Calcul précis de la trajectoire de la Terre en fonction du temps

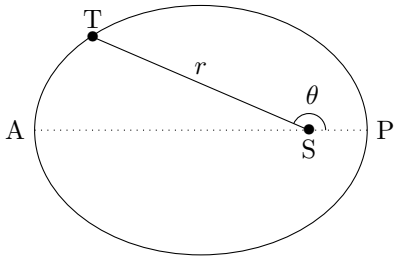
Arthur Barjot

2021

Contents

1 Introduction

Je vais présenter dans ce document la méthode que j'ai suivi pour établir une fonction qui à une date t renvoie les coordonnées de la Terre dans le référentiel héliocentrique (ayant choisi une position initiale). Pour cela nous allons utiliser la mécanique Newtonienne et des outils tels que l'intégration numérique. Le but est donc de déterminer une équation "horaire" de la trajectoire de la Terre; pour nous, cela consistera à transformer une relation à variable polaire en deux fonctions du temps. Nous chercherons dans un premier temps la distance Terre-Soleil (distance r du schéma) en fonction du temps, puis l'angle entre les vecteurs $\overrightarrow{SP}$ et $\overrightarrow{ST}$ (l'angle θ du shéma) s'en déduira (lui aussi en fonction du temps). Avec S le Soleil, T la Terre, P et A les positions de la Terre aux périhélie et aphélie de la trajectoire.



La trajectoire de la Terre est très bien connue en coordonnées polaires, c'est à dire la distance Terre-Soleil (r) en fonction de θ, car cette trajectoire est une ellipse, qui, comme toute conique, a une équation polaire de la forme :

$$r(\theta) = \frac{p}{1 + ecos(\theta)}$$

p est le paramètre de la conique et e son excentricité, que l'on sait exprimer avec les grandeurs mécaniques.

2 Équation différentielle

Il nous faut maintenant exprimer r et θ en fonction du temps t, mais comme nous venons de le voir, ces deux fonctions sont liées, de l'une on pourra déduire l'autre, exprimons donc $r(t)$.
Nous pouvons exprimer, à l'aide de la mécanique Newtonienne, l'énergie mécanique de la Terre lorsqu'elle ne subit qu'une force centrale (attraction du Soleil) :

$$Em = \frac{1}{2}m\dot{r}^2 + \frac{mC^2}{2r^2} - \frac{k}{r}$$

Avec $\dot{r}$, la dérivée de r par rapport au temps, m la masse de la Terre, C la constante des aires, et $k = GM_sm$, G la constante gravitationnelle et M_s la masse du Soleil.

On sait de plus que pour une trajectoire elliptique telle que celle étudiée, l'énergie mécanique est constante et peut s'exprimer :

$$Em = \frac{-k}{2a}$$

Avec a, le demi-grand axe de la parabole. On obtient :

$$\begin{aligned} \frac{-k}{2a} &= \frac{1}{2}m\dot{r}^2 + \frac{mC^2}{2r^2} - \frac{k}{r} \\ \dot{r} &= \sqrt{-\frac{k}{ma} + \frac{2k}{mr} - \frac{C^2}{r^2}} \\ &= \sqrt{C_1 + \frac{C_2}{r} + \frac{C_3}{r^2}} \end{aligned}$$

Avec,

$$C_1 = -\frac{k}{ma} \quad C_2 = \frac{2k}{m} \quad C_3 = -C^2$$

des constantes parfaitement calculables numériquement.

Malheureusement, les mathématiques actuelles ne permettent pas d'intégrer ce type d'équation, nous allons donc devoir utiliser une méthode numérique.

