PROJET SIMULATION NUMÉRIQUE - LANGAGE C AVANCÉE ET E.D.O.

Travail individuel et personnel

Table des matières

1 Résolution numérique d'équations différentielles ordinaires

1.1 Problème de Cauchy

Le problème de Cauchy est de trouver $\boldsymbol{y} \in \mathcal{C}^1([a;b];\mathbb{R}^d)$ vérifiant

$$\left\{ \begin{array}{rcll} \boldsymbol{y}'(t) & = & \boldsymbol{f}(t,\boldsymbol{y}(t)) & \forall t \in [a,b] \\ \boldsymbol{y}(a) & = & \boldsymbol{y}_0 & \boldsymbol{y}_0 \text{ donné dans } \mathbb{R}^d \end{array} \right. \tag{1.1}$$

On a le théorème suivant

Théorème 1 *Si $f : [a,b] \times \mathbb{R}^d \to \mathbb{R}^d$ vérifie*

1. *$\boldsymbol{f}$ continue,*
2. *$\boldsymbol{f}$ lipschitzienne en $\boldsymbol{y}$ dans $[a,b] \times \mathbb{R}^d$. (i.e. $\|\boldsymbol{f}(t,\boldsymbol{y}_1) - \boldsymbol{f}(t,\boldsymbol{y}_2)]\| \leq K \|\boldsymbol{y}_1 - \boldsymbol{y}_2\|$; $K > 0$; $\forall t \in [a,b]$; $\forall \boldsymbol{y}_1 \in \mathbb{R}^d$, $\forall \boldsymbol{y}_2 \in \mathbb{R}^d$*

Alors le problème de Cauchy (1.1) admet une unique solution de classe $\mathcal{C}^1([a,b];\mathbb{R}^d)$.

1.2 Algorithmes numériques

On cherche à résoudre numériquement le problème de Cauchy (1.1) pour une fonction $\boldsymbol{f}$ donnée.

1.2.1 Principe

On subdivise l'intervalle $[a,b]$ par des points $t_0, \cdots, t_N$ équidistants :

$$t_i = a + ih \quad ; \quad i \in \{0, \ldots, N\} \text{ avec } h = \frac{b-a}{N}.$$

On cherche à calculer $N+1$ nombres $\boldsymbol{y}_0, \cdots, \boldsymbol{y}_N$ tels que

$$\boldsymbol{y}_i \approx \boldsymbol{y}(t_i)$$

Les méthodes numériques que nous utiliserons sont dites *méthode numérique par pas.* Ces algorithmes par pas sont divisés en deux types :

- *Les algorithmes à pas séparés ou méthodes à un pas :*
  Le calcul de $\boldsymbol{y}_{i+1}$ s'effectue uniquement à partir de $\boldsymbol{y}_i$ (méthode explicite) ou à partir de $\boldsymbol{y}_i$ et de $\boldsymbol{y}_{i+1}$ (méthode implicite).
- *Les algorithmes à pas liés ou méthodes à pas multiples :*
  Le calcul de $\boldsymbol{y}_{i+1}$ s'effectue à partir des $\boldsymbol{y}_i$, $\boldsymbol{y}_{i-1}$, ... (méthode explicite) ou à partir des $\boldsymbol{y}_{i+1}, \boldsymbol{y}_i$, ... (méthode implicite).

Nous n'étudierons ici que les méthodes à un pas explicite.

1.2.2 Méthode d'Euler-Cauchy

Du développement de Taylor, on obtient l'algorithme numérique suivant :

$$\left\{ \begin{array}{lcll} \boldsymbol{y}_{i+1} & = & \boldsymbol{y}_i + h\boldsymbol{f}(t_i, \boldsymbol{y}_i) & \forall i \in \{0, \ldots, N-1\} \\ \boldsymbol{y}_0 & = & \boldsymbol{y}(a) & \end{array} \right.$$

On a alors le théorème de convergence :

Théorème 2 *Si $\boldsymbol{f}$ vérifie les hypothèses du théorème 1 alors la méthode d'Euler-Cauchy converge :*

$$\max_{i \in \{1, \cdots, N\}} \|\boldsymbol{y}_i - \boldsymbol{y}(t_i)\| \xrightarrow[N \to \infty]{} 0.$$

