# Projet 8 :

## Simulation numérique de la trajectoire des sondes Voyager

Cavaillès Gaëtan
Delbeque Antoine
Guiblin Maxime
Shih Francis

Tuteur du projet : Mr Nuttin Alexis

# Sommaire:

# Introduction

Le projet sonde voyager a pour but de simuler la trajectoire d'une sonde partant de la Terre en direction d'Uranus. La simulation prend en compte l'attraction des principaux astres présents dans le système solaire.

Ce projet a été lancé par la NASA dans les années 75'. La mission de la sonde voyager est de prendre des photos des différentes planètes les plus éloignées du système solaire. Pour cela, le temps de trajet calculé est de 16 ans. En faisant des simulations du trajet, les ingénieurs de la NASA se sont rendus compte que ce temps de trajet pouvait être réduit en profitant de l'effet fronde gravitationnelle de certaines planètes. Ce phénomène permet d'accélérer la sonde grâce à la proximité d'un astre en mouvement de masse importante. Le but de la mission étant de récupérer des images des planètes, la condition de proximité est alors primordiale.

Nous avons donc réalisé un programme informatique simulant ce trajet. Il permet de mettre en évidence l'influence de l'effet de fronde. En effet, la simulation étant facilement malléable, on peut rapidement remarquer que l'effet de fronde est quelque chose de très précis. C'est donc pour cela que deux sondes ont été envoyées, la précision des calculs ne pouvait prévoir exactement la bonne date et comme la position idéale des planètes ne se reproduit que tous les 176 ans, il fallait assurer la mission.

Dans toute la suite, la procédure pour réaliser une telle simulation sera expliquée. Nous calculons d'abord la trajectoire grâce à un programme informatique, puis nous affichons les résultats de ces calculs via une interface graphique.

# I. Méthodes utilisées

## A. Simulation informatique

Afin de réaliser notre projet de simulation numérique, nous avons décidé d'utiliser le langage de programmation C++ afin de pouvoir bénéficier d'un langage d'assez bas niveau, pour mieux gérer la mémoire, et profiter des apports de la programmation orientée objet. En effet, ce langage nous a permis de structurer notre programme de façon efficace, ce qui donne un code sur lequel travailler à plusieurs est beaucoup plus simple. La programmation orientée objet est particulièrement adaptée à notre projet puisque dans notre simulation numérique, nous avons plusieurs objets ayant des interactions entre eux : le soleil, la sonde et les différentes planètes du système solaire. Chaque objet comporte des *attributs* (variables internes) et des *méthodes* (fonctions internes).

Nous avons défini trois classes :

- Sonde : les attributs sont la position, la vitesse, le temps depuis le début de la simulation, la liste des distances entre la sonde et les différents astres, et un pointeur vers la liste des astres. On implémente dans cette classe une méthode permettant la mise à jour de la position et de la vitesse de la sonde à l'instant t.
- Astre : les attributs sont la position, la vitesse et la masse de l'astre. On implémente dans cette classe une méthode permettant la mise à jour de la position et de la vitesse de l'astre à l'instant t.
- Matrice : les attributs sont la taille de la matrice et les valeurs des différents coefficients. On implémente dans cette classe les opérations élémentaires d'addition, de soustraction, de multiplication et d'addition. La définition de cette classe n'était pas essentielle pour mener à bien notre simulation numérique, mais nous avons souhaité tirer parti de la programmation orientée objet un maximum, et avoir un code agréable à lire et à déboguer.

Une fois la structure de notre programme clairement définie, nous avons mis en équation la force d'attraction gravitationnelle sous forme matricielle afin de calculer la force d'attraction totale issue des différents astres et subie par la sonde à un instant $t$, et implémenté la méthode d'Euler afin de déterminer les nouvelles position et vitesse de l'astre à l'instant $t+h$, $h$ étant un pas relativement petit. Nous nous sommes restreints à un système solaire bidimensionnel, rendant plus facile la représentation graphique de la simulation. On note $x_s, y_s$ les positions en abscisse et en ordonnée de la sonde, et $x_i, y_i$ celles de l'astre $i$. On met sous équation ci-dessous l'interaction gravitationnelle entre la sonde et un astre quelconque de masse $m_{astre}$, la distance les séparant étant $r_{i,s}$.

$$X_s(t) = \begin{pmatrix} x_s(t) \\ \dot{x}_s(t) \end{pmatrix}$$

$$Y_s(t) = \begin{pmatrix} y_s(t) \\ \dot{y}_s(t) \end{pmatrix}$$

$$\dot{X}_s(t) = \begin{pmatrix} \dot{x}_s(t) \\ \ddot{x}_s(t) \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ \frac{-G*m_i}{r_{i,s}^3} & 0 \end{pmatrix} \begin{pmatrix} x_s(t) - x_i(t) \\ \dot{x}_s(t) - \dot{x}_i(t) \end{pmatrix}$$

$$\dot{Y}_s(t) = \begin{pmatrix} \dot{y}_s(t) \\ \ddot{y}_s(t) \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ \frac{-G*m_i}{r_{i,s}^3} & 0 \end{pmatrix} \begin{pmatrix} y_s(t) - y_i(t) \\ \dot{y}_s(t) - \dot{y}_i(t) \end{pmatrix}$$

On généralise cette formule pour $n$ astres :

$$\dot{X}_s(t) = \left( \sum_{i=1}^{n} \begin{pmatrix} 0 & 0 \\ \frac{G*m_i}{r_{i,s}^3} & 0 \end{pmatrix} \begin{pmatrix} x_i(t) - x_s(t) \\ \dot{x}_i(t) - \dot{x}_s(t) \end{pmatrix} \right) + \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x_s(t) \\ \dot{x}_s(t) \end{pmatrix}$$

$$\dot{Y}_s(t) = \left( \sum_{i=1}^{n} \begin{pmatrix} 0 & 0 \\ \frac{G*m_i}{r_{i,s}^3} & 0 \end{pmatrix} \begin{pmatrix} y_i(t) - y_s(t) \\ \dot{y}_i(t) - \dot{y}_s(t) \end{pmatrix} \right) + \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} y_s(t) \\ \dot{y}_s(t) \end{pmatrix}$$