3 Intégration numérique

Nous allons alors nous intéresser à la méthode de Runge-Kutta d'ordre 4, cette méthode est dérivée de la méthode d'Euler, et vient perfectionner celle-ci. C'est aujourd'hui l'une des méthodes d'intégration numérique les plus efficace. Voici une implémentation possible sous Python (adapté à notre problème):

```
import numpy as np

C1=-887384455.8905718 ;
C2=2.6549172e+20 ;
C3=-1.9852204086241444e+31

def fonction(r):
    x=(C1+C2/r+C3/r**2)**(1/2)
    return x


def rungekutta4(f, r0, t0, tf, nb):
#(fonction à intégrer; r initial; t initial; t final; nombre de points)
    T=np.linspace(t0,tf,nb)
    n = len(T)
    R = np.zeros(n)
    R[0] = (r0)
```

```
h=(T[len(T)-1]-T[0])/n
for i in range(n-1):
    k1 = f(R[i])
    k2 = f(R[i] + k1 * h / 2)
    k3 = f(R[i] + k2 * h / 2)
    k4 = f(R[i] + k3 * h)
    R[i+1] = R[i]+(h / 6) * (k1 + 2*k2 + 2*k3 + k4)
return R,T
```

Cette méthode prend en argument : la valeur initiale de r : r_0, la date de début d'intégration à laquelle r_0 est atteinte (il faudra minutieusement choisir ces valeurs), la date de fin d'intégration, la précision, et enfin la fonction à intégrer. Cette algorithme renvoie deux tableaux, l'un des dates et l'autre des distances Terre-Soleil. Les valeurs des deux tableaux nous permettent de tracer une courbe avec les temps en abscisse et les distances TS en ordonnée. Cette courbe ne sera pas continue, mais composée d'un ensemble de points. Avec un pas d'intégration assez grand, la courbe semblera continue (nous avons par la suite utiliser un pas de 10000).

4 Problème rencontré

4.1 Présentation

Nous allons utiliser la méthode d'Euler pour illustrer notre problème, car il est le même dans les deux méthodes. Pour calculer un point l'algorithme utilise le précédent, en effet : Si nous appellons $r' = f(r)$ l'équation différentielle à intégrer, r(t) son intégrale, $r(t_0), .., r(t_k), .., r(t_n)$ les valeurs discrètes renvoyées, et h le pas d'intégration. L'égalité qu'utilise la méthode d'Euler est :

$$r(t_{k+1}) = h \times r'(t_k) + r(t_k)$$

Le problème rencontré est que dans notre cas, $r'(t_0) = f(r_0) = 0$, en effet, nous choisissons notre origine des temps lorsque la Terre se trouve au périhélie de sa trajectoire et nous intégrons jusqu'à l'aphélie de celle-ci. Ce choix est fait car ces points sont bien connus, ils seront atteints pour nous à $t = 0$ et à $t = \frac{1}{2}an$, puis l'entièreté des points s'en déduira par symétrie d'axe AP (voir schéma ci-dessous). Par définition, le périhélie est la position de la Terre lorsqu'elle est au plus proche du Soleil : c'est un minimum pour la fonction $r(t)$ et on a bien $r'(t_0) = 0$. Par récurrence, on démontre que la méthode d'Euler, utilisée dans les conditions présentées ($r'(t_0) = 0$), renvoie une liste de n fois la valeur r_0. Ce problème peut s'étendre, de la même façon on démontre facilement que:

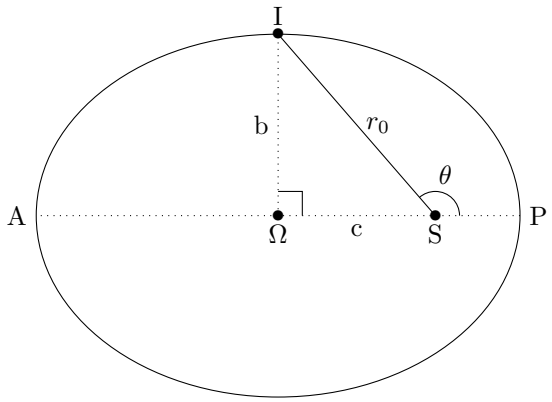
$$\exists p \in [0, n] : r'(t_p) = 0 \Rightarrow r(t_p) = r(t_{p+1}) = ... = r(t_n)$$

Ce problème fausse les résultats que renvoie notre algorithme.

