Projet de fin d'année CIR 1

professeur@isen-yncrea.ouest.fr

22 juin 2022

1 Introduction

Nous proposons dans ce projet de calculer et d'afficher la trajectoire d'astres dans le système solaire. Le calcul sera effectué par la résolution approchée d'équations différentielles à l'aide de la méthode d'Euler. Votre objectif est à la fois de vous approprier les notions scientifiques abordées dans ce sujet mais aussi de fournir une implémentation du système.

2 Un peu de physique

L'objectif principal de ce projet est de calculer la trajectoire des astres du système solaire. Ces trajectoires seront modélisées par le couple suivant : la position de l'astre $\overrightarrow{r(t)}$ et sa vitesse $\overrightarrow{v(t)}$ en fonction du temps t, comme indiqué sur la figure 1. Le système est composé de l'ensemble des astres (soleil compris). On se placera dans un référentiel héliocentrique en trois dimensions.

Pour commencer le projet, il sera possible de considérer le système Terre-Soleil. Cependant, toutes les explications seront données pour l'ensemble du système. On considère pour le début du projet que les trajectoires sont toutes dans le même plan. **Gardez néanmoins la troisième dimension dans le projet quitte à la laisser à 0.**

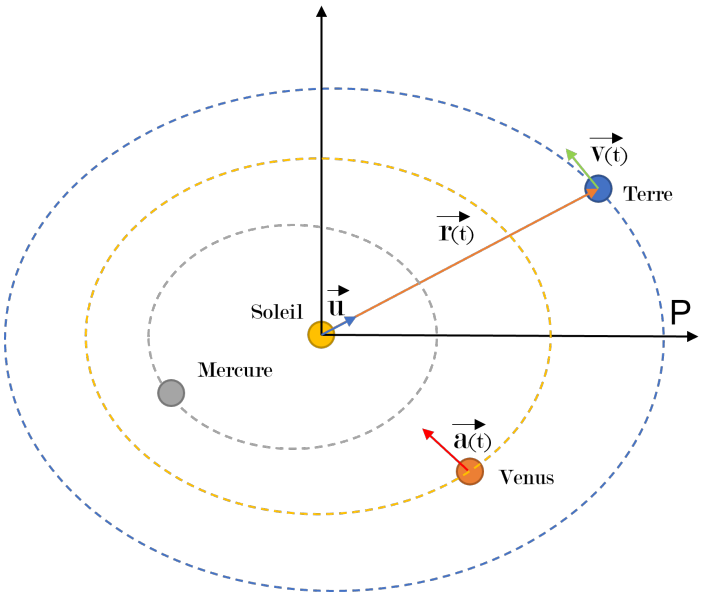

FIGURE 1 – Une partie du système solaire.

2.1 Équation gravitationelle

Nous nous plaçons dans le cadre de plusieurs corps soumis à la gravitation. On considère concrètement l'ensemble des planètes qui gravitent autour du soleil et le soleil lui-même. Nous proposons

une méthode qui permet de calculer de manière approchée la trajectoire des différentes planètes du système solaire.

D'après le principe fondamental de la dynamique : dans un référentiel galiléen, l'accélération du centre d'inertie d'un système de masse m constante est proportionnelle à la résultante des forces qu'il subit, et inversement proportionnelle à m. Soit :

$$\sum_i \overrightarrow{F_i} = m.\overrightarrow{a} \tag{1}$$

où

— chaque $\overrightarrow{F_i}$ désigne les forces extérieures exercées sur l'objet ;

— m est sa masse inertielle ;

— $\overrightarrow{a}$ est l'accélération de son centre d'inertie ;

D'autre part, d'après la loi universelle de la gravitation (Isaac Newton) :

Deux corps ponctuels de masses respectives m_A et m_B s'attirent avec des forces vectoriellement opposées et de même valeur absolue.

$$F_{A/B} = F_{B/A} = \frac{G.m_A.m_B}{r^2} \tag{2}$$

avec m_A et m_B en kilogrammes (kg), r en mètres (m), $F_{B/A}$ et $F_{A/B}$ en newton (N). G est la constante gravitationnelle :

$$G = 6.67408 N.kg^{-2}.m^2 \tag{3}$$

On peut réécrire cette équation sous forme vectorielle en utilisant un vecteur unitaire qui sera dirigé vers le centre de l'objet qui attire le premier.

$$\overrightarrow{F_{A/B}} = -\overrightarrow{F_{B/A}} = \frac{G.m_A.m_B}{r^2}\overrightarrow{u} \tag{4}$$

où $\overrightarrow{u}$ représente le vecteur unitaire (i.e. $||\overrightarrow{u}|| = 1$) dirigé de B vers A (voir figure 1).

Pour s'affranchir de ce vecteur unitaire on réintroduit le vecteur $\overrightarrow{r} = \overrightarrow{BA}$, en sachant que $\overrightarrow{r} = r.\overrightarrow{u}$ on obtient simplement :

$$\overrightarrow{F_{A/B}} = -\overrightarrow{F_{B/A}} = \frac{G.m_A.m_B}{r^3}\overrightarrow{r} \tag{5}$$

Notez que le dénominateur est maintenant en r^3 puisqu'on a multiplié le numérateur par r.

