Étude numérique du problème de temps de crise par Lagrangien augmenté

par

Miralles Florian

1er juin 2019

Dirigé par Térence Bayen

Table des matières

1 Introduction

La théorie du contrôle analyse les propriétés des systèmes commandés, c'est-à-dire des systèmes dynamiques sur lesquels on peut agir au moyen d'un contrôle avec d'éventuelles contraintes d'état. D'un point de vue mathématique, pour modéliser ces systèmes, on peut avoir recours à des équations différentielles. Une fois l'analyse du système commandé résolue, on peut de plus vouloir passer de l'état initial à l'état final en minimisant certains critères, on parle alors de problème de contrôle optimal :

$$\inf_{u(\cdot)\in\mathcal{U}} \int_0^T l(x(t),u(t))dt \quad \text{tel que} \quad \begin{cases} \dot{x}(t) = f(x(t),u(t)), \\ x(0) = x_0, \\ x(T) \in C. \end{cases}$$

Ce mémoire a pour but d'approcher numériquement les solutions du problème d'optimisation du temps de crise dans le cadre du système de Lotka-Volterra. Le problème du temps de crise consiste à minimiser le temps pris par les solutions d'un système dynamique contrôlé, à l'extérieur d'un certain ensemble K donné. Il s'agit d'un problème non classique car $l(x,u) = \mathbf{1}_{K^c}(x)$, ce qui introduit une discontinuité par rapport à l'état x. Cet ensemble décrit des contraintes sur l'état du système. Décrivons la construction du travail (basé sur [2],[3]) :

- Tout d'abord, le problème du temps de crise n'a de sens que si les solutions sortent de l'ensemble K donné, nous verrons dans la section 2 quelle condition suffit à cela.
- Dans la section 3, nous définirons plus en détail ce qu'est le temps de crise.
- Dans la quatrième section, nous verrons les conditions d'optimalité d'un tel problème.
- Dans la section 5, nous parlerons du Lagrangien augmenté et des mises en œuvre sur un plan pratique, et la section 6 sera destinée à choisir la meilleure méthode numérique, pour la résolution de notre problème, entre celle d'Euler et celle de Runge-Kutta.
- Puis dans la dernière section, nous comparerons deux approches numériques du problème. Une approche directe et une approche reformulée.

La plus grande partie de mon travail a consisté à résoudre numériquement le problème du temps de crise lié au système de Lotka-Volterra par Lagrangien augmenté. C'est une méthode différente des autres méthodes basées sur la régularisation.

2 Description du contexte

2.1 Théorie de la viabilité

Afin d'introduire le problème du temps de crise, nous proposons de décrire le contexte dans lequel il a du sens. En effet, commençons par quelques définitions pour amener le problème.

Définition 1. (Fonction viable)
Soit $K \subset X$ non vide, X est un espace vectoriel de dimension finie,une application $x : [0,T] \longrightarrow X$ est viable dans K sur [0,T], si pour tout $t \in [0,T]$ $x(t) \in K$.

Dans la suite, nous considérons $\Omega \subset X$ un sous ensemble ouvert de X et une application régulière ($\mathcal{C}^1$) $f : \Omega \longrightarrow X$. Considérons le problème de Cauchy défini comme suit :

$$\begin{cases} x'(t) &= f(x(t)), \\ x(0) &= x_0. \end{cases} \tag{1}$$

Définition 2. (Sous-ensemble viable)
Soit $K \subset \Omega$, on dira que K est localement viable par f, si pour n'importe quel $x_0 \in K$ il existe $T > 0$ et il existe une solution $x^* : [0,T] \longrightarrow \Omega$ solution de (1) telle que x^* soit viable.

Définissons maintenant ce dont nous aurons besoin par la suite, le noyau de viabilité.

Définition 3. (Noyau de viablité de K par f)
Noté $Viab_K(f)$,il est défini comme le plus grand sous ensemble viable fermé de K par f

De manière équivalente, une façon de redéfinir le noyau est la définition qui suit :

Définition 4. Notons $x(\cdot, x_0)$ la solution de (1), le noyau de viabilité s'écrit :

$$Viab_K(f) = \{x_0 \in K, \forall t \in [0,T] \;\; x(t,x_0) \in K\}.$$

2.2 Systèmes controlés

Un système controlé est une équation différentielle qui dépend d'une fonction u supplémentaire à valeurs dans $\mathbb{R}^m$.