4.2 Tentative de solution

- Nouvelle origine

  Nous avons alors tenté de choisir une nouvelle origine, différente de P, comme I par exemple :



  Ici, nous connaissons r_0, qui vaut $\sqrt{b^2 + c^2}$, b et c étant des caractéristiques calculables de l'ellipse. Le problème est que l'on peut dire, si on souhaite, que $t_0 = 0$ pour T se trouvant en I mais dans ce cas on ne connaît plus la date à laquelle la Terre arrive en A, où en P... Si seulement la vitesse était uniforme, le trajet de I à A durerait $\frac{1}{4}$ d'année...

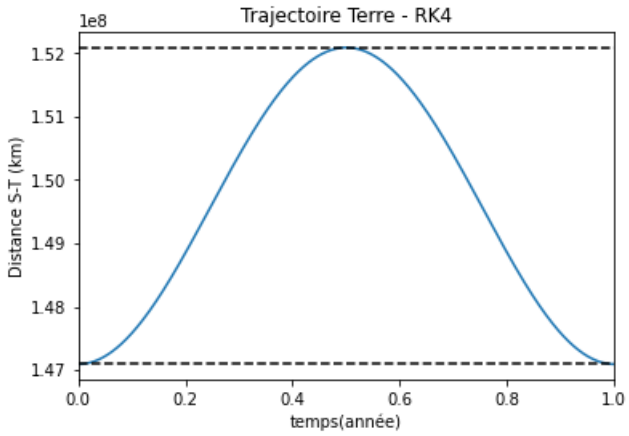
- Modification de r_0

  La meilleure solution que nous avons pu trouver a été de rester sur notre choix de la date $t_0 = 0$ lorsque la Terre se trouve au périgée de sa trajectoire (P du schéma ci-dessus) mais de modifier très légèrement la valeur de la distance Terre Soleil dans la compilation de l'algorithme, tout en gardant la vraie valeur pour les calculs établissant l'équation différentielle. Ainsi, la courbe fournie par python est bien celle recherchée. Nous avons ensuite cherché le plus petit $\Delta = |r_{périgée} - r_{modifiée}|$ tel que l'algorithme ne considère pas $r'(t_0) = 0$, et on trouve sur python:

  $$\Delta = 0.001053 \text{ (arrondi au millionième)}$$

  Δ étant en mètre, il s'agit ici d'une modification de seulement 1mm, quand r est de l'ordre de 150 million de kilomètres, ce Δ est ici négligeable.

Nous avons pu tracer les courbes formées des points fournis par la méthode de Runge-Kutta grâce au module matplotlib de Python, avec ici les valeurs extrêmes en pointillés :



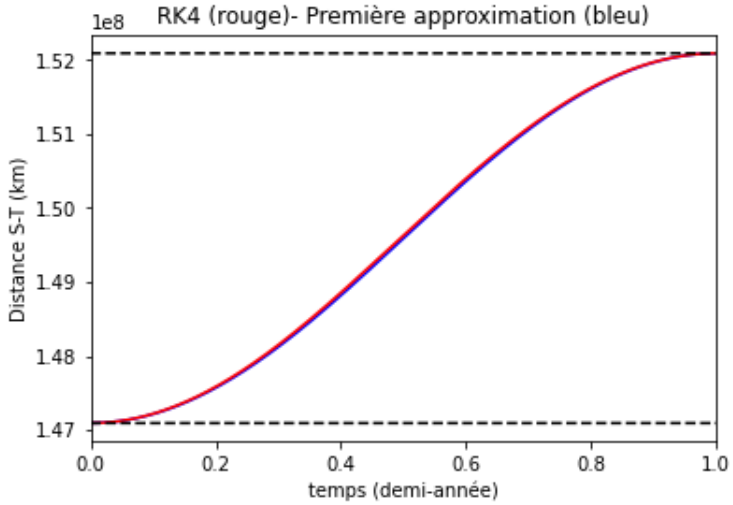
5 Approximation

La méthode de Runge-Kutta ne renvoie pas une fonction mais un ensemble de points car elle utilise une solution 'discrète' du problème. L'étape suivante, est donc de chercher une fonction qui à une date t,
entre 0 et 1 (en année), nous renvoie la valeur de la distance Terre-Soleil à cette date. Pour cela nous avons utilisé une méthode plutôt 'manuelle', c'est à dire que nous avons cherché une fonction usuelle qui s'approche au maximum de la courbe souhaitée, puis nous avons calculé l'écart entre les points de la trajectoire réelle et ceux de notre approximation, cet écart que nous allons à nouveau approcher par la même méthode, jusqu'à obtenir un écart acceptable et enfin sommer toutes les approximations.