On peut maintenant implémenter la méthode d'Euler à l'aide de la formule ci-dessus.

$$X_s(t+h) = X_s(t) + h\dot{X}_s(t) = \left(\sum_{i=1}^{n} C_i(X_i(t) - X_s(t))\right) + DX_s(t)$$

$$Y_s(t+h) = Y_s(t) + h\dot{Y}_s(t) = \left(\sum_{i=1}^{n} C_i(Y_i(t) - Y_s(t))\right) + DY_s(t)$$

Avec :

$$C_i = \begin{pmatrix} 0 & 0 \\ \frac{h*G*m_i}{r_{i,s}^3} & 0 \end{pmatrix}$$

$$D = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$$

```
31  void Sonde::Update(double h)
32  {
33      Matrice Xnew(2,1);
34      Matrice Ynew(2,1);
35      Matrice C(2,2);
36      Matrice* Xi;
37      Matrice* Yi;
38      Matrice D(2,2);
39      D.SetValue(0,0,1);
40      D.SetValue(0,1,h);
41      D.SetValue(1,1,1);
42
43
44          //On calcule les distances entre la sonde et chaque astre
45      for(int i=0; i<GetnbAstres(); i++)
46      {
47          astre[i].Update(t);
48          Xi = astre[i].GetX();
49          Yi = astre[i].GetY();
50
51          //Distance entre la sonde et l'astre i
52          (*d).SetValue(i,0,sqrt(pow((*Xi).GetValue(0,0)-X.GetValue(0,0),2) + pow((*Yi).GetValue(0,0)-Y.GetValue(0,0),2)));
53      }
54
55
56
57      for(int i=0; i<GetnbAstres(); i++)
58      {
59          //Ceci est l'influence de l'astre i sur la sonde avec l'approximation de la methode d'Euler
60          C.SetValue(1,0,h*G*(astre[i].Getmass())/(pow((*d).GetValue(i,0),3))); // (C = pas*G*M/(R^3) )
61          Xi = astre[i].GetX();
62          Yi = astre[i].GetY();
63
64                  //On l'ajoute à la position actuelle et à l'influence des autres astres
65          Xnew = Xnew + (C*(*Xi-X));
66          Ynew = Ynew + (C*(*Yi-Y));
67      }
68      Xnew = Xnew + D*X;
69      Ynew = Ynew + D*Y;
70      X=Xnew;
71      Y=Ynew;
72      t=t+h;
73  }
```

*Figure 1 : Implémentation de la méthode d'Euler en C++*

## B. Exploitation des résultats

Après avoir définit et écrit les classes nous avons dû ajouter les données de chaque planète dans la classe “astre”. Nous avons choisi d’exprimer toutes les valeurs en **u.a** (unité astronomique) dont les caractéristiques sont les suivantes :

- unité de temps : une année (365.25 jours terrestres)
- unité de distance : une unité astronomique (distance Terre-Soleil)
- unité de masse : masse de la Terre

Nous avons ensuite dû vérifier que les classes que nous venions d'implémenter fonctionnaient correctement. A ce stade le projet était encore en console et affichait les coordonnées de la sonde ; pour confronter ces résultats à la réalité nous avons programmé une **interface graphique** directement dans la **console** de Windows pour observer “en direct” le déplacement de notre sonde.

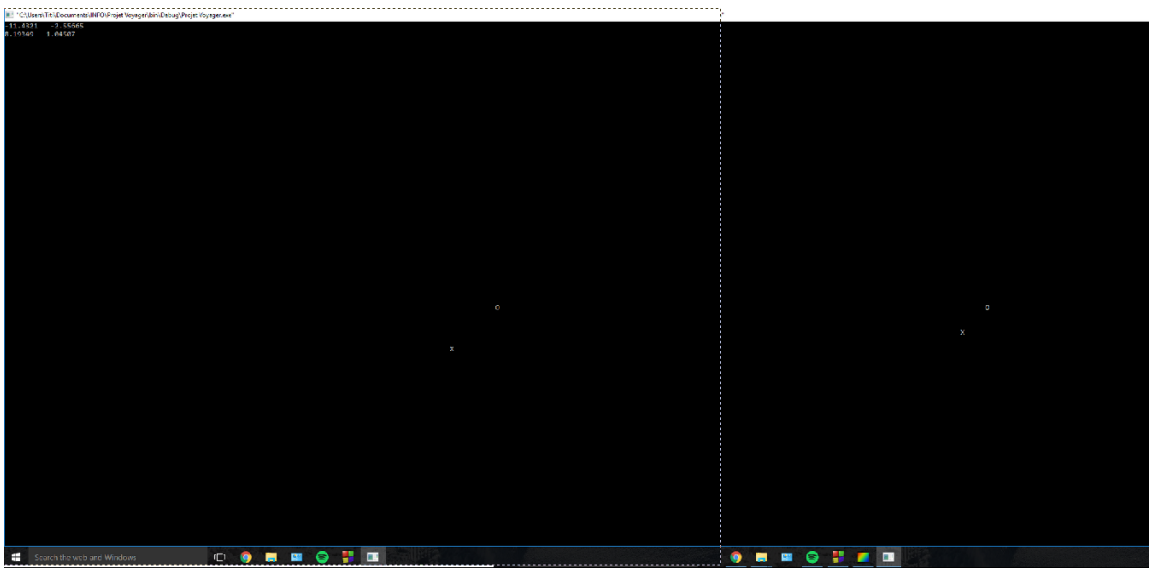


*Figure 2 : Déplacement de la sonde sur la console*

Les premiers essais consistaient à vérifier la gravitation en tentant de faire tourner la sonde autour du Soleil (sans implémenter d'autres astres). Nous avons pas eu de problèmes particuliers à cette étape. Ensuite nous avons ajouté les planètes et tenté d'observer d'éventuels signe d'assistance gravitationnelle en modifiant l'angle initial des planètes. En effet nous avons choisi de nous concentrer sur la mise en évidence de ce phénomène plutôt que sur une éventuelle reconstitution historique de trajet des sonde au vu des approximations que nous sommes obligés de faire.