Définition 5. (Contrôle admissible)
Soit U un sous-ensemble fixé non vide de $\mathbb{R}^m$ et $T > 0$. Un contrôle u est admissible, si $u \in \mathcal{U}$ où $\mathcal{U} = \{u : [0,T] \longrightarrow U, u \text{ mesurable}\}$.

Soit $f : \mathbb{R}^n \times \mathbb{R}^m \longrightarrow \mathbb{R}^n$ de classe $\mathcal{C}^1$. le problème (2) est un système contrôlé :

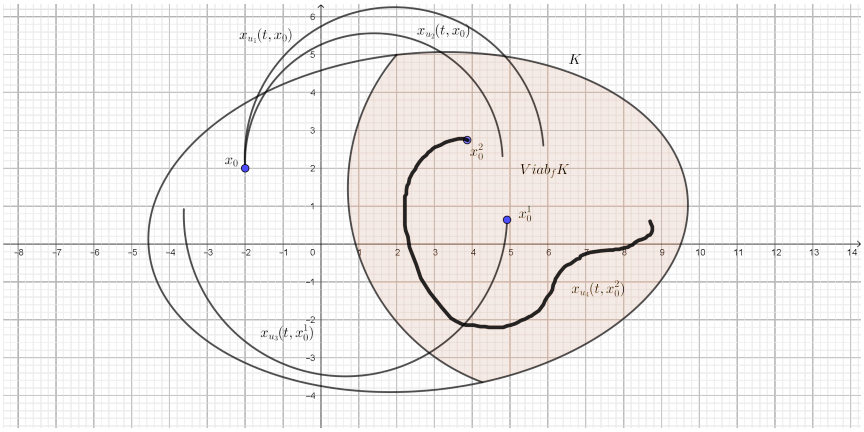
$$\begin{cases} x'(t) &= f(x(t), u(t)), \\ x(0) &= x_0. \end{cases} \tag{2}$$

Définition 6. Par un tel système contrôlé, le noyau de viabilité de l'ensemble K sous la dynamique f est défini par :

$$Viab_K(f) = \{x_0 \in K, \; \exists u \in \mathcal{U} \;\; \forall t \in [0,T] \;\; x(t,x_0) \in K\}.$$

Exemple :

Dans cet exemple nous avons représenté 4 trajectoires respectivement commandées par 4 contrôles u_1, u_2, u_3, u_4. Nous voyons sur le graphique que x_0^1 appartient au noyau de viabilité et par conséquent que la trajectoire contrôlée par u_3 reste dans K.



3 Problème du temps de crise

3.1 Définition

Soit $f : \mathbb{R}^n \times \mathbb{R}^m \longrightarrow \mathbb{R}^n$ de classe $\mathcal{C}^1$, on considère l'équation différentielle (on notera $x_u(\cdot, x_0)$ la solution).

$$\left\{ \begin{array}{rcl} \dot{x}(t) & = & f(x(t), u(t)), \\ x(0) & = & x_0. \end{array} \right. \tag{3}$$

Soit $K \subset \mathbb{R}^n$ fermé non vide, le problème du temps de crise est équivalent à minimiser le temps pris par les trajectoire solution de (3) à l'extérieur de K par rapport au contrôle admissible u.

Définition 7. (Temps de crise)
Le temps de crise est le problème d'optimisation suivant :

$$\inf_{u \in \mathcal{U}} \int_0^T \mathbf{1}_{K^c}(x_u(t, x_0))dt \ ; \ T \in \mathbb{R} \cup \{+\infty\}$$

où

$$\mathbf{1}_{K^c}(x) := \left\{ \begin{array}{r} 1 \text{ si } x \notin K, \\ 0 \text{ sinon.} \end{array} \right.$$

Dans la suite du document nous prendrons toujours T finie.