1.2.3 Etude générale des méthodes à un pas

L'algorithme numérique est :

$$\left\{ \begin{array}{lcll} \boldsymbol{y}_{i+1} & = & \boldsymbol{y}_i + h\boldsymbol{F}(t_i, \boldsymbol{y}_i, h) & \forall i \in \{0, \ldots, N-1\} \\ \boldsymbol{y}_0 & = & \boldsymbol{y}(a) & \end{array} \right. \tag{1.2}$$

où $\boldsymbol{F}$ est une fonction à déterminer.
Le théorème suivant nous donne les hypothèses que doit vérifier $\boldsymbol{F}$ pour que l'algorithme (1.2) converge :

Théorème 3 *Soit $\boldsymbol{F}$ vérifiant*

1. $\boldsymbol{F}(t, \boldsymbol{y}, 0) = \boldsymbol{f}(t, \boldsymbol{y}) \quad \forall t \in [a,b] \quad ; \quad \forall \boldsymbol{y} \in \mathbb{R}^d$
2. $\|\boldsymbol{F}(t, \boldsymbol{y}_1, h) - \boldsymbol{F}(t, \boldsymbol{y}_2, h)\| \leq L \|\boldsymbol{y}_1 - \boldsymbol{y}_2\| \quad ; \quad \forall h \in [0, h_0] \quad ; \quad \forall t \in [a,b] \quad ; \quad \forall \boldsymbol{y}_1, \boldsymbol{y}_2 \in \mathbb{R}^d$ *et avec $L > 0$ indépendant de h.*

Alors la méthode (1.2) converge.

On a ensuite le théorème suivant

Théorème 4 *Si la méthode à un pas est stable et d'ordre p et si $\boldsymbol{f} \in \mathcal{C}^p$ alors il existe $C > 0$ tel que*

$$\|\boldsymbol{y}(t_i) - \boldsymbol{y}_i\| \leq Ch^p, \ \forall i \in \{0, \ldots, N-1\}$$

1.2.4 Exemples de fonction F vérifiant les hypothèses du théorème 3

- Méthode de la tangente améliorée (ordre 2)

$$\boldsymbol{F}(t,\boldsymbol{y},h) = \boldsymbol{f}(t+\frac{h}{2},\boldsymbol{y}+\frac{h}{2}\boldsymbol{f}(t,\boldsymbol{y}))$$

- Méthode d'Euler modifiée (ordre 2)

$$\boldsymbol{F}(t,\boldsymbol{y},h) = \frac{1}{2}(\boldsymbol{f}(t,\boldsymbol{y}) + \boldsymbol{f}(t+h,\boldsymbol{y}+h\boldsymbol{f}(t,\boldsymbol{y}))$$

- Méthode de Heun (ordre 2)

$$\boldsymbol{F}(t,\boldsymbol{y},h) = \frac{1}{4}\boldsymbol{f}(t,\boldsymbol{y}) + \frac{3}{4}\boldsymbol{f}(t+\frac{2h}{3},\boldsymbol{y}+\frac{2h}{3}\boldsymbol{f}(t,\boldsymbol{y}))$$

- Méthode de Runge-Kutta classique d'ordre 4

$$\boldsymbol{F}(t,\boldsymbol{y},h) = \frac{1}{6}(\boldsymbol{k}_1 + 2\boldsymbol{k}_2 + 2\boldsymbol{k}_3 + \boldsymbol{k}_4)$$

où $\boldsymbol{k}_1 = \boldsymbol{f}(t,\boldsymbol{y})$, $\boldsymbol{k}_2 = \boldsymbol{f}(t+\frac{h}{2},\boldsymbol{y}+\frac{h}{2}\boldsymbol{k}_1)$, $\boldsymbol{k}_3 = \boldsymbol{f}(t+\frac{h}{2},\boldsymbol{y}+\frac{h}{2}\boldsymbol{k}_2)$ et $\boldsymbol{k}_4 = \boldsymbol{f}(t+h,\boldsymbol{y}+h\boldsymbol{k}_3)$.

2 Exemple d'E.D.O. : le pendule pesant

On considère un pendule de masse M, fixé à une tige rigide de longueur L et de masse négligeable (voir figure 1), dans un milieu visqueux dont le coefficient de viscosité vaut k. On note θ l'angle formé par le pendule et l'axe verticale : il vérifie l'équation différentielle suivante (principe fondamental de la dynamique) :

$$\theta''(t) = -\frac{g}{L}\sin(\theta(t)) - \frac{k}{ML^2}\theta'(t),\ \forall t \geq 0 \tag{2.1}$$

avec les conditions initiales

$$\theta(0) = \theta_0 \text{ et } \theta'(0) = \theta'_0. \tag{2.2}$$

θ_0 est l'angle initial en radian et θ'_0 la vitesse angulaire initiale en radian/seconde.