Après s'être rendu compte de la difficulté de trouver ces effets de fronde nous avons automatisé le processus en utilisant une version de notre programme n'utilisant aucune interface graphique. Ce programme a pour but de minimiser la distance sonde-Jupiter (puis Saturne) en ajustant l'angle initial de la planète.

### C. Interface graphique

Le choix et l'écriture de l'interface graphique s'est fait par étape, en suivant plusieurs objectifs :

- afficher la sonde et les astres avec des images à leur position,
- pouvoir régler les paramètres du trajet directement via l'interface (sans passer par le code),
- agrandir l'image sur la sonde au passage d'une planète,
- afficher l'orbite des planètes,
- afficher le chemin parcouru par la sonde,
- approximer et afficher la trajectoire future de la sonde.

Les capacités de la console étant limitées nous avons d'abord choisi d'utiliser la *SDL* pour avoir un meilleur rendu. En procédant par étapes nous avons pu réaliser peu à peu la plupart des objectifs souhaités.

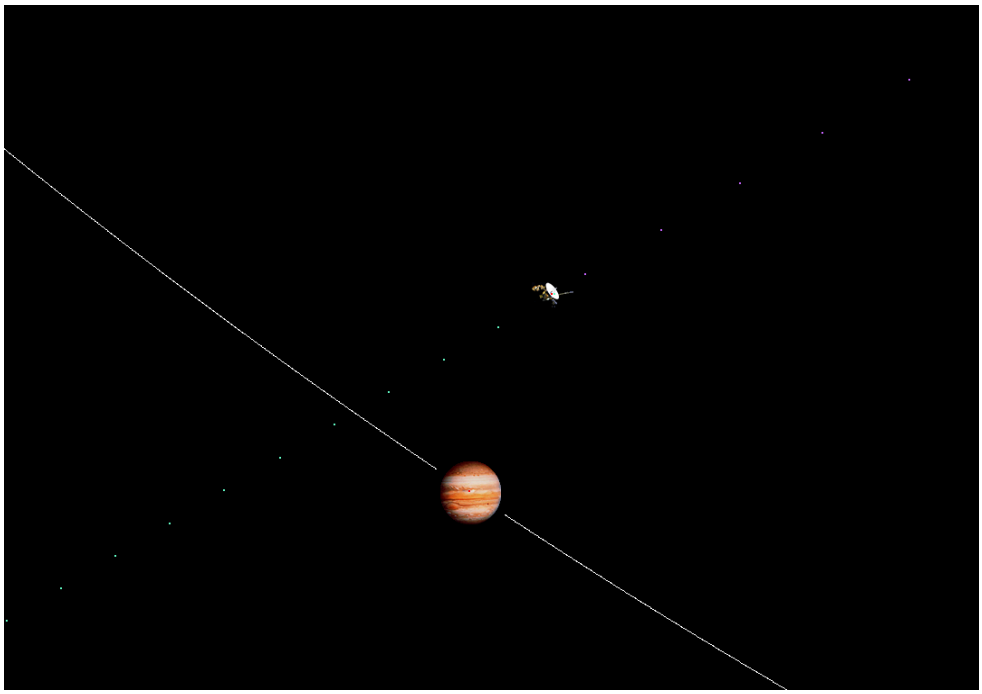


*Figure 3 : La sonde passant à proximité de Jupiter*

L'interface Homme-machine (pouvoir modifier les conditions initiales au début de la simulation) que nous avons choisi utilise l'IDE *QtCreator*. Cependant il est compliqué d'installer la SDL sur cet IDE.
Dans un premier temps nous avons donc tenté de réécrire le code SDL en utilisant les bibliothèques proposées par Qt. Nous voulions faire une interface tout-en-un proposant une fenêtre de préférences et une scène en tant que widget principal. Cette interface est composée d'une fenêtre principale héritant de la classe QMainWindow, gérant les fenêtres principales, et la fenêtre de préférences est un QWidget composé de plein d'autres objets héritant de la classe QWidget.