Définition 8. (Temps de saut)
Le temps de saut (ou point de croisement) de K à K^c est l'instant $t_c \in]0, T[$ pour lequel il existe $\epsilon > 0$ tel que :

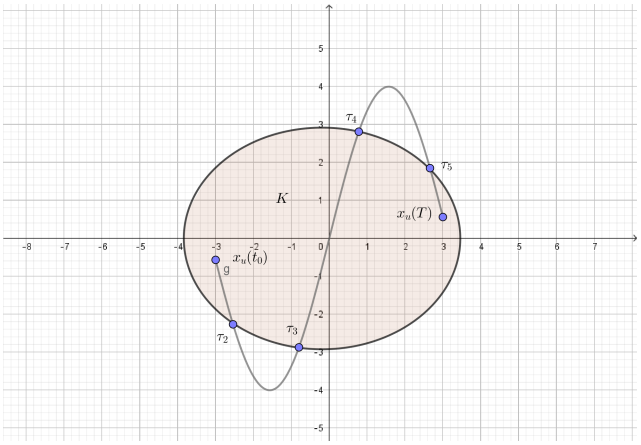
$$\forall t \in]t_c - \epsilon, t_c[, \ \ x(t) \in K,$$
$$\forall t \in]t_c, t_c + \epsilon[, \ \ x(t) \in K^c.$$

Définition 9. Un temps de saut t_c est transverse de K à K^c si

- le contrôle u associé est continu à droite et à gauche en t_c,
- et $\dot{x}(t_c) \cdot \nabla g(x(t_c)) \neq 0$.

Remarque :
On aura besoin de la définition 9 plus tard, les temps de saut transverse nous servent à garantir la stabilité des solutions, c'est à dire une petite perturbation de la trajectoire nominale aura aussi le même nombre de point de croisement transverse.
Exemple
Dans le cas ci dessous nous avons tracé la trajectoire x pour le u optimal, dans ce cas

$$\inf_{u\in\mathcal{U}} \int_0^T \mathbf{1}_{K^c}(x_u(t,x_0))dt = T - \tau_2 + \tau_3 + \tau_4 + \tau_5.$$



De la définition il en résulte la remarque suivante : l'intégrande $t \longmapsto \mathbf{1}_{K^c}(x_u(t,x_0))$ est discontinue à chaque fois qu'on passe de K à K^c.

3.2 Condition nécessaire

Dans l'exemple qui précède nous avons supposé qu'il existait bien un u, solution au problème du temps de crise. Nous allons voir justement dans cette partie les conditions nécessaires à cette affirmation.

Théorème 1. *Soit $f : \mathbb{R}^n \times \mathbb{R}^m \longrightarrow \mathbb{R}^n$*

- *f continue par rapport à x et u et de classe $\mathcal{C}^1$ par rapport à x.*
- *Il existe $c_1, c_2 > 0$ tel que $\forall x \in \mathbb{R}^n$ et $u \in \mathcal{U}$*

$$\|f(x,u)\| \leq c_1\|x\| + c_2$$

où $\|\cdot\|$ est norme euclidienne.
Si ces conditions sont réunies alors il existe une unique solution $x_u(\cdot, x_0)$ au problème :

$$\begin{cases} x'(t) &= f(x(t),u(t)) \ \ \forall t \in [0,T], \\ x(0) &= x_0. \end{cases}$$

Remarque :
Il s'agit du théorème de Cauchy-Lipschitz amélioré.

Supposons maintenant $K \subset \mathbb{R}^n$ fermé tel que $\mathring{K} \neq \emptyset$

Proposition 1. *Si :*

- *les conditions sur f du théorème [1] sont réunis.*
- *L'ensemble $\{f(x,u)|\ u \in U\}$ est compact, convexe.*
- *$U \neq \emptyset$ compact convexe de $\mathbb{R}^m$.*
- *$K \neq \emptyset$ fermé convexe de $\mathbb{R}^n$.*

Alors il existe $u^* \in \mathcal{U}$ tel que :

$$\inf_{u\in\mathcal{U}} \int_0^T \mathbf{1}_{K^c}(x_u(t,x_0))dt = \int_0^T \mathbf{1}_{K^c}(x_{u^*}(t,x_0))dt.$$

Démonstration. On renvoie à la démonstration à l'article [1] proposition 2.1 page 266 (par suite minimisante). □

4 Conditions d'optimalité

Cette partie s'appuie sur un résultat fondamental de l'article [2]. En effet, dans cet article, les auteurs ont reformulé le problème de différentes manières, afin de pouvoir démontrer le *théorème* 2. Dans la suite $K = \{x \in \mathbb{R}^n|\ g(x) \leq 0\}$.