En combinant les équations (1) et (5) on obtient donc une équation différentielle du second degré pour l'élément A (ou l'astre A) attiré par les autres éléments I :

$$\sum_I \overrightarrow{F_{A/I}} = m_A.\overrightarrow{a} = -\sum_I \frac{G.m_A.m_I}{||AI||^3}\overrightarrow{AI} \tag{6}$$

L'accélération et la position sont des fonctions du temps. Comme l'accélération est la dérivée seconde de la position en fonction du temps, on obtient une équation dont les inconnues sont la position et ses dérivées. On remarquera que l'on peut aisément simplifier l'équation par m_A qui intervient dans les deux termes.

$$\overrightarrow{a} = -\sum_I \frac{G.m_I}{||AI||^3}\overrightarrow{AI} \tag{7}$$

Si on ne considère que deux astres, par exemple la terre (T) et le soleil (S), on peut simplifier l'équation (6) :

$$\overrightarrow{a} = -\frac{G.m_S}{||r||^3}\overrightarrow{r} \tag{8}$$

où m_S est la masse du soleil, et r la distance Terre Soleil à l'instant t.

L'accélération est la dérivée seconde de la position en fonction du temps. L'équation différentielle du second degré qui exprime la position d'un astre à chaque instant s'écrit donc :

$$\frac{\overrightarrow{d^2r}}{dt^2} = -\frac{G.m_S}{||r||^3}\overrightarrow{r} \tag{9}$$

2.2 Résolution approchée de l'équation différentielle

On sait déjà qu'il est possible de résoudre de manière exacte ce type d'équation différentielle. On obtiendra dans le cas de la terre une ellipse. La méthode de résolution exacte ne sera pas abordée dans ce projet. L'objectif de ce projet est d'apprendre à approcher la résolution de n'importe quelle équation différentielle avec la méthode d'Euler. L'intérêt est de pouvoir vérifier si votre résolution approchée est réussie, puisque tout le monde sait que la Terre tourne autour du Soleil en 365 jours et quart environ.

2.2.1 Principe de résolution d'une équation différentielle par la méthode d'Euler

En considérant que la vitesse est la dérivée de la position on a (pour l'axe des abscisses par exemple) :

$$v_{x_t} = \frac{dx}{dt} \tag{10}$$

On va chercher à établir une suite de positions $r_t = (x_t, y_t, z_t)$ et de vitesses $v_t = (v_{x_t}, v_{y_t}, v_{z_t})$ à partir des équations vues dans la section précédente. Si on considère un petit pas de temps Δt on approche la valeur de la position (en x par exemple) à l'instant $t+1$ avec l'équation suivante :

$$x_{t+1} = x_t + v_{x_t}.\Delta t \tag{11}$$

On pourra généraliser cette approche sur les trois dimensions dans le référentiel choisi. De la même façon, l'accélération nous permet de retrouver la nouvelle vitesse à l'instant $t+1$:

$$a_x = \frac{dv_x}{dt} \tag{12}$$

On approche donc la vitesse de cette façon :

$$v_{x_{t+1}} = v_{x_t} + a_{x_t}.\Delta t \tag{13}$$

De la même manière on généralisera sur les trois dimensions.

Pour finir il faut trouver un moyen de recalculer l'accélération sur les trois dimensions. pour cela, rien de plus simple : on réutilise l'équation (8).

Avec la méthode d'Euler, nous allons donc reconstituer à chaque pas de temps t la nouvelle position, la nouvelle vitesse et la nouvelle accélération de notre planète. Toutes les positions successives nous donneront une trajectoire que nous pourrons exporter puis afficher sur un graphique.

- on calcule les accélérations à partir des positions au temps t_n ;
- on calcule les positions au temps t_{n+1} à partir des vitesses aux temps t_n ;
- on calcule les vitesses au temps t_{n+1} à partir des accélérations précédemment calculées.

La méthode d'Euler donne une première approximation de la fonction de position d'une planète. Cependant, vous constaterez par vous-même que cette méthode ne donne pas une solution stable : la trajectoire diverge rapidement.

La précision de la résolution va dépendre notamment du pas de temps Δt choisi. L'analyse du choix de ce paramètre sera examiné lors du rendu. Vous devrez a minima implémenter cette méthode de résolution pour votre projet. Cette méthode constitue le minimum à comprendre pour le projet. Vous pouvez pour l'instant sauter les paragraphes suivants (dans un premier temps) et lire le paragraphe 2.3 et les suivants.

Nous allons voir maintenant deux méthodes qui permettent d'améliorer la résolution approchée de l'équation différentielle. La première est la méthode de Runge Kutta (prononcer Roune - g - Kouta). La deuxième est une résolution d'Euler dite asymétrique (elle est plus simple à comprendre). Ces deux méthodes sont des extensions du projet.