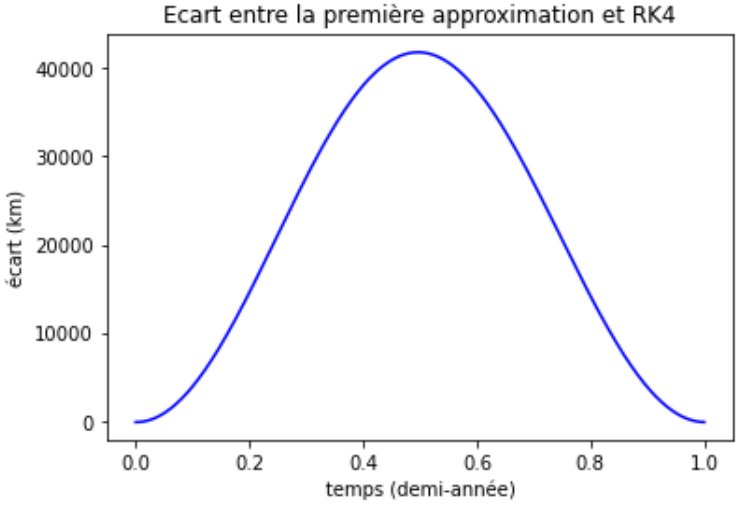
- Approche sinusoïdale

  Étant donnée la forme de la courbe obtenue (voir courbe ci-dessus) nous avons d'abord pensé à une sinusoïde, en modifiant quelques paramètres on obtient :

$$f1(x) = 149592275.5 - 2501131.5 \times \cos(\pi x)$$



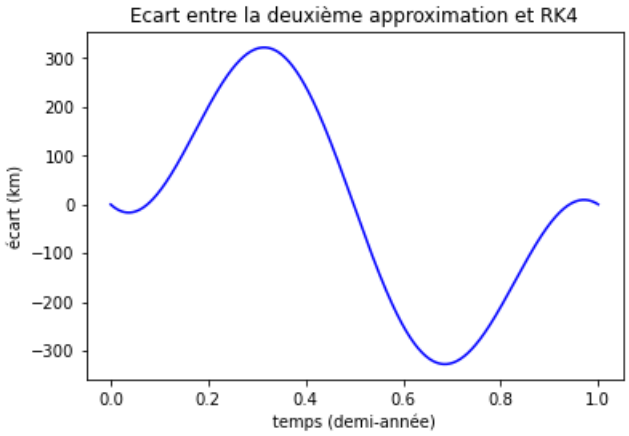
L'écart entre l'approximation et la réalité est alors :



On a encore un écart de 40000km par endroit par rapport à la réalité, ce n'est pas acceptable. Mais nous avons à nouveau identifié cette courbe à une sinusoïde :

$$f2(x) = 20890.230143353343 - 20890.230143353343 \times \cos(2\pi x)$$

Et le nouvel écart entre la somme des deux premières approximations et réalité est alors :



En sommant les deux approximations sinusoïdales on obtient un écart maximale de 300km, déjà petit par rapport au diamètre de la Terre (environ 12742 km), mais ce n'est pas encore acceptable. La courbe à approximer est encore clairement sinusoïdale mais elle ferait intervenir une somme de fonction sinus ou cosinus, ce qui n'est pas aussi simple à trouver que les approximations précédentes. Nous allons utiliser un autre moyen d'approximation.