Problème 1

$$(P) \inf_{u\in\mathcal{U}} \int_0^T \mathbf{1}_{K^c}(x_u(t,x_0))dt$$

où x_u solution de

$$\begin{cases} \dot{x}(t) &= f(x(t),u(t)) \ \ \forall t \in [0,T], \\ x(0) &= x_0. \end{cases}$$

Problème 2 Écriture du problème avec un temps de saut transverse, et avec la bijection

$$\pi_\tau : s \in [0,2] \longmapsto \begin{cases} \tau s \ \ si\ s \in [0,1], \\ (T-\tau)s + 2\tau - T \ \ si\ s \in [1,2]. \end{cases}$$

$$(P_1) \inf_{\tilde{u}\in\tilde{\mathcal{U}},\tau\in]0,T[} T-\tau \ \text{ sous la contrainte } \ g(\tilde{x}_{\tilde{u},\tau}(1)) = 0,$$

où

$$\begin{cases} \tilde{x}(s) &= x(\pi_\tau(s)), \\ \tilde{u}(s) &= u(\pi_\tau(s)), \end{cases}$$

et aussi $\tilde{x}_{\tilde{u}}$ est solution du problème de Cauchy

$$\begin{cases} \dot{\tilde{x}}(s) &= \frac{d\pi_\tau(s)}{ds} f(\tilde{x}(s), \tilde{u}(s)) \ \ \forall s \in [0,2], \\ \tilde{x}(0) &= x_0. \end{cases}$$

On se réfère à [2] pour avoir le lien entre (P) et (P_1). En effet il est montré que si (u, τ) est solution de (P) alors $(\tilde{u}, \tau)$ est solution de (P_1).

Problème 3 Notons $y = (y^{(1)}, y^{(2)}, \xi)$ où $y^{(1)}, y^{(2)} \in \mathbb{R}^n$ et $\xi \in \mathbb{R}$, $v = (u^{(1)}, u^{(2)})$ où $u^{(1)}, u^{(2)} \in \mathbb{R}^m$. Considérons maintenant

$$F(y, v) := \begin{pmatrix} \xi f(y^{(1)}, u^{(1)}) \\ (T - \xi) f(y^{(2)}, u^{(2)}) \\ 0 \end{pmatrix},$$

et

$$G(y_0, y_1) = \begin{pmatrix} y_0^{(1)} \\ y_0^{(2)} - y_1^{(1)} \\ g(y_1^{(1)}) \end{pmatrix}, \quad \begin{array}{l} y_0 = (y_0^{(1)}, y_0^{(2)}, \xi_0), \\ y_1 = (y_1^{(1)}, y_1^{(2)}, \xi_1). \end{array}$$

L'ensemble des contrôles admissibles est

$$\mathcal{V} := \{v = (u^{(1)}, u^{(2)}) : [0,1] \longrightarrow U \times U; v \ mesurable\}.$$

Et notons l'ensemble $C = \{x_0\} \times]0, T[\times \{0_{\mathbb{R}^n}\} \times \{0\}$.
Le nouveau problème s'écrit :

$$(P_2) \inf_{v \in \mathcal{V}} T - \xi \ \text{ sous la contrainte } \ G(y_v(0), y_v(1)) \in C.$$

Où y_v est solution de $\frac{dy_v}{ds}(s) = F(y(s), v(s)) \ \ s \in [0,1]$. On a également que si u est solution de (P) alors v est solution de (P_2).

Problème 4 Quant à ce dernier problème, il est utilisé explicitement dans la démonstration du théorème qui suit, il s'agit en fait du problème (P_2) reformulé dans le cas de r temps de saut transverse.

$$(P_3) \inf_{\mathbf{v} \in \mathbb{V}} \sum_{j \in [\![1, E(\frac{r+1}{2})]\!]} \xi_{2j} - \xi_{2j-1} \ \text{ sous la contrainte } \ \mathbf{G}(\mathbf{y_v}(0), \mathbf{y_v}(1)) \in C.$$

