Valentin Frydrychowski
Clément Truillet

UNIVERSITÉ TOULOUSE III PAUL SABATIER Université de Toulouse

Projet C : Modélisation de la trajectoire d'un point

1. Introduction

Le but de ce projet est de permettre de représenter graphiquement la trajectoire d'un point étant données les équations aux dérivées partielles définissant cette trajectoire.
Lorsque le point est soumis à des forces dépendant uniquement de sa position et de sa vitesse, le Principe Fondamental de la dynamique implique qu'il est possible de calculer sa trajectoire en fonction de ces deux paramètres.
Dans ce projet, on part des équations définissant la vitesse en fonction de la vitesse et de la position, et on met à jour la position du point en fonction de la vitesse et d'un petit incrément de temps paramétrable dt.1

Ce projet se réalise par binôme qui est composé ici de Valentin Frydryschowski et de Clément Truillet.

2. Librairies

2.1. Log

Afin de pouvoir avoir une trace de ce que fait le programme et de pouvoir rapidement voir où sont les erreurs de programmation, nous avons décidé de créer une librairie log dont la seule fonction est d'écrire dans un fichier .log créé au préalable et stocké dans le répertoire log.

Le nom de ce fichier .log sera constitué de la date et l'heure du lancement du programme afin que chacun soit unique.

Exemple : le fichier .log associé au lancement de l'exécutable le 10/11/2018 à 10h 58 et 23s sera nommé 20181011-105823.log)

2.1.1. Création du fichier .log

Fonction prenant en paramètre un tableau de char et créant un fichier de la forme date-time.log (ouverture en mode w).
Le nom du fichier log est stocké dans la variable FICHIER afin de pouvoir écrire dans ce fichier log pendant l'exécution.

Cette fonction nécessite la bibliothèque <time.h> et ne renvoie rien.

Prototype : `void crea_log(char * FICHIER);`

2.1.2. Ecriture dans le fichier .log

Fonction prenant en paramètres une chaîne de caractères et l'écrivant dans le fichier .log (ouverture en mode a).

Cette fonction renvoie 0 (l'écriture a été réalisée) ou 1 (écriture impossible, fin du programme et message d'erreur).

Prototype : `int w_log(char * FICHIER, char * str);`

2.2. Entrées

Dans la mesure où certains objectifs de ce projet nécessitent une entrée des valeurs de la part de l'utilisateur, nous avons décidé de dédier une librairie pour cela.
Cette librairie utilisera majoritairement la fonction scanf() de la bibliothèque <stdio.h> afin de récupérer les données entrées par l'utilisateur.
On pourrait reprocher à cette librairie de ne comporter que des fonctions extrêmement similaires (voire identiques) mais elles sont divisées par un volontée de clarté de notre côté.
De cette manière, nous pouvons effectuer un contrôle sur les valeurs entrées et proposer un mode "par défaut" où les valeurs sont prédéfinies.

Aussi, afin de contrôler le bon fonctionnement de ces interactions, une écriture dans le fichier .log sera faite en début et en fin de chaque fonction avec les valeurs entrées.

2.2.1. Demande des positions initiales

Fonction prenant en paramètres l'adresse d'un point de type Coordonnees *(cf. Position)* et l'adresse d'un float Tmax (temps d'arrêt du calcul de la position) et leur associe les valeurs x, y, z et Tmax entrées par l'utilisateur (par défaut, x = 1, y = 2 et z = 3).

Prototype : `void init_position(Coordonnees *point, float *Tmax);`

2.2.2. Demande des paramètres

Fonction prenant en paramètres les adresses des trois paramètres des équations de lorentz de type float (S,B et P représentent respectivement σ, β et ρ) et leur associe les valeurs entrées par l'utilisateur (par défaut σ = 10, β = 8/3 et ρ = 28).

Prototype : `void init_param(float *S, float *B, float *P);`

2.2.3. Demande de la mise à jour de la vitesse

Fonction prenant en paramètres l'adresse d'un float dt et qui lui associe l'incrément de la vitesse entrée par l'utilisateur.

Prototype : `void maj_vitesse(float *dt);`

2.3. Position

Ce projet s'axant assez principalement sur le calcul de nouvelles positions d'un point repéré dans l'espace à 3 dimensions, il nous semblait assez évident de créer une librairie qui va définir la structure Coordonnees et calculer la nouvelle position à t+dt.
Un appel à w_log sera fait en cas d'erreur.