On peut prendre, par exemple, $M = 1kg$, $L = 1m$, $g = 9.8m.s^{-2}$ et $k = 0.5USI$.

L'E.D.O. (2.1-2.2) ne peut être résolue de manière exacte. On se propose d'utiliser la méthode de Runge-Kutta d'ordre 4 pour l'étude des différents types de mouvements possibles, suivant les conditions de l'expérience (dans le vide $k = 0$, dans l'air $k = 0.1$, dans l'eau $k = 0.5$,...) et les conditions initiales imposées (pendule lancé, laché, ...). On souhaite ensuite tracer les deux courbes discrètes $\theta(t)$ et $\theta'(t)$.

3 Langage C

Les structures imposées sont :

```
typedef struct{
  int dim;
  double *pval;
}Vector;
```

```
/*************************************************************/
/* Structure pour le stockage de la solution d'un systeme d'EDO */
/*   - systeme de <dim> EDO du 1er ordre                     */
/*   - <N> nombre de discretisation (N < DISMAX)             */
/*   - <T> vecteur de dimension <N+1>                        */
/*   - <tY> tableau de <N+1> Vecteurs de dimension <dim>     */
typedef struct{
```

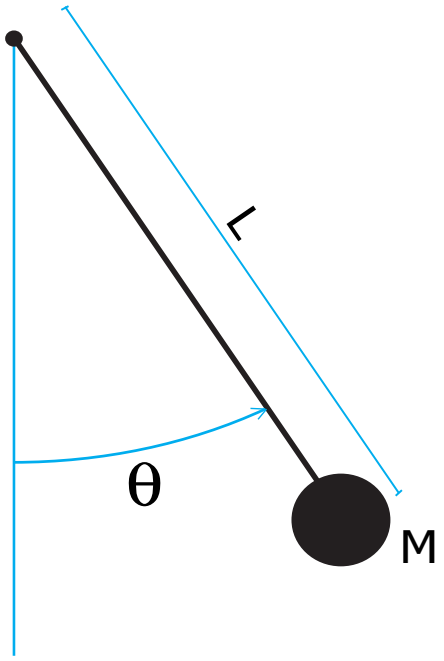

FIGURE 1 – Pendule pesant

```
8    int dim;   /* Systeme de <dim> EDO du 1er ordre */
9    int N;     /* Nombre d'intervalles de discretisation */
10   Vector T;
11   Vector *pY;
12 }SolSysEDO;
13 /*****************************************************************/
```

Ces structures sont loin d'être un choix optimal mais elles permettent d'illustrer les notions de langage C vues en cours.

3.1 programmes de validation

Ecrire un ensemble de programmes **structurés et commentés** permettant de tester et valider vos codes de résolution d'un problème de Cauchy (1.1). La compilation et l'éxecution de vos programmes se fera à l'aide d'un Makefile.

On peut prendre comme exemples d'E.D.O. le problème de Cauchy avec :

– Exemple 1 :

$$f(t,y) = -\sin(t) \quad \text{et} \quad y(0) = 1.$$

Dans ce cas la solution exacte est

$$y(t) = \cos(t).$$

– Exemple 2 :

$$\boldsymbol{f}(t,\boldsymbol{y}) = \begin{pmatrix} \cos(t) \\ -\sin(t) \\ t \end{pmatrix} \quad \text{et} \quad \boldsymbol{y}(0) = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}.$$

Dans ce cas la solution exacte est

$$\boldsymbol{y}(t) = \begin{pmatrix} \sin(t) \\ \cos(t) \\ \frac{t^2}{2} \end{pmatrix}.$$

Pour la validation des codes, il peut être intéressant de :

- représenter, pour un problème donné, l'erreur commise à chaque itération pour chacun des quatres schémas proposés (voir figure 2)