Ici $\mathbf{y_v}$ solution de

$$\frac{d\mathbf{y}}{ds}(s) = \mathbf{F}(\mathbf{y}(s), \mathbf{v}(s)).$$

Où cette fois ci $\mathbf{F}$ et $\mathbf{G}$ sont définies ainsi :

$$\mathbf{F}(\mathbf{y}, \mathbf{v}) := \begin{pmatrix} (\xi_1 - \xi_0) f(y^{(1)}, u^{(1)}) \\ \vdots \\ (\xi_{r+1} - \xi_r) f(y^{(r+1)}, u^{(r+1)}) \\ 0_{\mathbb{R}^r} \end{pmatrix} \quad \begin{array}{r} \mathbf{y} = (y^{(1)}, ..., y^{(r+1)}, \xi_1, ..., \xi_{r+1}) \\ \mathbf{v} = (u^{(1)}, ..., u^{(r+1)}) \end{array}$$

$$\mathbf{G}(\mathbf{y}_0,\mathbf{y}_1)=\begin{pmatrix} y_0^{(1)} \\ \xi_1^0 \\ g(y_1^{(1)}) \\ y_0^{(2)}-y_1^{(1)} \\ \vdots \\ \xi_r^0 \\ g(y_1^{(r)}) \\ y_1^{(r)}-y_0^{(r+1)} \end{pmatrix} \quad \mathbf{y}_i=(y_i^{(1)},...,y_i^{(r+1)},\xi_1^i,...,\xi_i^r)\ .$$

L'idée à travers ces différentes reformulations est que l'on peut construire la solution (P_3) à partir de la solution de (P). L'avantage du problème (P_3), c'est qu'il est un problème classique régulier, alors que (P) n'est pas un problème régulier puisqu'on a vue que son intégrande était discontinue.

Définition 10. (Hamiltonien)
Soit le problème de contrôle optimal :

$$\inf_{u\in\mathcal{U}} \phi(x_u) \text{ tel que } x_u \text{ est solution de (2).} \tag{4}$$

On appelle Hamiltonien associé au problème (4), la fonction :

$$\begin{cases} H & :\mathbb{R}^n\times\mathbb{R}^n\times\mathbb{R}^m\longrightarrow\mathbb{R} \\ & (x,p,u)\longmapsto p\cdot f(x,u) \end{cases}$$

Théorème 2. *(Conditions d'optimalité du premier ordre)*
Supposons que l'on ait $g:\mathbb{R}^n\longrightarrow\mathbb{R}$ de classe $\mathcal{C}^1$ telle que $K=\{x\in\mathbb{R}^n \mid g(x)\leq 0\}$
Soit $u\in\mathcal{U}$ une solution optimale au problème (P) telle que la trajectoire x_u solution de (2) possède exactement $r\in\mathbb{N}^*$ temps de sauts transverses $\tau_1<...<\tau_r$ dans $]0,T[$ tels que pour tout i impaire (resp paire) on ait τ_i est un temps de saut de K à K^c (resp de K^c à K)
Alors il existe une fonction $p:[0,T]\longrightarrow\mathbb{R}^n$ telle que :

1 p satisfait

$$\begin{cases} \dot{p}(t) &= -\nabla_x H(x_u(t),p(t),u(t)) \ \ \forall t\in[0,T], \\ p(T) &= 0, \end{cases}$$

2 le contrôle u satisfait le condition Hamiltonienne pour tout $t\in[0,T]$:

$$\nabla_u H(x_u(t),p(t),u(t))\cdot(\omega-u(t))\geq 0 \ \ \forall\omega\in U,$$

3 pour chaque temps de saut τ_i $1\leq i\leq r$:

$$p(\tau_i^+)-p(\tau_i^-)=\frac{(-1)^i+p(\tau_i^-)\cdot(\dot{x}_u(\tau_i^-)-\dot{x}_u(\tau_i^+))}{\nabla g(x_u(\tau_i))\cdot\dot{x}_u(\tau_i^+)}\nabla g(x_u(\tau_i)) \tag{5}$$

Démonstration. Les transformations vues plus haut permettent de prouver ce théorème(voir [2]). □

Remarque : Le dénominateur dans 5 est non nul car les temps de saut sont transverses. Une différence fondamentale avec le principe du maximum de Pontryagin, est que le vecteur adjoint p est discontinue sur $[0, T]$.
L'objectif de ce mémoire a été d'utiliser la reformulation du problème 1 au problème 2pour développer une nouvelle méthode pour étudier le temps de crise, plutôt que d'utiliser le théorème 2.

5 Approche par Lagrangien augmenté

5.1 Régularisée de Moreau-Yosida

Dans cette section nous ne nous contenterons pas de donner simplement la définition du Lagrangien augmenté, mais nous allons montrer qu'il est une conséquence d'une régularisation. En effet, plutôt que de le voir comme un Lagrangien auquel on a rajouté un terme de pénalisation, nous allons voir que ce terme en plus devrait être vu comme un terme de régularisation.

