TD 4 - Résolution de systèmes d'équations différentielles ordinaires

1 Introduction

Le but de l'exercice est d'utiliser une librairie pour calculer la solution d'un système d'équations différentielles ordinaire. Pour cela, il faut :

1. Installer la librairie "phynum".
2. Télécharger le fichier objet contenant l'équation différentielle.
3. Ecrire un programme principal (main) permettant de calculer pas à pas la solution.
4. Sauvegarder cette solution dans un fichier de sortie.
5. Examiner les résultats graphiquement avec gnuplot.

2 Installation de la librairie "phynum"

- Téléchargez le fichier "libphn.tar.gz" depuis le site indiqué.
- Décompressez le avec la commande "tar"
- Modifiez le Makefile pour l'adapter à votre compilateur puis compilez
- N'oubliez pas d'indiquer le chemin d'accès à "libphn.a" dans votre propre Makefile

3 Création du programme

Vous devez écrire un "main" qui fait appel à la fonction "rk4_p()" pour résoudre l'équation différentielle, et qui utilise le fichier objet "la_fp.o" contenant l'équation différentielle. Il faut faire attention au Makefile, et dans le programme C proprement dit aux variables globales utilisées pour passer les paramètre à la fonction "fp" SANS avoir besoin de les transmettre à la fonction "rk4_p".

– Le fichier “Equ_diff.h” peut ressembler à ceci :

```
#define _EQUDIF_H
#include "phynum.h"
void fp(double *, double *, int) ;
```

La définition de “fp” est obligatoire. C’est ainsi qu’elle est appelée par rk4_p, dont la définition est dans “phynum.h” :

```
void rk4_p (double *, double, double, int , double, int *,
            void (*fp) (double *, double *, int)) ;
```

Notez la façon d’utiliser un pointeur de fonction.

Le programme principal “main()” doit être précédé dans le fichier principal (“Equ_diff.c” par exemple) de la définition de 4 variables locale :

```
double prm_1 = 0.706 ;
double prm_2 = 0.0055 ;
double prm_3 = 0.1 ;
double prm_4 = 0.5 ;
```

Celles-ci sont utilisées dans “la_fp.o” grace à une déclartation “external” :

```
extern double prm_1, prm_2, prm_3, prm_4 ;
void fp(double *x, double *xp, int n) {
}
```

Je n’indique pas ici le contenu même de cette fonction.

4 Utilisation

Le système à résoudre comporte 3 équations différentielles couplées. Il faut donc créer un vecteur de taille 3, et l’initialiser. Vous prendrez comme conditions initiales

```
x[0]=x[1]=x[2]=0.0 ;
```

Prévoyez de pouvoir changer la valeur de la variable “prm_1”, et elle seulement dans l’intervalle $[0.70, 0.72]$. Le résultat est très sensible.

Les différents arguments de “rk4_p” sont :

```
rk4_p (x, tt, dt, neq, eps, &suite, fp) ;
```

x “double *”. Vecteur d’état du système à l’instant “tt” en entrée, et à l’instant “tt+dt” en sortie

tt “double”. Valeur de la variable libre (le temps)

dt “double”. Pas de temps. Ici, on conseille d’utiliser des valeurs entre 0.1 et 0.01.

neq “int”. Nombre d’équations différentielles (ici 3)

eps “double”. Précision relative exigée. testez son influence entre 10^{-3} et 10^{-9}.

suite “int”. “Flag” permettant de savoir si c’est le premier appel à “rk4_p” ou non. En pratique, mettez “suite = 0 en début de programme, et n’y touchez pous ensuite.

fp “void (*fp) (double *, double *, int)”. Fonction contenant l’équation différentielle, telle que définie plus haut.

5 Travail demandé

Une fois que votre programme fonctionne, tracez l’évolution des 3 variables en fonction du temps avec gnuplot pour différentes valeurs de “prm_1” pour des déurées d’évolution de quelques centaines d’unités. Variez la précision demandée et le pas de temps. Tracez également la trajectoire de phase à 3 dimension.

Ce système est l’exemple utilisé dans le cours de “CT8” du master recherche.