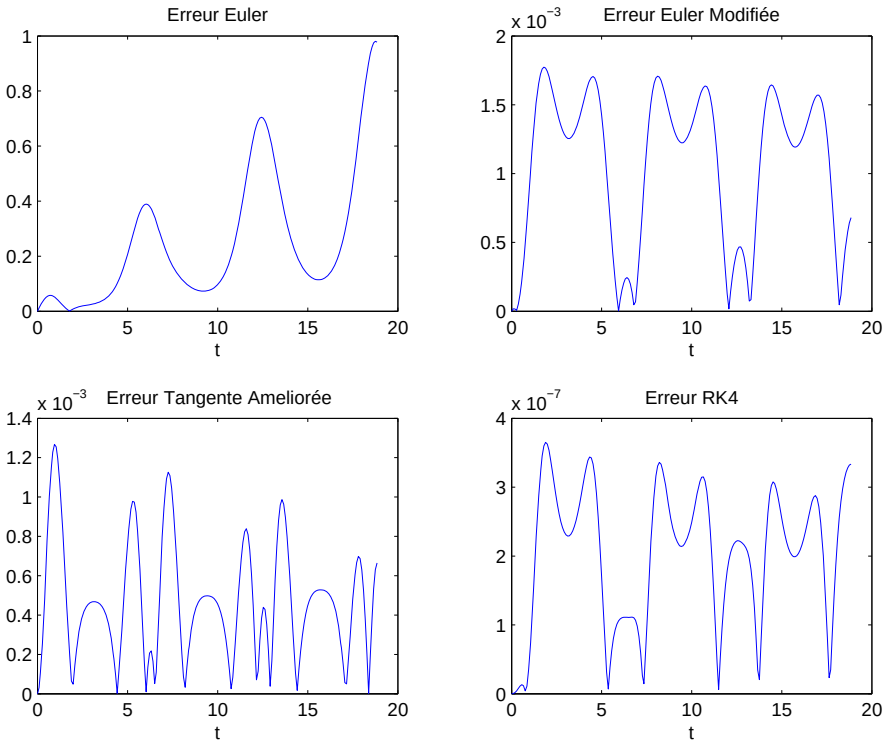

FIGURE 2 – Erreurs des différents schémas

- retrouver graphiquement l'ordre des schémas utilisés (voir figure 3)

Pour les représentations graphiques, les logiciels **Matlab** ou **octave** (version freeware de Matlab) seront utilisés.

3.2 Résolution du problème du pendule pesant

Ecrire un programme permettant de résoudre numériquement le problème du pendule pesant (2.1)-(2.2). La représentation graphique se fera à l'aide de **Matlab** ou **octave** par la lecture de fichier(s) de résultats créés par votre programme. Il faudra que votre programme soit écrit de manière à pouvoir très facilement modifier :

- les données initiales,
- le nombre de discrétisation,
- le schéma numérique utilisé.

La compilation et l'éxecution de vos programmes se fera à l'aide d'un Makefile.

On pourra, par exemple, écrire plusieurs programmes permettant de :

1. représenter les positions et vitesses d'un pendule
2. représenter les plans de phase d'un pendule (voir figure et)

4 Modalité de contrôle

- Avant le 25 janvier 2013 minuit [1], envoyer par mail à l'adresse `cuvelier@math.univ-paris13.fr` les fichiers archives (format unix `tar.gz` créé avec la commande `tar`) `<NOM.PRENOM>_EDOValidation.tar.gz` ou `<NOM.PRENOM>_EDOValidation.zip` [2] contenant l'ensemble des fichiers sources compressés permettant

1. Pénalité de retard (sur 20) -2 points pour tout retard et -1 par jour de retard
2. Pour l'étudiant Alain Dupont, le champ `<NOM.PRENOM>` va correspondre à `DUPONT.ALAIN`.