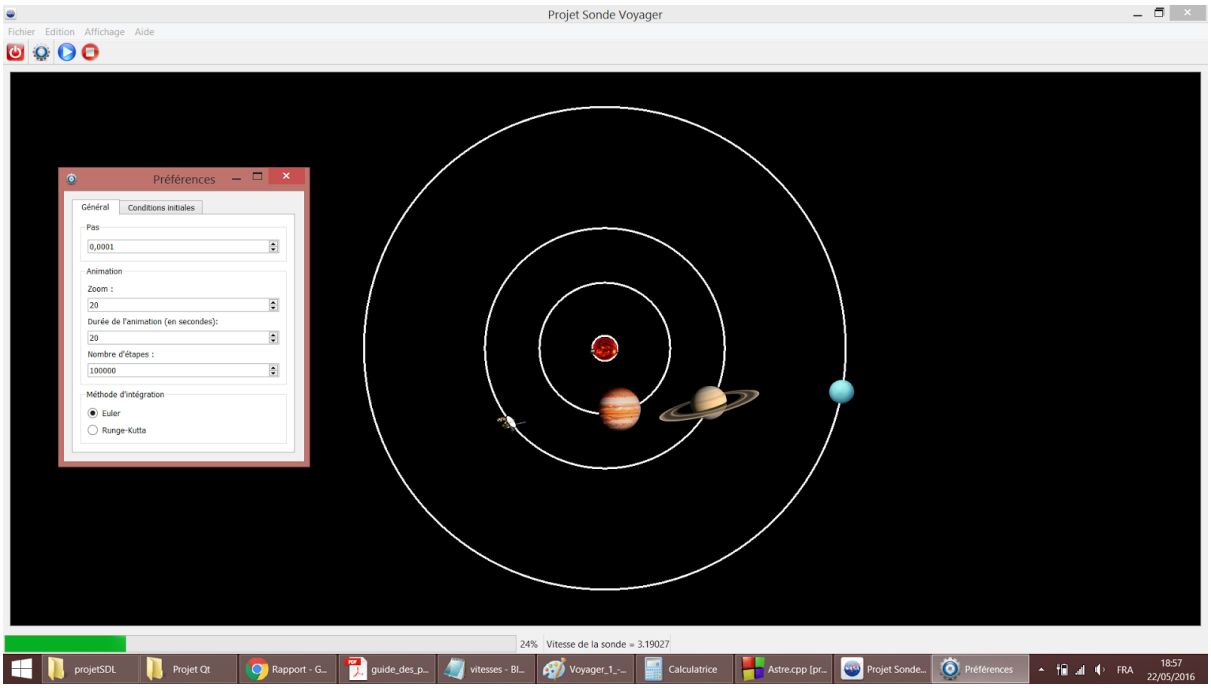



*Figure 4 : Logiciel tout-en-un développé uniquement avec la bibliothèque Qt*

Bien que le résultat d'image était le même, la perte de fluidité nous à obligé à chercher une autre solution.

Nous avons donc décidé de créer les deux programmes de façon indépendante, d'une part le programme avec la fenêtre de conditions initiales et un bouton pour lancer la simulation et d'autre part le programme SDL que nous avions programmé auparavant.

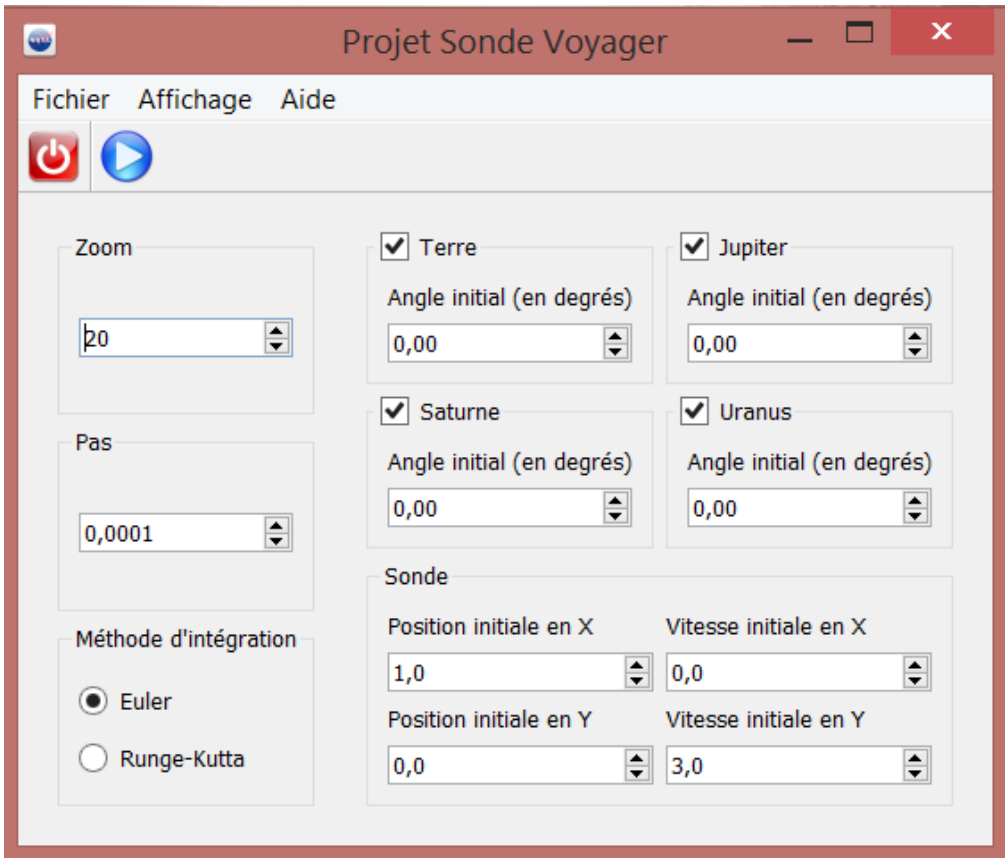



*Figure 5 : Interface homme-machine finale développée avec Qt*

```
class FenPrincipale : public QMainWindow
{
    Q_OBJECT

    public:
        FenPrincipale();
        ~FenPrincipale();
        void ecrireFichier();

    public slots:
        void ouvrirFenetreAPropos();
        void lancerSimulation();

    private:
        FenPreferences *fenetrePreferences;
        QWidget *zoneCentrale;
        QLayout *layout;
        QGraphicsScene *scene;
        QGraphicsView *view;
};
```

*Figure 6 : Classe utilisée pour l'interface homme-machine*

Pour transférer les paramètres d'un programme à l'autre nous utilisons un fichier texte. Le premier programme remplit ce fichier et le second le lit.