2.2.2 Résolution approchée par méthode de Runge Kutta.

Plutôt que de calculer un nouveau point après chaque pas de temps Δt on va subdiviser les calculs en plusieurs étapes pour affiner la résolution. La méthode du Runge Kutta peut s'appliquer avec plus ou moins d'étapes. La méthode de Runge Kutta d'ordre 1 que l'on nommera RK1 consiste à se projeter en avant d'un seul pas Δt pour calculer un coefficient k_1 que l'on applique pour trouver le point suivant.

Avant de poursuivre, il faut noter que dans notre cas, l'équation différentielle est du deuxième ordre. Nous pouvons transformer notre équation différentielle du second degré en un système de deux équations à deux inconnues qui seront la position et la vitesse. En reprenant les équations (10) et (9) on obtient le système d'équations suivant :

$$\begin{cases} \dfrac{d\overrightarrow{r}}{dt} = \overrightarrow{v} \\ \dfrac{d\overrightarrow{v}}{dt} = \overrightarrow{a} = \dfrac{-G.m_S}{||r||^3}\overrightarrow{r} \end{cases} \tag{14}$$

Où $\overrightarrow{r}$ est le vecteur position de l'astre et $\overrightarrow{v}$ son vecteur vitesse. On transforme seulement la notation sous forme vectorielle : le premier élément est la position, le deuxième la vitesse. On obtient alors :

$$\frac{d}{dt}\begin{pmatrix}\overrightarrow{r}\\ \overrightarrow{v}\end{pmatrix} = \begin{pmatrix}\overrightarrow{v}\\ \dfrac{-G.m_S}{||r||^3}\overrightarrow{r}\end{pmatrix} = f(t, p(t)) \tag{15}$$

On peut modéliser notre problème avec :

- un vecteur $p(t) = \begin{pmatrix}\overrightarrow{r}\\ \overrightarrow{v}\end{pmatrix}$
- une fonction $f(t, p(t))$ telle que :

$$f : \left| \begin{array}{ccc} \mathbb{R}^3 \times \mathbb{R}^3 & \longrightarrow & \mathbb{R}^3 \times \mathbb{R}^3 \\ p(t) & \longmapsto & \begin{pmatrix} 0 & 1 \\ \dfrac{-G.m_S}{||r||^3} & 0 \end{pmatrix}.p(t) \end{array} \right.$$

Dans ce qui suit, p(t) sera appelé un "point" qui sera donc constitué d'une position, d'une vitesse et d'une date.

La méthode de Runge Kutta d'ordre 1 consiste à calculer un coefficient k_1 à l'aide de la fonction f de la manière suivante :

$$k_1 = \Delta t.f(t_n, p_n) \tag{16}$$

Où p_n est le vecteur point (position, vitesse) à l'instant t_n. En pratique, la méthode de Runge Kutta d'ordre 1 revient à appliquer la méthode d'Euler.

La méthode de Runge Kutta d'ordre 2 consiste à séparer l'intervalle de temps en deux pour calculer la dérivée en deux temps. la première étape est la même que celle d'ordre 1.

On peut ensuite reprendre le calcul au milieu de l'intervalle et calculer un deuxième coefficient k_2 qui va venir pondérer la dérivée obtenue avec le seul coefficient k_1.

$$k_2 = \Delta t.f(t_n + \frac{\Delta t}{2}, p_n + \frac{k_1}{2}) \tag{17}$$

On observe sur la figure 2 que le point où on applique f est au centre du rectangle délimité par $[t_n, t_{n+1}]$ et $[p_n, p_n + k_1]$. On a aussi $\Delta t = t_{n+1} - t_n$. Le coefficient k_2 est en définitive la dérivée au point milieu de l'intervalle délimité par Δt

Une fois les coefficients calculés, il suffit d'ajouter $k1$ et k_2 à p_n avec les bonnes pondérations pour obtenir p_{n+1}.

$$p_{n+1} = p_n + k_2 \tag{18}$$

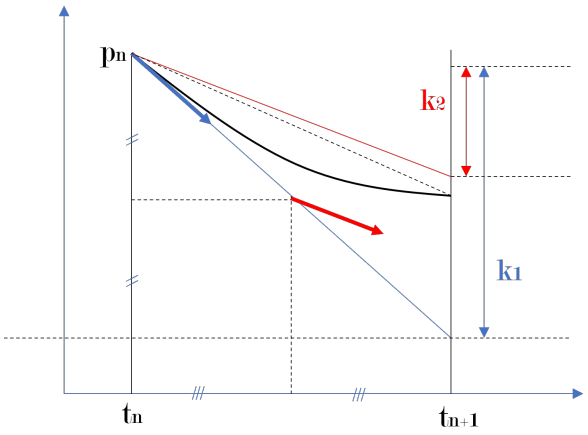

FIGURE 2 – Résolution d'une équation différentielle par méthode de Runge Kutta d'ordre 2. Source :[2].

On observe sur la figure 2 que le point calculé en $p_n + k_2$ est plus proche de la trajectoire réelle que le point qu'on aurait calculé en $p_n + k_2$ avec la méthode de Runge Kutta d'ordre 1.

