UNIVERSITÉ TOULOUSE III PAUL SABATIER Université de Toulouse

Projet C : Modélisation de la trajectoire d'un point

---

1. Librairies

1.1. 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 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)

1.1.1. Génération du nom du fichier log à partir de l'heure courante

Fonction prenant en paramètres l'adresse de deux variable de type long et leurs associant respectivement la date et l'heure de l'exécution du programme

Prototype : `void seed(long *date, long *time);`

1.1.2. Création du fichier .log

Fonction ne prenant pas de paramètres et créant un fichier de la forme date-time.log (ouverture en mode w).

Cette fonction nécessite l'appel de la fonction seed et renvoie 0 (le fichier .log a bien été créé) ou 1 (le fichier .log n'a pas été créé, fin du programme et message d'erreur).

Prototype : `int crea_log();`

1.1.3. Ecriture dans le fichier .log

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

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

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

1.2. Entrées

Dans la mesure où certains objectifs de ce projet nécessite une entrées 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 (voir 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.

1.2.1. 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 et leurs associent 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);`

1.2.2. Demande des paramètres

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

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

1.2.3. Demande de la MàJ de la vitesse

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

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

1.3. 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.

1.3.1. Structure Coordonnées

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

1.3.2. 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 new_position(Coordonnees *point, float dt, float S, float B, float P);`

1.4. 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.

1.4.1. Initialisation et création du fichier

Fonction ne prenant pas de paramètres et qui crée un fichier vide nommé position.txt. Si ce fichier existe déjà, il est vidé (ouverture en mode w+).

Prototype : `void init_fichier();`

1.4.2. 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)

Prototype : `void w_fichier(Coordonnees * point);`

1.4.3. Lecture dans le fichier

Fonction prenant en paramètres l'adresse du point de type Coordonnees et un temps de type float et qui stocke dans point la position x,y et z à t = temps.

Prototype : `void read_fichier(Coordonnees * point, float temps);`

1.5. 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 que, entre l'écriture de ce rapport et la fin du projet, certaines fonctions s'ajoutent dans cette librairie.
De la même façon, les entrées des fonctions sont susceptibles de changer.

1.5.1. Tracer la courbe

Fonction prenant en paramètres l'adresse du point de type Coordonnees et qui dessine le point et le relie avec le point du temps t-dt.

Prototype : `void trace(Coordonnees * point);`

1.5.2. 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);`

2. Dépendance des librairies

2.1. Bibliothèques standard

Afin de pouvoir avoir un code fonctionnel, nous aurons besoin des bibliothèques <stdlib.h> et <stdio.h>.
Aussi, pour la fonction seed 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).

2.2. 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.

3. Organisation du programme principal

3.1. Initialisation

En premier lieu, le fichier .log est créé puis l'initialisation du fichier position.txt (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).

3.2. 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).

3.3. 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).

4. Difficultées attendues et Solutions

4.1. 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, voir 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 choisi en cas de dt trop petit et/ou négatif.
- Fixer une valeur maximum pour Tmax ainsi qu'une valeur minimum.
- Quand les paramètres sont nuls, prendre les paramètres par défaut et non ceux entrées ($\sigma = 10$, $\beta = 8/3$ et $\rho = 28$).
- Redemander l'entrée des valeurs si elles ne sont pas dans le type attendu, si trop de redemande, arrêter le programme.

4.2. De la compréhension de Gnuplot

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