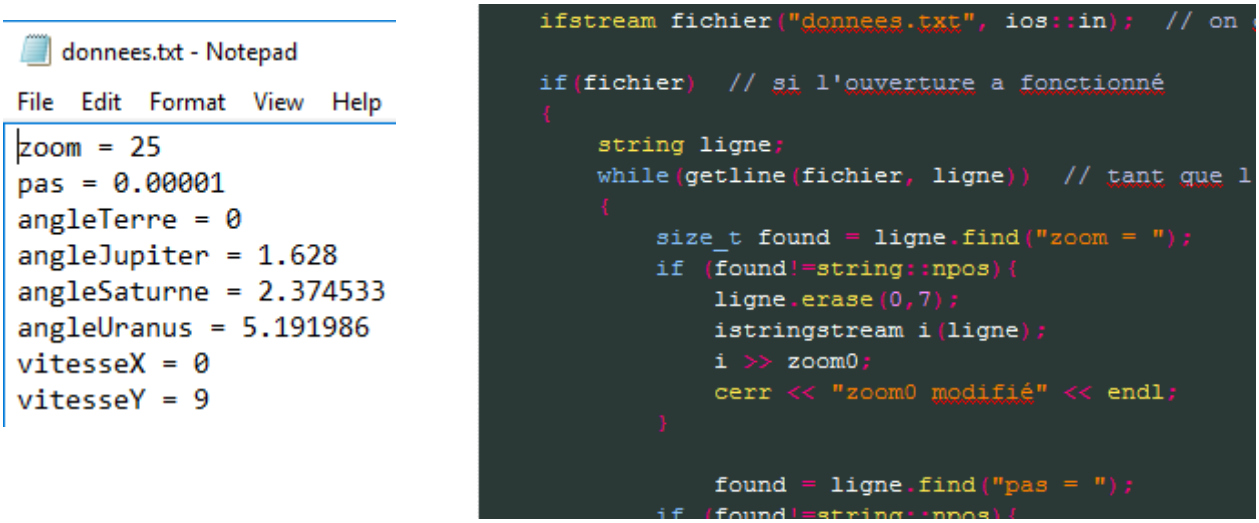



*Figure 6 : Fichier texte contenant les données et partie du code de la fonction permettant de les lire*

# II. Résultats

## A. Simulation de la sonde dans le système solaire

Grâce aux différents outils que nous avons créés, nous pouvons en premier lieu observer le mouvement d'une sonde dans le système solaire. Ensuite, nous pouvons observer l'influence de l'effet fronde sur la trajectoire de la sonde.

En effet, notre modélisation nous permet d'observer le déplacement d'une sonde partant de la Terre et allant dans une direction choisie tout d'abord de manière aléatoire. Ceci nous permettant tout d'abord d'observer les phénomènes de gravitation sur une sonde dans l'espace. Ce phénomène étant celui qui régit la trajectoire de la sonde, il est donc primordial de se rendre compte de l'influence de la distance entre deux astres. En effet, la force gravitationnelle est inversement proportionnelle au carré de la distance séparant deux objets. Cette force a donc un fort impact quand deux objets sont à des distance très petites.

Ainsi quand un objet se rapproche d'un autre beaucoup plus massif alors la force de gravitation devient de plus en plus importante. Chaque objet sera attiré l'un vers l'autre avec une force proportionnelle à la masse de l'autre objet. Dans le cas où l'on considère une sonde et une planète alors la planète ne verra pas sa trajectoire modifiée. Au contraire, la sonde sera déviée de plus en plus selon sa distance à la planète. On peut donc se servir de ce phénomène à notre avantage. Lorsque la sonde arrive derrière une planète elle sera attiré vers elle et verra donc sa vitesse augmenter. Ensuite, la planète et la sonde ayant des trajectoires différentes, l'attraction de la planète sur la sonde la ralentit un petit peu. Mais comme la planète s'éloigne rapidement de la sonde la force considérée redevient faible. On obtient donc une accélération et une modification de trajectoire rappelant celle de la fronde, cet effet est donc appelé effet fronde ou assistance gravitationnelle.

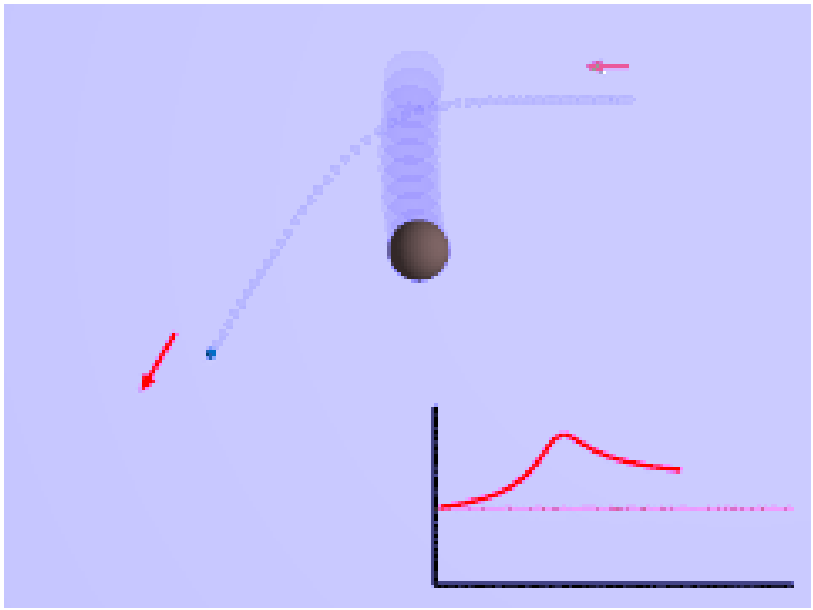


*Figure 7 : Schéma de l'effet fronde et courbe de la vitesse de l'objet*

La simulation permet de voir la trajectoire de la sonde lorsque l'on la dirige vers Uranus, mais cette simulation n'a pas grand intérêt si seule la trajectoire change. En effet, on peut facilement observer l'effet fronde sur la trajectoire de la sonde Voyager.