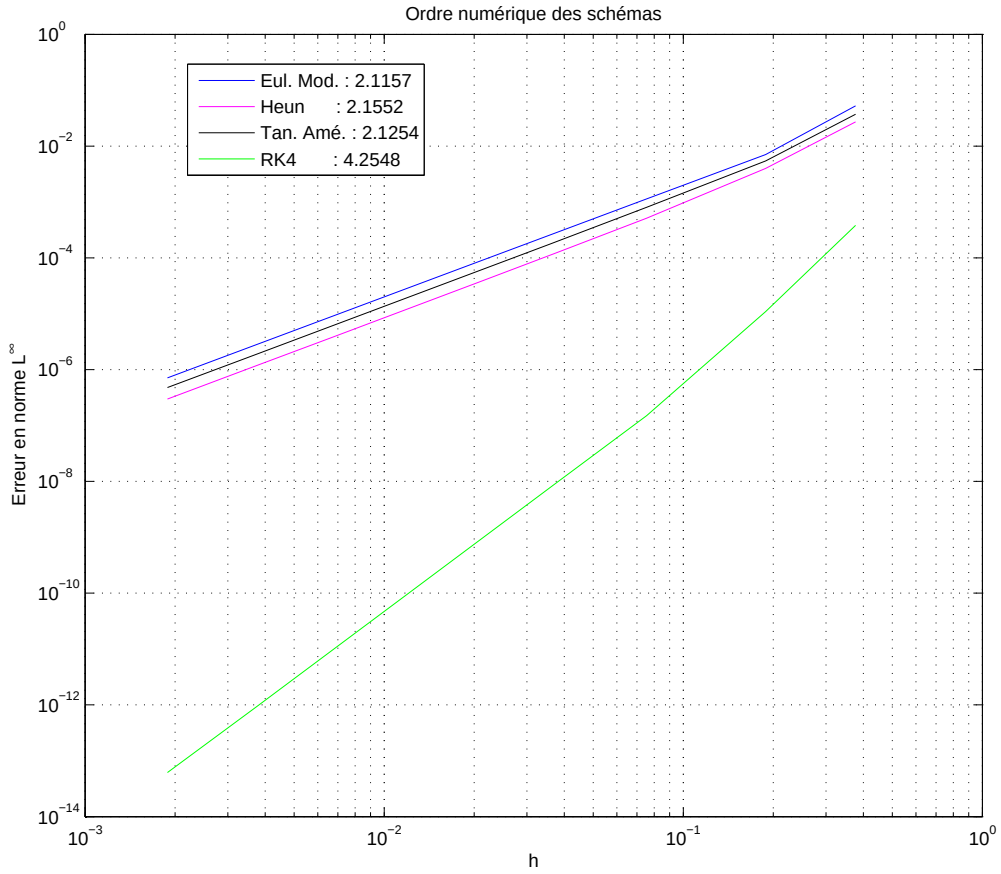

FIGURE 3 – Ordre numérique des différents schémas

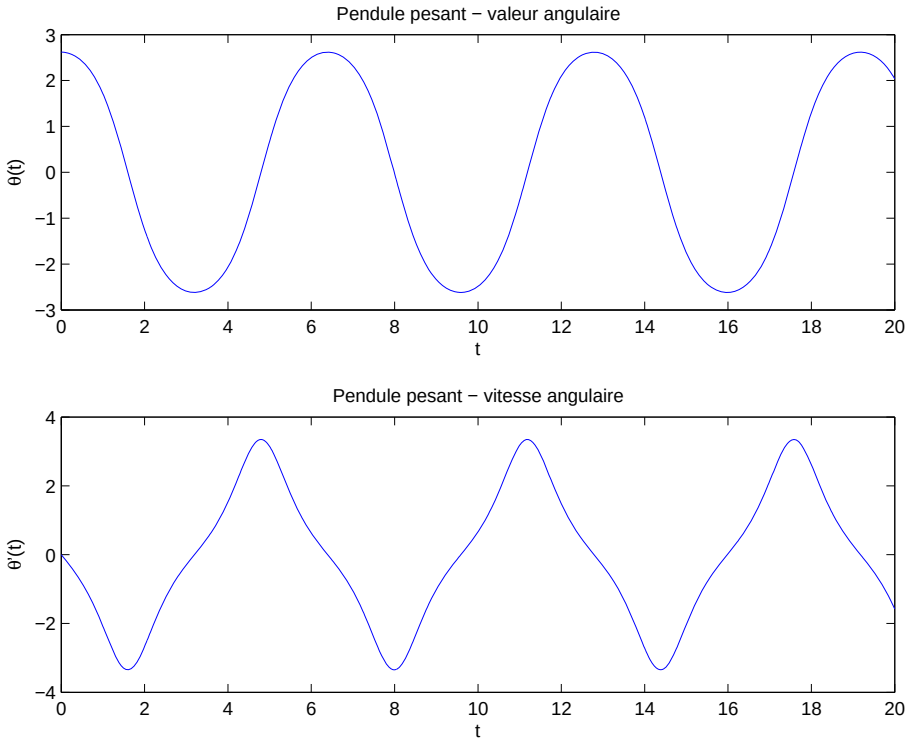

FIGURE 4 – Positions et vitesses d'un pendule sans viscosité (voir figure)