- Approximation polynomiale

  Nous allons alors nous intéresser à méthode d'interpolation de Lagrange, et en proposer une implantation Python. Explication du principe de l'algorithme : à une liste de points d'un plan, il associe une fonction polynomiale passant par tous ces points. L'algorithme utilise plusieurs autres sous programmes nécessaires à son fonctionnement mais voici une portion de notre code :

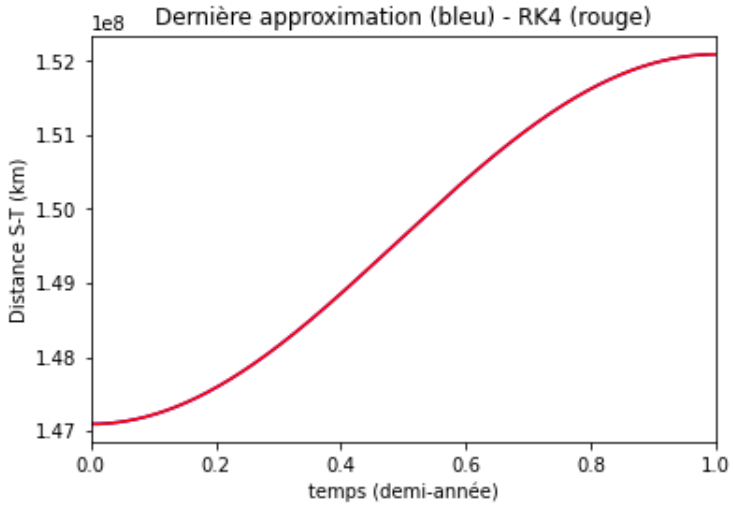
```
def lagrange(l,i):
    if i>=len(l):
        raise ValueError("xi n'est pas dans la liste")
    L=[1]
    for j in range(i):
        A=mult_scal([-l[j],1],1/(l[i]-l[j]))
        L=multiply(L,A)
    for j in range(i+1,len(l)):
        A=mult_scal([-l[j],1],1/(l[i]-l[j]))
        L=multiply(L,A)
    return L
```

```
def interpole(l,L):
    if len(L)!=len(l):
        raise ValueError("les listes n'ont pas la même taille")
    P=[0]
    for i in range(len(l)):
        A=mult_scal(lagrange(l,i),L[i])
        P=somme(P,A)
    return P
```

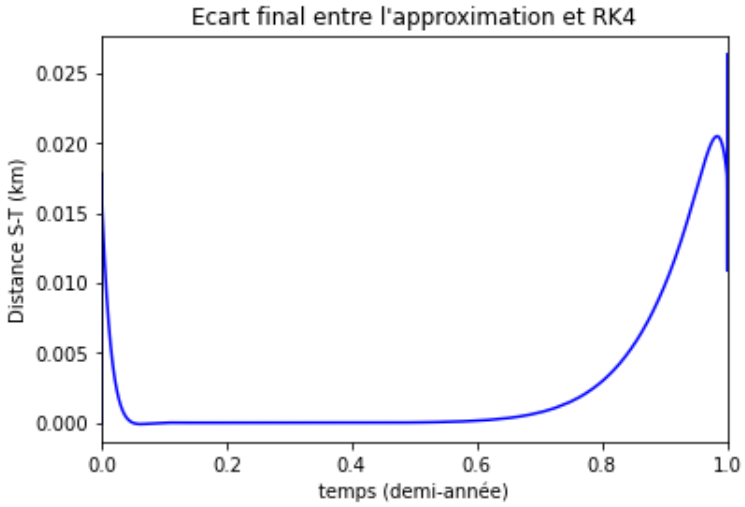
On obtient donc en appliquant deux fois l'approximation polynomiale notre fonction finale (somme des quatre approximations en tout) :