PS : L'écriture de ce rapport ayant été effectué avant avoir vu les structures en TP, la structure Coordonnees est définie, ici, dans le fichier .h.
Maintenant éclairés par les bonnes pratiques, nous définirons la structure dans le fichier .c (ce qui sera pris en compte dans la réalisation du deuxième rapport).

2.3.1. Structure Coordonnées

Structure regroupant quatre float nommés, dans l'ordre, t, x, y et z.

2.3.2. Création d'une position

Fonction prenant en paramètres l'adresse du point de type Coordonnees ainsi que quatres variables de type float nommées respectivement t, x, y et z et qui les stockent dans la variable point.

Prototype : `void new_coord(Coordonnees *point, float t, float x, float y, float z);`

2.3.3. Calcul de la nouvelle position

Fonction prenant en paramètres l'adresse du point de type Coordonnees,l'incrément dt de type float ainsi que les paramètres σ, β et ρ et qui calcule la nouvelle position et la stocke dans point.

Prototype : `void position_next(Coordonnees *point, float dt, float S, float B, float P);`

2.4. Fichier

Le stockage des positions des points en fonction du temps étant demandé, nous avons choisi de consacrer une librairie, similaire à Log, afin de créer, stocker les positions et lire dans ce fichier nommé position.txt.
De la même façon qu'avec Position, un appel à w_log sera fait en cas d'erreur.

2.4.1. Initialisation et création du fichier

Fonction prenant en paramètres l'adresse d'un tableau de char nommé FICHIER (fichier de stockage des positions) et un tableau de char nommé LOG (nom du fichier .log) et qui crée un fichier vide nommé FICHIER.dat. Si ce fichier existe déjà, il est vidé (ouverture en mode w+).
La variable FICHIER est constante car elle n'est pas sensée changer durant toute l'exécution du programme.

Prototype : `void init_fichier(const char * FICHIER, char * LOG);`

2.4.2. Ecriture dans le fichier

Fonction prenant en paramètres l'adresse du point de type Coordonnees et qui écrit dans le fichier à la ligne (ouverture en mode a).
PS : Les premiers tests portant sur l'écriture dans le fichier ont montrés une certaine lenteur d'exécution dans le cas d'ouverture et fermeture du fichier à chaque écriture d'une nouvelle position (2min30 au lieu de moins d'une seconde dans le cas d'une seule ouverture/fermeture). Voici pourquoi la fonction w_fichier prend en paramètre un pointeur sur une variable de type FILE.

Prototype : `void w_fichier(FILE * fichier, Coordonnees * point);`

2.5. Gnuplot

L'affichage de la courbe étant, ici, extrêmement lié à la bibliothèque Gnuplot, il nous a semblé pertinent de créer une librairie Gnuplot qui servira à tracer la courbe à partir des positions calculées.
La lecture de la documentation Gnuplot étant aujourd'hui inachevée, il est fort probable qu'entre l'écriture de ce rapport et la fin du projet, certaines fonctions s'ajoutent dans cette librairie.
De la même façon, les paramètres des fonctions sont susceptibles de changer.

2.5.1. Tracer la courbe

Fonction ne prenant rien en paramètre et qui appelle le fichier lorentz.sh qui lance gnuplot avec les bons paramètres (contenus dans le fichier parameters).
Prototype : `void trace_courbe();`

2.5.2. Afficher le temps par un changement de couleur

Fonction prenant en paramètres l'adresse du point de type Coordonnees et la valeur Tmax de type float et qui renvoie la couleur dans laquelle dessiner le point afin d'avoir un dégradé temporel.