Définition 11. (Régularisée de Moreau-Yosida)

- Soit $J : H \longrightarrow \mathbb{R}$ une fonction convexe, où H est un espace de Hilbert. On appelle régularisée de Moreau-Yosida de J la fonction J_c définie par :

$$J_c(x) = \inf_{y \in H} \left(J(y) + \frac{1}{2c} \|y - x\|^2 \right)$$

- Pour J concave la régularisée s'écrit :

$$J_c(x) = \max_{y \in H} \left(J(y) - \frac{1}{2c} \|y - x\|^2 \right)$$

Exemple :
Prenons pour exemple $J(x) := |x|$
Soit $x \in \mathbb{R}$ fixée, on a par convexité

$$\inf_{y \in H} \left(|y| + \frac{1}{2c} |y - x|^2 \right) = \min_{y \in H} \left(|y| + \frac{1}{2c} |y - x|^2 \right)$$

Ainsi distinguons les cas,posons tout d'abord $f_x(y) = |y| + \frac{1}{2c}(y - x)^2$

- cas $y > 0$
  $\frac{df_x}{dy}(y) = 1 + \frac{1}{c}(y - x) = 0$
  $\Leftrightarrow -c = y - x$
  $\Leftrightarrow y_0 = x - c > 0 \Rightarrow x > c$
  Injectons ceci dans f_x on a
  $f_x(y_0) = x - c + \frac{1}{2c}(x - c - x)^2$
  $= x - c + \frac{c}{2}$
  $= x - \frac{c}{2}$

- cas $y < 0$
  on a cette fois ci $y_0 = x + c$ et $x < -c$ et $f_x(y_0) = -x - \frac{c}{2}$

- cas $y = 0$
  $f_x(0) = \frac{1}{2c}x^2$ pour $-c \leq x \leq c$

De là la régularisée s'écrit

$$J_c(x) = \begin{cases} -x - \frac{c}{2} \; si \; x < -c \\ \frac{x^2}{2c} \; si \; -c \leq x \leq c \\ x - \frac{c}{2} \; si \; x > c \end{cases}$$

5.2 Application au Lagrangien

Supposons $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ convexe semi-continue inférieurement et soit $K \subset \mathbb{R}^n$ un convexe fermé non vide.
Le problème

$$\min_{x \in K} f(x)$$

admet une solution. Réécrivons le problème :

$$\min_{x \in K} f(x) + \chi_K(x)$$

où

$$\chi_K(x) := \begin{cases} 0 \; si \; x \in K \\ +\infty \; sinon. \end{cases}$$

On a par définition du régularisée de Moreau-Yosida :

$$f_c(x) = \min_{y \in \mathbb{R}} \left(f(y) + \chi_K(y) + \frac{1}{2c}\|y - x\|^2 \right)$$

$$= \min_{y \in K} \left(f(y) + \frac{1}{2c}\|y - x\|^2 \right)$$

Proposition 2. *Dans ces conditions, on a :*

$$\text{argmin}_{x \in K} f(x) = \text{argmin}_{x \in \mathbb{R}} f_c(x)$$

Démonstration. Supposons $x_0 \in \text{argmin}_{x \in \mathbb{R}} f_c(x)$ puisque $f_c(x) \leq f(x)$ pour tout x alors $x_0 \in \text{argmin}_{x \in \mathbb{R}} f(x)$
L'autre sens est moins évident. □

Dans le cas sous contraintes on a :

$$\min_{x \in K} f(x) \text{ sous les contraintes } g_i(x) = 0 \;\; \forall i \in [\![1, m]\!]$$

Le Lagrangien classique s'écrit :

$$\mathcal{L}(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x)$$

Notons maintenant la fonction :

$$H(\lambda) := \min_{x \in K} \mathcal{L}(x, \lambda)$$

existe si les contraintes sont qualifiées et si f et g sont suffisamment régulières.