C’est dans l’objectif de mieux percevoir l’effet fronde que nous avons réalisé une fonction qui trace la trajectoire de la sonde au fur et à mesure ainsi qu’une autre qui est la trajectoire sans perturbation extérieur. On observe ainsi clairement le changement de direction que prend la sonde lorsqu’elle approche d’une planète. De plus on observe les phases d’accélération grâce à l’affichage de la vitesse en temps réel.

Ainsi notre simulation permet d’observer la trajectoire que décrira la sonde si on l’envoie avec les différentes conditions initiales définies. En effet, on choisit la vitesse initiale de la sonde ainsi que sa direction mais aussi la positon angulaire des planètes au départ de la simulation.

## B. Confrontation avec le modèle théorique

Pour aller de la Terre à Uranus en empruntant une demi-ellipse, il faut théoriquement 16 ans sans effet de fronde.

*Quel gain de temps peut-on alors obtenir avec l’assistance gravitationnelle ?*

Pour répondre à cette question, nous allons nous servir de la simulation.
Tout d’abord, nous voulions reprendre les mêmes conditions initiales de la sonde (position et vitesse au lancement) pour reproduire au mieux ce qui a été réalisé. Ensuite, nous avons réalisé un graphique représentant la vitesse de la sonde en fonction de sa distance par rapport au Soleil.

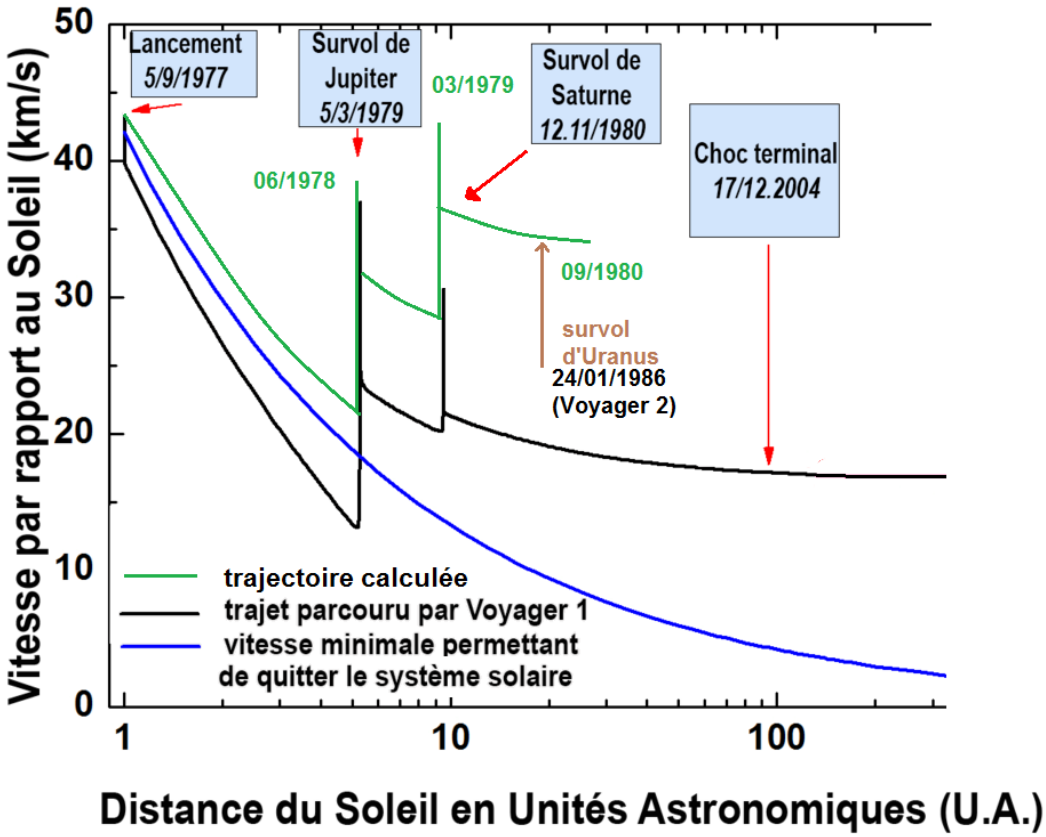



*Figure 8 : Vitesse de la sonde par rapport au Soleil en fonction de sa distance*

D'après la figure 8, on constate avec la simulation que la sonde prend 3 ans pour atteindre Uranus alors que Voyager 2 a mis 8 ans et 5 mois. La durée pour atteindre Uranus a donc été divisée par 2 dans les faits et divisée par 5 avec la simulation. L'assistance gravitationnelle a donc été très bénéfique. Cependant, on constate que la vitesse de la sonde dans la simulation a été plus grande que dans la réalité, on aurait donc pu économiser du carburant.

### C. Approximations réalisées

Différentes raisons permettent d'expliquer les différences constatées entre la théorie et la simulation.

Tout d'abord, on a utilisé des positions de planètes qui nous arrangeaient le plus pour la modélisation de l'effet fronde. En effet, l'envoi effectif de la sonde ne peut s'effectuer qu'à certaines dates, à une fréquence de 10, 100, 1000 ans, voire jamais selon le nombre et la précision des critères de la configuration choisie. Par exemple, la configuration particulière des quatre géantes gazeuses qui a rendu le survol possible ne se reproduit que tous les 176 ans. La configuration de nos planètes a été trop optimisée et ces conditions optimales n'ont sûrement pas été possibles en pratique.

Ensuite, nous traitons le problème qu'en deux dimensions. En implémentant la 3D, les distances relatives ne seront plus les mêmes et un effet de fronde mal orienté aurait pu faire dévier la trajectoire du plan de l'écliptique. De même, en réalité l'orbite des planètes est elliptique donc cela influe sur la distance à parcourir.