$$\begin{aligned} TS(t) =& 149613165.730144 - 291.843 \times t\pi + 1270.065 \times (t\pi)^2 + 72.904 \times (t\pi)^3 \\ & - 1026.934 \times (t\pi)^4 - 1026.934 \times (t\pi)^5 - 49.051 \times (t\pi)^6 + 483.358 \times (t\pi)^7 \\ & - 567.044 \times (t\pi)^8 + 416.296 \times (t\pi)^9 - 239.087 \times (t\pi)^{10} + 95.315 \times (t\pi)^{11} \\ & - 14.949 \times (t\pi)^{12} - 9.552 \times (t\pi)^{13} + 8.666 \times (t\pi)^{14} - 3.735 \times (t\pi)^{15} \\ & + 1.051 \times (t\pi)^{16} - 0.202 \times (t\pi)^{17} + 0.0258 \times (t\pi)^{18} - 0.002 \times (t\pi)^{19} \\ & + 6.948 \times 10^{-05} \times (t\pi)^{20} - 2501131.5 \times cos(t\pi) - 20890.230 \times cos(2t\pi) \end{aligned}$$

Cet fonction est certes un peu longue mais très précise (on a ici arrondi les coefficients). Voici le graphique de cette dernière approximation comparée à la courbe réelle :

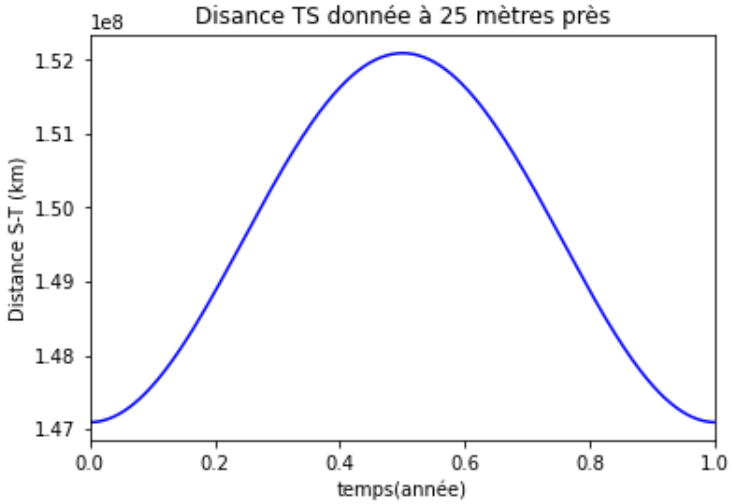


On ne peut plus distinguer les deux courbes qui ont pourtant des couleurs bien distinctes, en effet, l'écart approximation-réalité est maintenant :



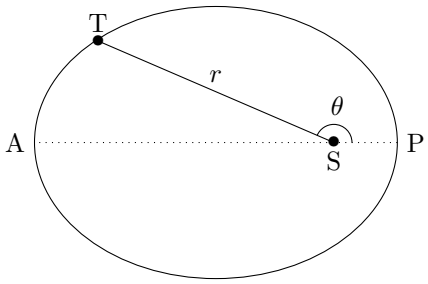
L'écart est maintenant de 25 mètres au maximum sur l'intégralité de la trajectoire de la Terre, cette distance comparée au diamètre de la Terre ou même à la taille d'une météorite (de l'ordre d'une dizaine de kilomètres) est négligeable.

On retient finalement :



6 Utilisation des donnés

Maintenant que nous connaissons précisément la distance Terre-Soleil en fonction du temps, soit r(t) si r est la première coordonnée polaire de la trajectoire de T :



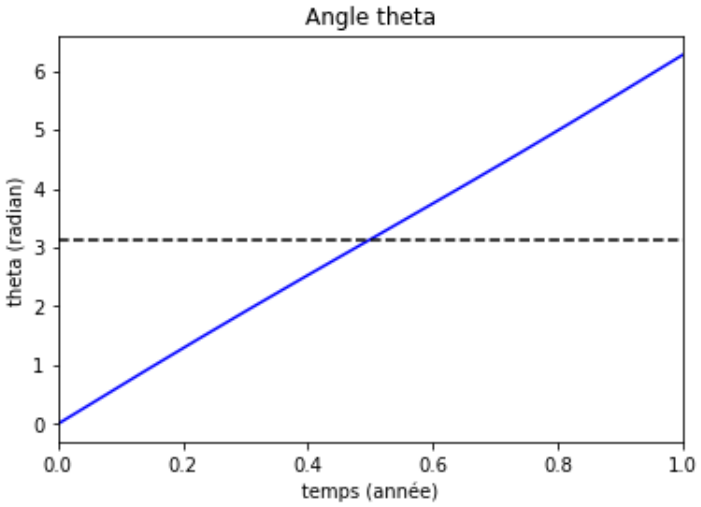
Nous pouvons aisément calculer θ(t) de part la formule reliant les coordonnées polaires :