Proposition 3. *(concavité de H)*
Soit $t \in [0,1]$ *on a :*

$$H(t\lambda + (1-t)\lambda) \geq tH(\lambda) + (1-t)H(\lambda)$$

Démonstration. Soit $t \in [0,1]$ et $\lambda_1, \lambda_2 \in \mathbb{R}^m$

$$\begin{aligned} H(t\lambda_1 + (1-t)\lambda_2) &= \min_{x \in \mathbb{R}} \left(f(x) + t\lambda_1 \cdot g(x) + (1-t)\lambda_2 \cdot g(x)\right) \\ &= \min_{x \in \mathbb{R}} \left(tf(x) + (1-t)f(x) + t\lambda_1 \cdot g(x) + (1-t)\lambda_2 \cdot g(x)\right) \\ &\geq \min_{x \in \mathbb{R}} \left(tf(x) + t\lambda_1 \cdot g(x)\right) + \min_{x \in \mathbb{R}} \left((1-t)f(x) + (1-t)\lambda_2 \cdot g(x)\right) \\ &\geq tH(\lambda_1) + (1-t)H(\lambda_2) \end{aligned}$$

□

Ainsi le problème dual au problème initial s'écrit :

$$\max_{\lambda \in \mathbb{R}^m} H(\lambda)$$

et il s'agit d'un problème de maximisation de fonction concave
De là, pour λ^* qui maximise H, il existe x^* qui minimise $\mathcal{L}(x^*, \lambda^*)$, alors x^* solution de $\min_{x \in K} f(x)$ sous les contraintes $g_i(x) = 0$ $i \in [\![1, m]\!]$.

Maintenant, pour ce qui est du lagrangien augmenté, montrons qu'il s'agit en fait d'une conséquence de la régularisation de H.
En effet

$$H_c(p) = \max_{q \in \mathbb{R}^m} \left(H(q) - \frac{1}{2c}\|q - p\|^2\right)$$

On a que :

$$\begin{aligned} H_c(p) &= \max_{q \in \mathbb{R}^m} \left\{ \min_{x \in \mathbb{R}^n} \left(f(x) + \sum_{i=1}^m \lambda_i \cdot g_i(x)\right) - \frac{1}{2c}\|q - p\|^2 \right\} \\ &= \min_{x \in \mathbb{R}^n} \left\{ f(x) + \max_{q \in \mathbb{R}^m} \left(\sum_{i=1}^m \lambda_i g_i(x) - \frac{1}{2c}\|q - p\|^2\right) \right\} \end{aligned}$$

Or si on calcule en quelles valeurs de q, $\max_{q \in \mathbb{R}^m} \left(\sum_{i=1}^m \lambda_i \cdot g_i(x) - \frac{1}{2c}\|q - p\|^2\right)$ est atteint, on obtient le résultat suivant :
La condition d'optimalité s'écrit :

$\nabla_q \left(\sum_{i=1}^{m} \lambda_i . g_i(x) - \frac{1}{2c}\|q-p\|^2 \right) = 0$
$\Leftrightarrow g(x) - \frac{1}{c}(q-p) = 0$
$\Leftrightarrow cg(x) - q + p = 0$
$\Leftrightarrow q^* = p + cg(x)$

Ainsi en injectant q^* dans ce qui précède on a :

$$H_c(p) = \min_{x\in\mathbb{R}^n} \left(f(x) + < p + cg(x), g(x) > - \frac{1}{2c}\|cg(x)\|^2 \right)$$

par linéarité du produit scalaire

$$= \min_{x\in\mathbb{R}^n} \left(f(x) + < p, g(x) > + \frac{c}{2}\|g(x)\|^2 \right)$$

Définition 12. (Lagrangien Augmenté)
Soit $c > 0$, on appel Lagrangien Augmenté la fonction définie par :

$$\mathcal{L}_c(x,p) := f(x) + < p, g(x) > + \frac{c}{2}\|g(x)\|^2$$

De là $H_c(p) = \min_{x\in\mathbb{R}^n} \mathcal{L}_c(x,p)$

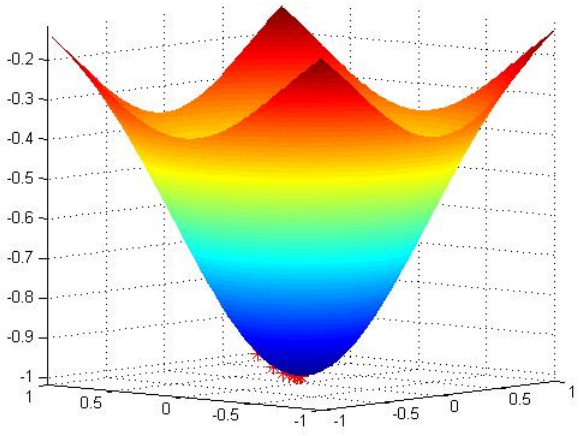
5.3 Application pratique

Nous proposons un code déterminant le minimum d'une fonction de deux variables soumis à une contrainte, en voici un exemple de fonctionnement :