De plus, on a considéré les planètes et la sonde comme ponctuelles. On s'est donc juste limité sur la mécanique du point. Les planètes telluriques n'ont pas été prises en compte non plus alors qu'elles ont une petite influence sur la sonde. La masse de la Terre a été considérée nulle car elle impactait trop sur le décollage de la sonde. En effet, la distance entre la Terre et la sonde au décollage étant trop petite, il fallait fournir une vitesse initiale importante.

Enfin, la résolution des équations différentielles par la méthode d'Euler est une autre source d'erreurs car elle est la plus simple à réaliser. En effet, si le pas de calcul est trop élevé, l'erreur induite peut être assez élevée. A l'inverse, si le pas est trop faible, le temps de calcul devient trop long. Il fallait donc choisir le meilleur compromis.

# III. Gestion de projet

## A. Description des tâches

Dans le but de réaliser le projet de la manière la plus optimale, nous avons organisé les différentes tâches entre nous. En premier lieu, il fallait préparer le travail. Ensuite, on avait besoin de programmer les différents outils de calcul nécessaires à la simulation. Enfin, il était nécessaire de réaliser une interface graphique pour représenter les résultats des calculs.

Dans un premier temps, un travail de préparation nous avait été confié par le tuteur. Ce travail a permis de définir les valeurs théoriques et les relations nécessaires à la réalisation du projet. On a donc défini la valeur du temps de trajet sans effet de fronde, les formules de gravitation portant sur la sonde ainsi que les formules de l'accélération de la sonde. Pour réaliser ce travail, une séance de projet a été nécessaire.

Suite au travail de préparation, les différentes classes ont été définies et on a procédé aux tests. On a d'abord essayé de vérifier la bonne interaction entre la sonde et le Soleil : la sonde devait tourner autour du Soleil si on lui fournissait une certaine vitesse. On a alors ajouté les planètes participant à l'effet fronde. Enfin, on a recherché les conditions initiales qui permettaient d'avoir un effet fronde optimal. Pour cela nous avions prévu entre 1 mois et 1 mois et demi.

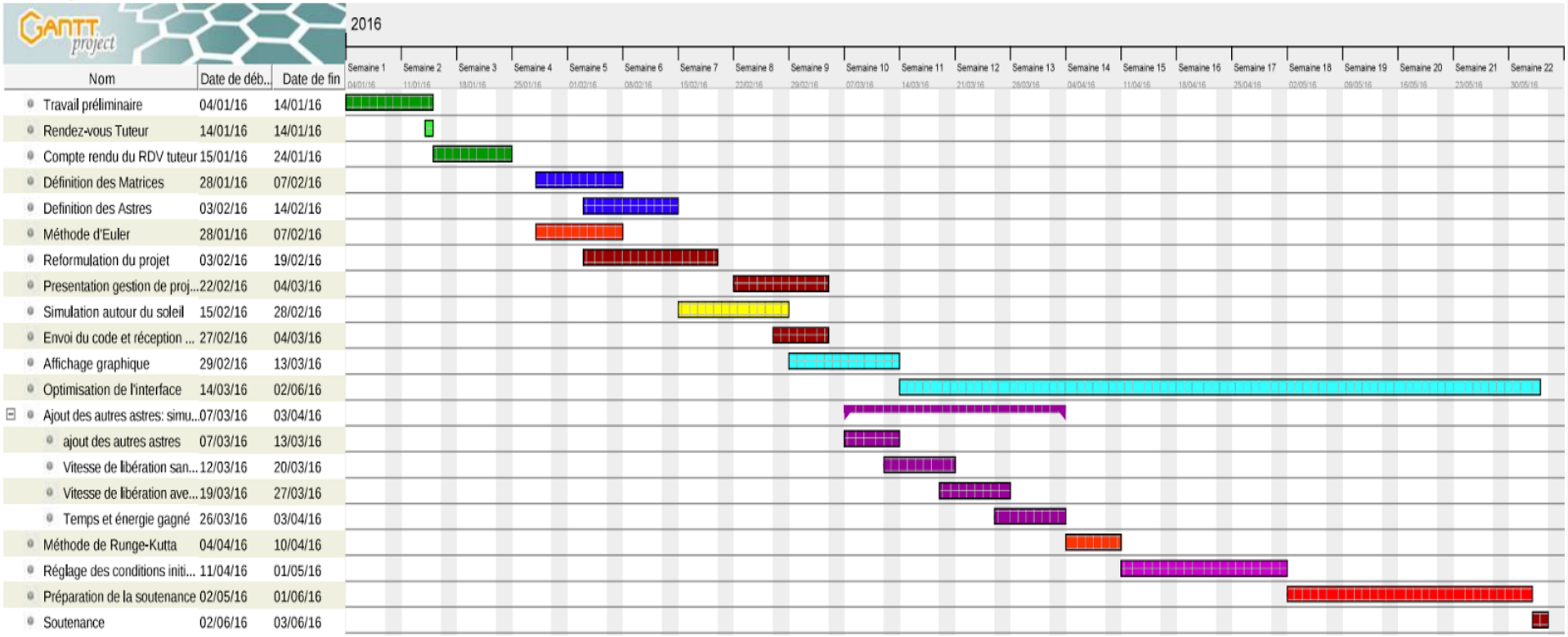
Enfin, une interface graphique a été développée afin de visualiser l'ensemble des résultats. Dans un premier temps, on a crée une animation qui montre la sonde qui se déplace dans le système solaire en 2 dimensions. On a consacré 2 semaines pour cela. Ensuite on a réalisé une interface qui permet à l'utilisateur de rentrer des valeurs des positions des planètes et de la sonde pour modifier la simulation. De plus, ceci avait pour but de nous permettre de lancer un programme pour voir la simulation et de ne pas appeler cette dernière via la console. Pour cela on a considéré que un mois de notre temps serait nécessaire si cela n'était pas notre priorité.

