svein LINGE et Hans Petter LANGTANGEN. Texts in Computational Science and Engineering. Cham : Springer International Publishing, 2020, p. 203-285. ISBN : 9783030168773. DOI : 10.1007/978-3-030-16877-3_8. URL : https://doi.org/10.1007/978-3-030-16877-3_8 (visité le 14/03/2023).

[4] *Problème à N corps*. fr. Page Version ID : 201128914. Fév. 2023. URL : https://fr.wikipedia.org/w/index.php?title=Probl%C3%A8me_%C3%A0_N_corps&oldid=201128914 (visité le 14/03/2023).

[5] Martin H. WEIK. "n-body problem". en. In : *Computer Science and Communications Dictionary*. Sous la dir. de Martin H. WEIK. Boston, MA : Springer US, 2001, p. 1074-1074. ISBN : 9781402006135. DOI : 10.1007/1-4020-0613-6_12120. URL : https://doi.org/10.1007/1-4020-0613-6_12120 (visité le 14/03/2023).