Conseils d'implémentation :

Comme nous avons un système d'équations, on peut séparer les coefficients k_1 et k_2 en deux parties $k_{1,r}$ et $k_{1,v}$ puis $k_{2,r}$ et $k_{2,v}$. Chacun de ces coefficients est un vecteur sur trois dimensions.

$$\overrightarrow{k}_{1,r} = \Delta t \times \overrightarrow{v_n} \tag{19}$$

$$\overrightarrow{k}_{1,v} = -\Delta t \times \frac{G.m_S}{||r||^3} \overrightarrow{r}_n \tag{20}$$

$$\overrightarrow{k}_{2,r} = \Delta t \times (\overrightarrow{v}_n + \frac{\overrightarrow{k}_{1,v}}{2}) \tag{21}$$

$$\overrightarrow{k}_{2,v} = -\Delta t \times \frac{G.m_S}{||\overrightarrow{r}_n + \frac{\overrightarrow{k}_{1,r}}{2}||^3} (\overrightarrow{r}_n + \frac{\overrightarrow{k}_{1,r}}{2}) \tag{22}$$

$$\overrightarrow{r}_{n+1} = \overrightarrow{r}_n + \overrightarrow{k}_{2,r} \tag{23}$$

$$\overrightarrow{v}_{n+1} = \overrightarrow{v}_n + \overrightarrow{k}_{2,v} \tag{24}$$

Pour aller plus loin : on remarquera que la méthode de Runge Kutta d'ordre 2 donne de bons résultats pour une première approximation (si le pas de temps est correctement choisi). Il existe néanmoins une méthode de Runge Kutta d'ordre 4 que certains pourront tester.

2.2.3 Résolution approchée par méthode d'Euler asymétrique.

La méthode de résolution par la méthode d'Euler asymétrique est plus simple à mettre en place que la méthode de Runge Kutta. Il suffit d'inverser les étapes de résolution de la manière suivante [1] :

— on calcule les positions au temps t_{n+1} à partir des vitesses au temps t_n ;

— on calcule les accélérations à partir de ces nouvelles positions ;

— on calcule les vitesses au temps t_{n+1} à partir de ces accélérations.

Il est vivement conseillé d'implémenter cette méthode ; contrairement à la méthode d'Euler, la divergence est faible.

2.3 Conservation de l'énergie

Nous allons vérifier dans ce projet si l'énergie de notre système se conserve. Cette vérification nous permettra de juger si les estimations faites par les méthodes précédentes sont correctes. Il y a deux énergies à calculer : l'énergie potentielle et l'énergie cinétique. On calculera ces deux valeurs à chaque pas de temps pour vérifier que leur somme est constante.

L'énergie potentielle se calcule de la manière suivante :

$$E_p = \frac{1}{2} \sum_i \frac{G.m_i.m_j}{||r||^2} \tag{25}$$

Dans le cas du système Terre - Soleil on aura donc :

$$E_p = \frac{G.m_T.m_S}{||r||^2} \tag{26}$$

L'énergie cinétique se calcule de la manière suivante :

$$E_c = \frac{1}{2} \sum_i m_i ||v||^2 \tag{27}$$

l'énergie totale du système est $E = E_p + E_c$. Elle doit être constante au fil du temps.

3 Modélisation et implémentation

L'implémentation du projet se divise en deux parties : le calcul des trajectoires des éléments du système d'une part et l'affichage de celles-ci dans une interface graphique d'autre part. Pour bien séparer les deux, un format de fichier d'échange est prévu pour exporter les trajectoires calculées d'une part et importer celles-ci dans une interface graphique d'autre part (voir la figure 3). Le format d'échange est fixé pour tous les élèves et ne peut pas être modifié.

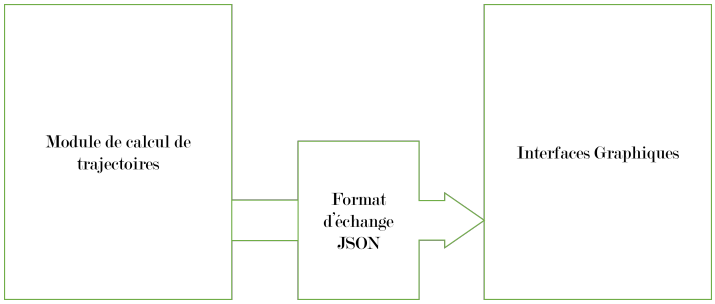

FIGURE 3 – Architecture globale.

Ce format d'échange unique a un double intérêt
— le développement de l'interface graphique ne bloque pas celui du calcul des trajectoires ;
— les données issues d'un projet peuvent être utilisées pour les réinsérer dans un autre.
Un fichier basique contenant quelques positions de trajectoires sera fourni pour aider au démarrage du projet.

3.1 Base commune pour l'implémentation du projet minimal

Certains aspects du projet sont imposés et ne doivent pas être modifiés. C'est le cas du format d'échange qui va permettre :
— d'enregistrer les points des trajectoires que vous aurez calculées dans un fichier d'une part ;
— de lire un fichier de trajectoires dans ce même format pour afficher les trajectoires ou faire des calculs de vérification sur celles-ci d'autre part.
Ainsi, vous recevrez en début de projet un fichier d'exemple contenant une ou plusieurs trajectoires dont vous pourrez vous servir comme point de départ.