# projet voyager

19 févr. 2016

## Diagramme de Gantt

3




| Nom | Date de déb.. | Date de fin |
|---|---|---|
| Travail préliminaire | 04/01/16 | 14/01/16 |
| Rendez-vous Tuteur | 14/01/16 | 14/01/16 |
| Compte rendu du RDV tuteur | 15/01/16 | 24/01/16 |
| Définition des Matrices | 28/01/16 | 07/02/16 |
| Definition des Astres | 03/02/16 | 14/02/16 |
| Méthode d'Euler | 28/01/16 | 07/02/16 |
| Reformulation du projet | 03/02/16 | 19/02/16 |
| Presentation gestion de proj... | 22/02/16 | 04/03/16 |
| Simulation autour du soleil | 15/02/16 | 28/02/16 |
| Envoi du code et réception ... | 27/02/16 | 04/03/16 |
| Affichage graphique | 29/02/16 | 13/03/16 |
| Optimisation de l'interface | 14/03/16 | 02/06/16 |
| Ajout des autres astres: simu... | 07/03/16 | 03/04/16 |
| ajout des autres astres | 07/03/16 | 13/03/16 |
| Vitesse de libération san... | 12/03/16 | 20/03/16 |
| Vitesse de libération ave... | 19/03/16 | 27/03/16 |
| Temps et énergie gagné | 26/03/16 | 03/04/16 |
| Méthode de Runge-Kutta | 04/04/16 | 10/04/16 |
| Réglage des conditions initi... | 11/04/16 | 01/05/16 |
| Préparation de la soutenance | 02/05/16 | 01/06/16 |
| Soutenance | 02/06/16 | 03/06/16 |

## B. Analyse critique

Ce projet étant fini, on peut tirer des conclusions de ce qui a été fait et ce qui ne l'a pas été. La gestion de ce projet a plusieurs aspects, tout d'abord le temps et ensuite l'ordre des priorités.

La gestion de notre temps pour réaliser se projet a été plutôt bonne. Nous avons essayé de respecter les intervalles de temps prévu dans le diagramme de Gantt. Dans la majorité des cas, ce temps été respecté mais c'était sans compter sur les différents problèmes que nous avons rencontré. Nous avons donc revu certaines priorités pour compenser le fait que nous n'avions pas prévu de temps pour résoudre les problèmes. De ce fait nous n'avons pas eu le temps de programmer la méthode de Runge-Kutta.

L'ordre des priorités a été dur à définir. En effet, nous avons essayé au début de définir un ordre qui nous semblait logique. Par la suite, nous avons réalisé les tâches dans un ordre où la réalisation de telle tâche était nécessaire et cela a différé dans certains cas.

Pour organiser nos différents avancements personnels du programme, nous avons eu recours à GitHub, un site internet qui nous permet de rassembler nos différentes parties du programme. Ceci a été d'une grande utilité car nous pouvions travailler ensemble en conférence audio et en travaillant sur le même code.

Notre budget a été parfaitement maîtrisé. En effet, il était de zéro euros et nous avons été capable de réaliser cette simulation sans aucun moyen financier.

## C. Apport personnel et évaluation latérale

**Gaëtan:**

Ce projet m'a permis d'associer mes connaissances de différents domaines. De plus pour mettre en commun ces connaissances de physique et d'algorithmique il m'a fallu apprendre différents langages informatiques. Enfin ce projet m'a appris beaucoup de choses dans le domaine du travail en groupe. L'organisation n'est pas aussi simple qu'il ne paraît, il est difficile d'estimer un temps de travail où on pourrait rencontrer des problèmes.

**Francis :**

Ce projet combinant deux domaines que j'apprécie, la mécanique et l'informatique, m'a permis d'apprendre les enjeux de la simulation numérique et la programmation orientée objet. De plus, il m'a permis de développer ma capacité à travailler en groupe grâce au partage du code et du travail.

**Antoine :**

La conception d'un programme de cette envergure m'a permis de mettre en application les connaissances de C++ que j'avais acquises jusqu'à présent. De plus le fait de travailler en groupe sur un projet informatique accélère la conception de celui-ci et permet de s'entraider en cas de problèmes.

**Maxime :**

Ce projet fût pour moi l'opportunité de mettre en pratique les connaissances que j'avais acquises en programmation orientée objet en C++, et de travailler en équipe sur un projet essentiellement informatique. Ce type de projet demande beaucoup de structuration et d'organisation, car sans cela, le travail est difficile à mettre en commun.

**Évaluation latérale:**

| note de : | Cavaillès Gaëtan | Delbeque Antoine | Guiblin Maxime | Shih Francis | Moyenne |
|---|---|---|---|---|---|
| Cavaillès Gaëtan | x | 18 | 18 | 20 | 18.7 |
| Delbeque Antoine | 19 | x | 20 | 19 | 19.3 |
| Guiblin Maxime | 18 | 20 | x | 18 | 18.7 |
| Shih Francis | 20 | 18 | 18 | x | 18.7 |

# Conclusion

Nous avons pu, à l'aide des différents algorithmes et de l'interface graphique développés, mettre en évidence l'effet de fronde gravitationnelle et son utilité dans le cadre d'un voyage stellaire, en dépit des approximations qui ont été faites.

Pour rendre la simulation plus réaliste, on pourrait travailler en 3D, avoir des orbites elliptiques, prendre en compte toutes les planètes et implémenter la méthode de Runge-Kutta.

De plus, ce projet permet de traiter d'autres problèmes physiques nécessitant des outils numériques. En effet, l'interaction entre deux masses est similaire à l'interaction de deux particules chargées.

Ce projet a été pour nous l'opportunité de pouvoir travailler en équipe, de mettre nos compétences et connaissances en informatique à l'épreuve et de s'inscrire dans une démarche de recherche dans le but de résoudre un problème relativement ouvert sans unique solution ; ce qui représente, à nos yeux, un savoir-faire indispensable dans la qualification de tout ingénieur.