Prototype : `long color_time(Coordonnees * point, float Tmax);`

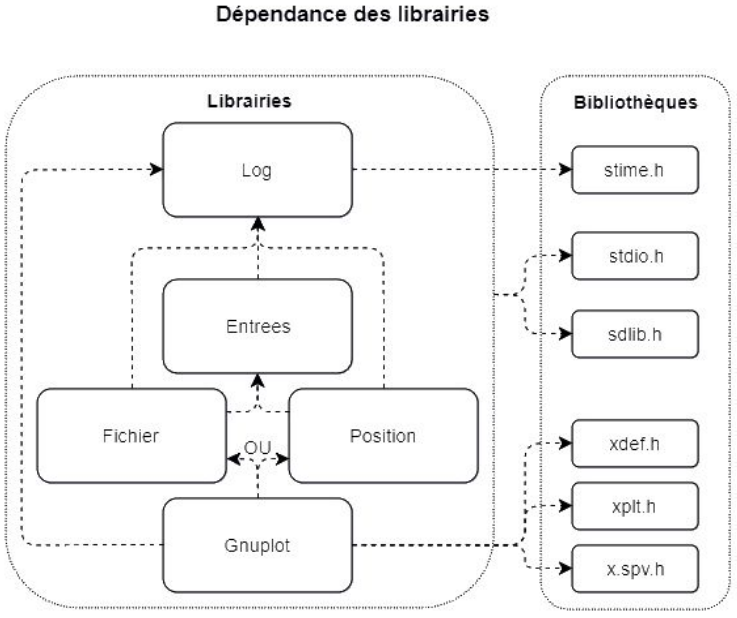
3. Dépendance des librairies

3.1. Bibliothèques standard

Afin de pouvoir avoir un code fonctionnel, nous aurons besoin des bibliothèques <stdlib.h> et <stdio.h>.
Aussi, pour la fonction crea_log de la librairie log, nous avons besoin de la bibliothèque <stime.h>.
Pour utiliser Gnuplot, nous avons aussi besoin de <xdef.h>, <xplt.h> (contrôle de la fenêtre Gnuplot) et <xspv.h> (initialiser des points et vecteurs).
PS : Après recherches, Gnuplot ne nécessite aucune librairies autre que son installation sur le système d'exploitation utilisé.
PS 2 : De plus, si le programme est lancé sous Windows (ou le sous système d'exploitation linux de windows), il faut installer et lancer un serveur X.

3.2. Librairies

Par définition, l'intégralité de nos librairies ont besoin de la librairie Log.
Aussi, les librairies Entrées et Position dépendent de Fichier afin de stocker les positions dans position.txt.
Dans une vision où l'affichage des points se fait à partir du fichier position.txt, Gnuplot nécessiterait la librairie Fichier.
Dans une autre vision où les points sont placés en même temps qu'ils sont calculés, Gnuplot nécessiterait la librairie Position.
Le choix entre ces deux options n'étant toujours pas élucidé, les deux cas seront détaillés par la suite.



4. Organisation du programme principal

4.1. Initialisation

En premier lieu, le fichier .log est créé puis l'initialisation du fichier lorentz.dat (Librairies Log et Fichier).
Une fois ces deux fichiers initialisés, la librairie Entrées rentre sur le devant de la scène afin de récupérer les valeurs de la position initiale et les paramètres (σ, β et ρ et dt).

4.2. Calcul

Avec la librairie Position, nous allons pouvoir calculer la position du nouveau point à t+dt et le stocker dans position.txt (librairie Fichier).

4.3. Affichage

L'affichage nécessite en premier lieu de la librairie Gnuplot ainsi que, au choix la librairie Fichier ou bien la librairie Position suivant de comment nous voulons afficher les points (cf. 2.2 Librairies).

5. Difficultées attendues et Solutions

5.1. De la confiance avec l'utilisateur

Après avoir débattu et suivant le vieil adage "Il ne faut jamais faire confiance à l'utilisateur" nous nous sommes rendu compte de plusieurs problèmes au niveau des entrées.

- dt est trop petit et/ou négatif
- Tmax est trop grand et/ou négatif
- Les paramètres sont nuls
- Certaines, voire toutes les entrées ne sont pas du type attendu.

Afin d'éviter au maximum (voir totalement) ces erreurs, voici, respectivement, des solutions proposées :

- Fixer une valeur minimum pour dt qui sera choisie en cas de dt trop petit et/ou négatif.
- Fixer une valeur maximum pour Tmax ainsi qu'une valeur minimum (pour ne pas faire tourner le programme trop longtemps et surcharger la courbe).
- Quand les paramètres sont nuls, prendre les paramètres par défaut et non ceux entrées (σ = 10, β = 8/3 et ρ = 28).
- Redemander l'entrée des valeurs si elles ne sont pas dans le type attendu, si trop de redemande, arrêter le programme.

5.2. De la compréhension de Gnuplot

Gnuplot étant nouveau pour nous, il va sans dire qu'il y a ici une certaine difficulté attendue tant par la compréhension, que par le tri des informations de la documentation Gnuplot et par l'utilisation de Gnuplot.