3.1.1 Format d'échange

Le format d'échange est fixe et doit rester le même pour tous les étudiants. Si une de vos extensions vous oblige à modifier ce format : essayez dans un premier temps de respecter le format d'échange. Adressez-vous à votre professeur pour proposer une modification avec une justification très claire. Dans

le cas peu probable où votre professeur autorise votre groupe à modifier ce format, notez bien les raisons qui vous ont poussés à le faire. Pensez aussi à pouvoir utiliser le format de base en parallèle à votre format.

Le format d'échange est en JSON. Il est constitué d'une liste de trajectoires d'astres : les données débutent par une accolade et finissent par une accolade. Chaque astre est séparé par une virgule. Une trajectoire commence par le nom de l'astre en guillemets et est suivi du caractère " :". S'ensuit une liste qui commence par un "[" et se termine par "]". Cette liste est composée de vecteurs [position, vitesse, temps] qui composent la trajectoire. Chaque position et chaque vitesse est un triplet [x,y,z] d'éléments flottants (double en C) écrits sous forme scientifique : [1.087309e+11, 1.000176e+11, 0.000000e+00] (format "%e" dans le printf). La troisième dimension doit être conservée même si elle n'est pas utilisée. Le temps est un entier. Les unités utilisées sont les unités du système international : mètres, kilogrammes, secondes, newtons. Il est possible d'accoler une autre chaine de caractères au nom d'une planète si besoin : par exemple, "earth-euler" ou "earth-rk2" pour montrer que vous avez calculé votre trajectoire avec telle ou telle méthode. Voici un exemple contenu dans un fichier trajectoire :

```
{"earth-euler" : [
[[1.470000e+11, 0.000000e+00, 0.000e+00],[0.000000e+00, 3.028629e+04, 0.000e+00], 0],
[[1.470000e+11, 2.616735e+08, 0.000e+00],[-5.306168e+01, 3.028629e+04, 0.000e+00], 1],
[[1.469995e+11, 5.233471e+08, 0.000e+00],[-1.061231e+02, 3.028619e+04, 0.000e+00], 2],
[[1.469986e+11, 7.850198e+08, 0.000e+00],[-1.591841e+02, 3.028601e+04, 0.000e+00], 3],
[[1.469972e+11, 1.046691e+09, 0.000e+00],[-2.122445e+02, 3.028572e+04, 0.000e+00], 4]
]
,
"earth-RK2" : [
[[1.470000e+11, 0.000000e+00, 0.000e+00],[0.000000e+00, 3.028629e+04, 0.000e+00], 0],
[[1.469998e+11, 2.616735e+08, 0.000e+00],[-5.306162e+01, 3.028624e+04, 0.000e+00], 1],
[[1.469991e+11, 5.233463e+08, 0.000e+00],[-1.061231e+02, 3.028610e+04, 0.000e+00], 2]
]
}
```

Dans ce fichier, il y a deux trajectoires. Les deux concernent la planète Terre. Une trajectoire est calculée avec la méthode d'Euler, la deuxième avec Runge Kutta d'ordre 2. La première trajectoire contient 5 points à des temps notés 0, 1, 2, 3, 4 (l'unité ici est un dixième de jour). Chaque point contient sa position $[x, y, z]$ suivi de sa vitesse $[v_x, v_y, v_z]$.

Les nombres seront sous forme scientifique : 7 chiffres significatifs suivis d'un exposant à deux chiffres. Il n'y aura pas forcément de retour à la ligne dans les fichiers finaux, ils sont ici pour vous aider à la lecture. *Nous avons volontairement enlevé trois zéros dans l'exemple pour que les lignes passent dans le document.*

3.2 Guide pour l'implémentation en C

Pour écrire dans un fichier d'échange, deux options sont envisageables : utiliser une méthode traditionnelle pour ouvrir un fichier en écriture, rajouter proprement chaque élément dans le fichier, fermer le fichier. Vous pouvez aussi tout écrire dans la console et rediriger le résultat vers le fichier d'échange en console linux avec le symbole de redirection ">".

3.2.1 Structures de données

Dans la section qui suit, nous proposons une trame de départ pour organiser les données à traiter dans le projet. Nous conseillons de commencer par traiter le projet de cette façon puis vous pourrez vous adapter par la suite. L'organisation et la séparation des fichiers sera un critère d'évaluation.

3.2.2 Structure vecteur

Nous travaillons en 3 dimensions avec des vecteurs, Vous devez donc créer une struct vector qui contient 3 double x, y et z. Ces vecteurs serviront à la fois pour la position, la vitesse mais aussi l'accélération.

3.2.3 Opérations

Conjointement à cette structure, il est demandé de créer des fonctions mathématiques : addition et soustraction de deux vecteurs, multiplication d'un vecteur par un scalaire, calcul de la norme d'un vecteur.