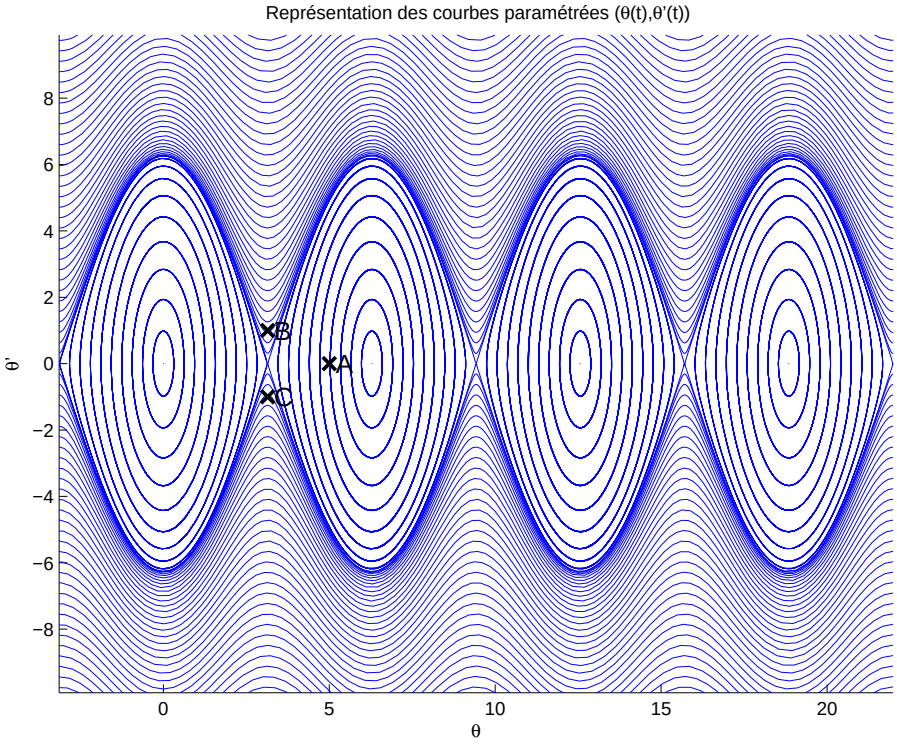

FIGURE 5 – plans de phase d'un pendule sans viscosité (voir figure)

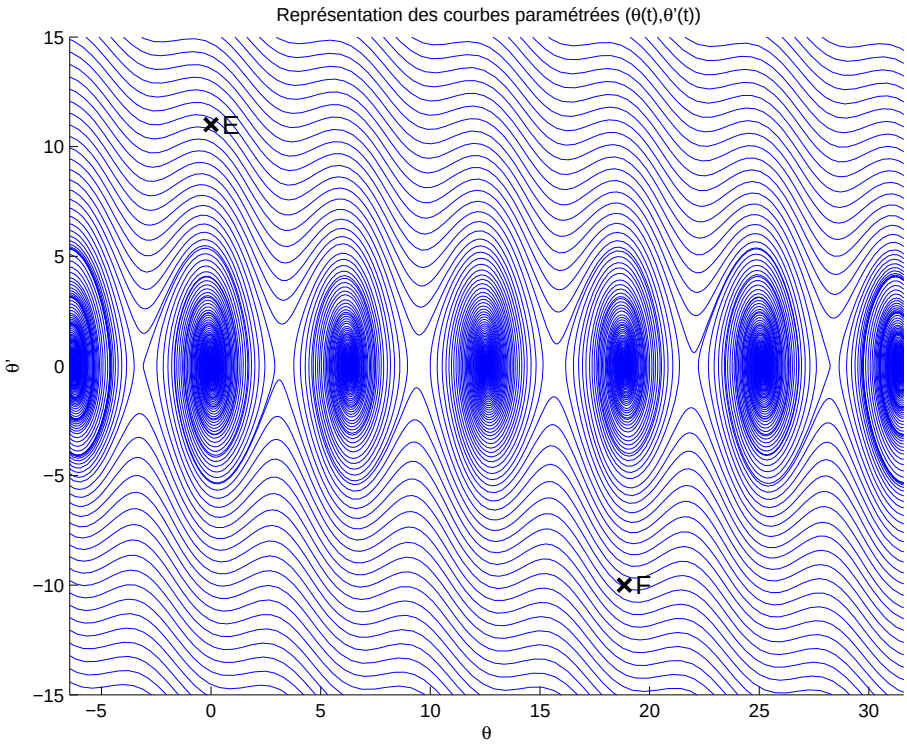

FIGURE 6 – plans de phase d'un pendule avec viscosité (voir figure)

d'executer vos programmes et d'effecter les representations graphiques. Un fichier `README` sera présent dans l'archive et décrira le but de chacun des codes que vous aurez écrits ainsi que les commandes permettant de les compiler et de les executer.

- Soutenance ?.