$$r(\theta) = \frac{p}{1 + ecos(\theta)}$$

On trouve :

$$\theta(t) = \arccos\left(\frac{p - r(t)}{e \times r(t)}\right)$$

Aucun problème de définition pour le arccosinus, mais les valeurs de θ lors de la deuxième moitié de l'année doivent être entre π et 2π, des valeurs qui n'appartiennent pas à l'image de la fonction arccosinus, mais ce problème a été facilement réglé. On trouve une courbe presque linéaire pour $\theta(t)$:



Nous n'avons pas à approximer cette fois car nous pouvons réutiliser les résultats trouvés pour r(t), voici le programme python donnant $\theta(t)$, vous remarquerez la disjonction de cas due au problème rencontré avec le arccosinus:

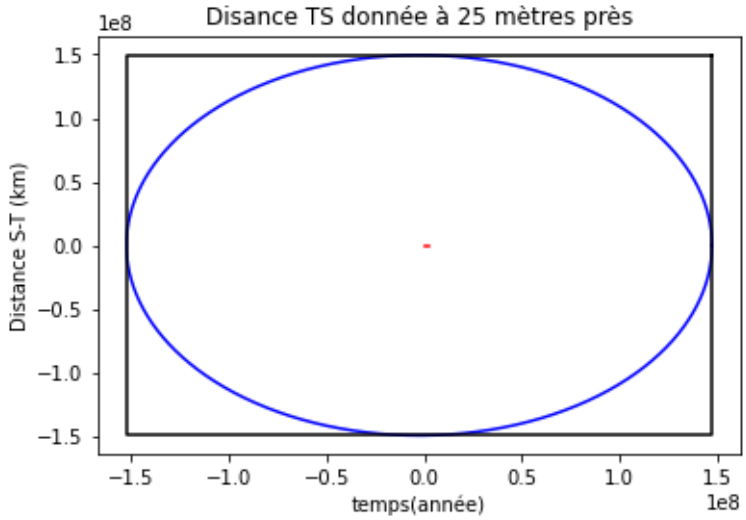
```
def theta(t):
    if t<=0.5:
        t=acos((149550457.4398135-TS(t))/(TS(t)*0.016719656757942607))
    else:
        t=2*pi-acos((149550457.4398135-TS(t))/(TS(t)*0.016719656757942607))
    return t
```

On reconnait TS(t), la fonction que nous avons établie précédemment pour la distance Terre-Soleil.

Finalement nous connaissons maintenant r et θ en fonctions du temps, et des relations très simple nous permettent de passer des coordonnées polaires aux coordonnées cartésiennes :

$$x(t) = r(t) \times \cos(\theta(t)) \quad et \quad y(t) = r(t) \times \sin(\theta(t))$$

Nous pouvons désormais facilement tracer la trajectoire de la Terre autour du Soleil, cette ellipse très proche d'un cercle si l'on regarde les graduations des deux axes. Mais maintenant nous connaissons les équations du temps se cachant derrière :



Nous avons encadré aux valeurs extrêmes, et vous pourrez remarquer en regardant de près, que ce n'est pas un point au centre mais un trait, reliant le centre de gravité de l'ellipse et l'un de ses deux foyers (position du Soleil), la

longueur de ce trait confirme que le foyer de l'ellipse est très proche du centre de gravité (ou centre de symétrie), et donc que l'ellipse est très proche d'un cercle (attention à la graduation des axes de la courbe ci-dessus). Nous aurions quand même perdu beaucoup de rigueur à considérer que cette trajectoire est un cercle, en effet il y a tout de même un écart de plus de cinq millions de kilomètres entre les maximum et minimum de distance Terre-Soleil, ce qui n'est pas négligeable.