Il est **obligatoire** de faire des tests. Nous proposons de créer, dans chaque module .c/.h, une fonction void moduleTest() (void vectorTest() dans notre cas) dans laquelle vous créez des vecteurs, vous les additionnez, les multipliez par un scalaire. Cette fonction doit afficher les résultats en console. Ces fonctions seront appelées dans la fonction main puis commentées une fois les tests validés.

3.2.4 Structure point

Un point est élément de la trajectoire, il regroupe la position notée r et la vitesse notée v. Ces deux éléments sont des vecteurs de type vector. Un point inclut également le temps. Les points sont notés p dans le sujet.

3.2.5 Structure trajectoire

Nous proposons de coder la trajectoire comme une file de points : le premier point calculé sera le premier affiché.

3.2.6 Structure planète

Une planète comprend un nom, une masse, sa trajectoire et sa périhélie. La périhélie est la distance maximale de la planète au soleil, à ne pas confondre avec le demi grand axe de l'ellipse. Documentez-vous sur internet pour connaitre la définition ainsi que les valeurs précises des périhélies de chaque planète.

Cherchez les différentes constantes pour les planètes à mettre dans le système.

Pour initialiser sa planète, il faut : une position initiale (placez la à la périhélie, point P de la figure 1. La vitesse est toujours perpendiculaire au rayon r. Pour calculer la vitesse initiale on utilisera la formule donnée sur la page Périhélie de Wikipédia. Vous pouvez aussi trouver la valeur calculée en vous documentant.

Créez une fonction init qui remplira les informations du premier point de la trajectoire de la planète avec toutes ces données.

3.2.7 Equation différentielle et méthode d'Euler

La méthode d'Euler consiste à calculer les informations du point suivant (position, vitesse, temps) en fonction du point courant. Il suffit d'appliquer les équations et la méthode vues en début de document.

3.3 Réalisation du projet

3.3.1 Projet Minimal

Nous détaillons ici ce qui est attendu au minimum d'un groupe d'étudiants. Ce minimum ne permet pas d'obtenir la moyenne mais au moins de s'en rapprocher. Il faudra au minimum être capable de calculer la trajectoire de la terre autour du soleil en utilisant la méthode d'Euler simple. La trajectoire sera exportée automatiquement dans le format d'échange unique vers un fichier. L'organisation du code en fichiers séparés, le choix des structures, des fonctions ainsi que la présence de commentaires dans le code pourront faire l'objet d'une évaluation.

Les étudiants devront également être capables d'expliquer à l'oral le principe général de la résolution d'équations différentielles par la méthode d'Euler.

Chaque groupe devra vérifier si la méthode choisie pour résoudre les équations différentielles respecte le principe de conservation de l'énergie dans le temps. Cette vérification sera effectuée au moyen de calculs précis dans le langage de programmation imposé.

Un travail minimal de recherche sur la valeur de certaines constantes physiques est demandé (la masse de la terre par exemple). Il est également conseillé, pour la culture générale de se renseigner sur les différents scientifiques qui ont permis au fil des siècles de mettre au point ce type de méthodes (Euler, Runge et Kutta, Copernic, Galilée, Kepler, Newton...).

Sauvegardez des versions fonctionnelles de votre projet au fur et à mesure dans des dossiers séparés. Mieux vaut avoir une démo incomplète de la veille qui fonctionne plutôt que la dernière version du jour qui ne compile même pas...

3.3.2 Projet Attendu

Puisque le projet minimal ne suffit pas à lui seul à obtenir la moyenne, il vous faudra proposer des extensions à celui-ci. Il est conseillé de réaliser deux à trois extensions. Pour chaque extension, il faudra :

— prévoir un cahier des charges avec votre groupe : résultats attendus, temps passé sur l'extension ;
— réaliser cette extension, la tester dans plusieurs cas de fonctionnement ;
— en fin de projet, faire une analyse de la différence entre le résultat attendu et les résultats obtenus.

Le but n'est donc pas d'avoir des milliers d'idées et de les laisser en suspens à la fin du projet mais plutôt de montrer que votre groupe a été capable de s'organiser, de prévoir et d'analyser ses résultats. Les extensions n'ont pas besoin d'être originales ou extravagantes, il faut plutôt choisir des extensions raisonnables et réalisables.

Vous pouvez proposer vos propres extensions. N'hésitez pas à vous diriger vers vos professeurs pour avoir leur avis avant de vous lancer.

Voici une liste non-exhaustive d'extensions possibles :

— réaliser une interface graphique pour afficher les trajectoires en Javascript (*très fortement recommandée*) ;
— réaliser une interface graphique pour afficher les trajectoires en SDL (C) (en 2D, en 3D !) ;
— ajouter plusieurs planètes au système et modéliser toutes les interactions du système (La Terre et Mars s'attirent entre-elles) ;
— modéliser des trajectoires plus réalistes : ajout d'une troisième dimension, amélioration de l'orientation des ellipses ;
— améliorer le système de résolution des équations différentielles (*fortement conseillée*) par une méthode de Runge Kutta et/ou la méthode d'Euler asymétrique : comparer les résultats obtenus.

Les idées suivantes sont plus compliquées à mettre en œuvre, à la limite du sujet :

— rechercher et lire des données de trajectoires de corps célestes trouvées sur des sites officiels (ESA, NASA) puis les importer pour les retranscrire dans le format d'échange du projet ;
— réussir à stabiliser la Lune autour de la Terre (challenge) ;
— modéliser la ceinture d'astéroïde ;
— prévoir une éclipse lunaire, un alignement de planète ;
— modéliser la trajectoire d'une comète, par exemple la comète de Halley (expliquer la forme de la trajectoire) ;
— mettre un satellite en orbite terrestre (attention, on sort presque du cadre du projet, c'est difficile) ;
— modéliser un autre système que le système solaire à partir d'une situation aléatoire ;
— créer un système de résolution d'équations différentielles générique et modéliser un autre système (comme un pendule ou un lancer de ballon de basket).

3.3.3 Citer ses sources

Vous allez très certainement avoir recours à des informations externes : il est important de noter au fur et à mesure les sources de vos informations : URL, ouvrages bibliographiques. Il vous sera demandé de vous justifier sur la source de vos informations. La copie simple de code sur un autre groupe ou sur le web est totalement interdite et sera sanctionnée. Cependant, il est tolérable de demander de l'aide. Tout ce qui a été réalisé dans le projet doit pouvoir être expliqué par les étudiants du groupe. Une solution copiée et non comprise sera sanctionnée (possiblement par un zéro). Il est possible de tester le comportement d'un fichier d'échange d'un autre groupe "pour voir si ça marche". Dans ce cas il faudra citer ses sources. Ce test ne donnera pas lieu à des points supplémentaires et ne sera pas sanctionné.

3.3.4 Analyse des résultats et des méthodes

Il faudra être capable en fin de projet de fournir une analyse scientifique de ses résultats. L'objectif n'est pas forcément de trouver les résultats exacts, mais de savoir expliquer comment ils ont été trouvés.

Dire que la Terre a une période de révolution de 365 jours et quart sans justification sera probablement sanctionné, montrer que votre algorithme vous donne une période de 370 jours en sachant expliquer d'où proviennent les erreurs sera bienvenu. Il est normal que les trajectoires divergent avec la méthode d'Euler. Il faut cependant pouvoir expliquer le phénomène.

4 Visualisation des données (extension)

La partie visualisation des données est comptée comme une extension possible de votre travail. Vous pouvez faire le choix de passer par la technologie que vous préférez, mais une solution vous est proposée ici pour vous aider.

4.1 Représentation des trajectoire sur une page web

Nous vous proposons de réaliser la présentation de votre travail sur une page web qui représentera les orbites des planètes que vous avez obtenues par calcul.

Nous vous proposons d'utiliser pour ce faire la librairie graphique **Processing** et plus particulièrement son implémentation en javascript présente à l'adresse suivante :

https://p5js.org

Nous vous fournissons un fichier contenant les données d'une planète (Mercure) pour vous permettre de commencer à tracer ces données si vous souhaitez vous répartir le travail au sein de votre groupe.

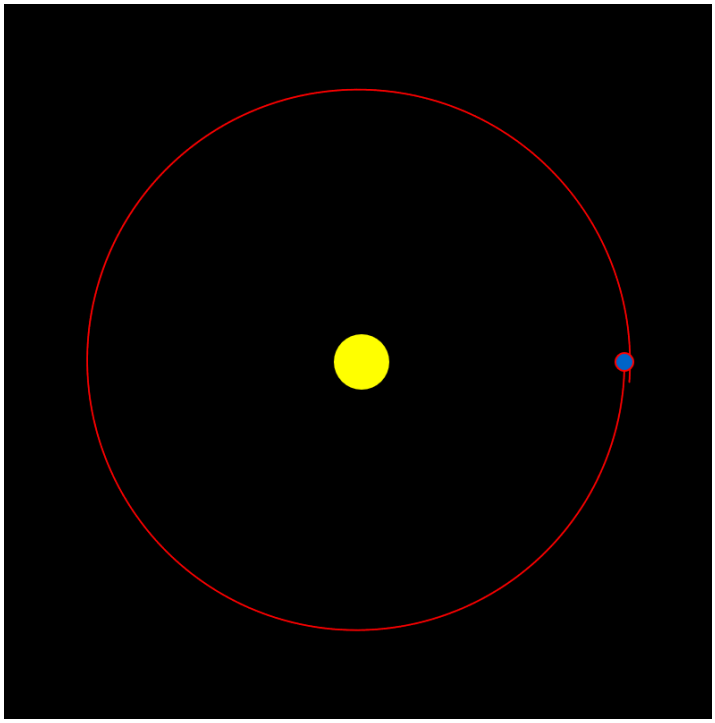

FIGURE 4 – Orbite de La Terre

La figure 4 montre une simple vue du Soleil et de la Terre ainsi que son orbite (les astres n'étant pas à l'échelle).

Ceci est bien entendu un exemple de représentation... A vous de choisir vos objectifs.

4.2 Objectifs de la partie graphique

Les objectifs sont libres (et comme dit plus haut optionnels) : à vous de décider ce que vous allez afficher. Plusieurs options sont possibles :

— les ellipses représentant les trajectoires des planètes

— les animations des mouvements des planètes

— l'affichage des vecteurs que vous avez calculés
— ...

Vous pouvez également penser à utiliser la page web pour autre chose que la présentation des orbites et présenter le travail qui vous est demandé. La présentation du projet sur une page HTML/CSS de bonne facture peut aussi vous apporter des points supplémentaires (ne prenez pas trop de temps pour ça).

Vous pouvez enfin enrichir par une interface pour paramétrer les informations que vous souhaitez afficher à l'écran. (affichage des orbites, des planètes, des vecteurs vitesses)...

NB : pour le chargement des données, l'idéal est de demander à l'utilisateur de charger les données depuis le fichier que vous aurez généré grâce au code C.

4.3 Documentation de p5js

p5.js est une librairie graphique (et multimédia) : elle va vous permettre d'ajouter des graphiques, des animations ou encore des sons à votre page web.

Tout cela se passe au sein d'un canvas dans lequel vous allez pouvoir charger un ensemble d'informations au chargement de la page, par le biais d'une fonction setup(). Vous allez pouvoir dessiner et animer votre canvas à l'aide d'une fonction draw().

Voici un exemple permettant l'affichage d'une ellipse dans un canvas et utilisant ces deux fonctions :

```
let x = 0;

function setup() {
  //le canvas se voit attribuer une taille...
  createCanvas(640, 360);
  //... et un fond gris foncé
  background(100, 100, 100);
}

function draw() {
  ellipse(x, height/2, 20, 20); //paramètres : position (x,y) et rayons (hauteur, largeur)
  x = x + 1;
}
```

À noter que la fonction draw est exécutée de manière continue : le dessin de l'ellipse sera mis à jour avec la nouvelle valeur x à chaque nouvelle exécution de la fonction.

Vous trouverez la documentation à la page suivante : https://p5js.org/reference/

Cette partie présente rapidement les fonctions pricipales de p5.js qui peuvent vous servir pour ce projet. Regardez dans la documentation pour plus de précisions et de nombreux exemples sur l'utilisation des différentes fonctions.

4.3.1 Fonction setup()

La fonction setup() permet d'initialiser le canvas (taille et fond notamment), ainsi que toute variable nécessaire à votre programme.

Elle ne doit être appelée qu'une seule fois.

4.3.2 Fonctions draw(), redraw(), loop() et noLoop()

Le principe de la fonction draw() est d'exécuter le code qu'elle contient en continu. Vous avez la possibilité de mieux contrôler ces étapes à l'aide de ces fonctions :

— la fonction noLoop() permet d'interrompre l'appel en continu de la fonction draw() (par exemple pour stopper une animation)
— la fonction loop() permet de relancer cet appel en continu
— la fonction redraw() permet d'appeler une fois la fonction draw(), par exemple lorsqu'on reçoit un évènement ou lorsque l'utilisateur clique sur un bouton.

4.3.3 La gestion d'évènements

Un ensemble de fonctions et de variables existantes permettent de manipuler les périphériques. Ces fonctions peuvent être redéfinies dans votre code pour appeler l'action correspondante. Les fonctions que vous pouvez utiliser sont par exemple :

— keyPressed(), appelée quand une touche du clavier est pressée. Vous pouvez découper plus finement le comportement en détectant la touche appuyée grâce à la fonction keyIsDown() ;
— mouseClicked(), qui lancera le code associé quand on clique sur la souris. Il existe des fonctions pour l'ensemble des possibilités d'interactions avec la souris (doubleClicked(), mouseDragged(), mouseWheel ou encore mouseMoved()) ;
— d'autres liées à l'utilisation d'un écran tactile ou d'un autre périphérique.

Les variables contiennent un ensemble d'informations qui sont renseignées dès qu'une action a lieu :

— vous pouvez récupérer les positions de la souris entre chaque frame (chaque récupération d'évènement) : movedX, movedY ;
— vous pouvez récupérer le nom des touches appuyées dans la variable key et keyCode pour les caractères non affichables (flèches, entrée, backspace...)

Voici un exemple de code issu de la documentation pour la gestion de l'appui sur deux touches du clavier :

```
let value = 0;
function draw() {
  fill(value);
  rect(25, 25, 50, 50);
}
function keyPressed() {
  if (keyCode === LEFT_ARROW) {
    value = 255;
  } else if (keyCode === RIGHT_ARROW) {
    value = 0;
  }
}
```

Ici, ce code permet lorsqu'on appuie sur la touche de droite ou de gauche de changer la valeur de la couleur du fond d'un rectangle.

4.3.4 Autres fonctionnalités

A vous de regarder un peu plus loin, mais p5.js vous permet également de jouer avec le DOM, de faire de la 3D ou de charger des données formattées en JSON.

Références

[1] L. Garcin. Méthode d'euler et gravitation.
https://lgarcin.github.io/2016-09-26-gravitation/.

[2] D. Lavabre and V. Pimienta. Cinétique chimique, intégration numérique.
http://cinet.chim.pagesperso-orange.fr/ex_sa/int_num.html.