<Student Version> Command Window

```
EDU>> LagrangienA(-1,1)
    'attention l existence d un minimum pour la fonction à optimiser est necessaire'
    'nous rajoutons C1 pour que l algo fonctionne'

entrez la fonction à optimiser (R2--->R) sous la syntaxe : @(x1,x2) f(x1,x2)
@(x,y) -exp(-x.^2-y.^2)
probleme sous contrainte ? 0 ou 1
1
entrez la fonction contrainte d egalite, fonction de R2 dans R de classe C1
@(x,y) x-y

ans =

   1.0e-04 *

    0.3710
   -0.3710
```

Voici le résultat graphique avec la descente de gradient marqué en point rouge :



CODE MatLab :

```
function  v=LagrangienA(varargin)
%debut
disp({'attention l existence d un minimum pour la fonction  optimiser
    est necessaire' ;'nous rajoutons C1 pour que l algo fonctionne'})
f=input('entrez la fonction  optimiser (R2-->R) sous la syntaxe : @(x1,
    x2) f(x1,x2) \n');
b=input('probleme sous contrainte ? 0 ou 1 \n');
if b==1
g=input('entrez la fonction contrainte d egalite, fonction de R2 dans R
    de classe C1 \n');
%% d but  de l'algorithme  i t ratif
lambda=lambda0;
y(1,1)=x(1);
y(1,2)=x(2);
bo=1;
while bo
    a=5;
    k=k+1;
    y(k,:)=x;
    i=0;
        while (norm(grad1La(f,g,lambda,c,x(1),x(2)),2)>tolerence)
        df=grad1La(f,g,lambda,c,x(1),x(2))';

        %armijo pour la recherche du pas
            while ((La(f,g,lambda,c,x(1),x(2))-La(f,g,lambda,c,x(1)-a*df
                (1),x(2)-a*df(2)))<(sigma*a*df*(df')))
                a=(a*(1-beta)+a*beta)/2;
            end
        x=x-a*df;
        history(k,1)=x(1);
        history(k,2)=x(2);
        i=i+1;
        K=1:i;
```

```
30
31          end
32          [X,Y]=meshgrid(-1:0.01:1);
33          Z=f(X,Y);
34
35          surf(X,Y,Z)
36          shading interp;
37          hold on;
38
39          H=f(history(:,1),history(:,2));
40          plot3(history(:,1),history(:,2),H,'*','MarkerSize',10,'
                MarkerEdgeColor','r')
41
42          pause;
43          if La(f,g,lambda,c,x(1),x(2))> La(f,g,lambda,c,y(k,1),y(k,2))
44              c=c-epsmu;
45          end
46       lambda=lambda+c*g(x(1),x(2));
47
48      bo=(abs(g(x(1),x(2)))>tolerence);
49
50 end
51 v=x;
52
53     [X,Y]=meshgrid(-1:0.01:1);
54          Z=f(X,Y);
55
56          surf(X,Y,Z)
57          shading interp;
58          hold on;
59
60          H=f(history(:,1),history(:,2));
61          plot3(history(:,1),history(:,2),H,'*','MarkerSize',10,'
                MarkerEdgeColor','r')
62
63
64
65 end
66
67 end
68
69 function La=La(f,g,lambda,c,varargin)%%Lagrangien, nombre reel, pour une
       fonction de Rn-->R
70 La=f(varargin{:})+lambda*g(varargin{:})+(c/2)*g(varargin{:})^2;
71 end
72
73 %approximation du gradient modifier d'ordre 2
74 function gf=gradf(f,varargin)
75 eps=0.001;
```

```
76 ndim=length(varargin);
77 gf(1)=(f(varargin{1}+eps,varargin{2:ndim})-f(varargin{1}-eps,varargin{2:
       ndim}))/(2*eps);
78 gf(ndim)=(f(varargin{1:ndim-1},varargin{ndim}+eps)-f(varargin{1:ndim-1},
       varargin{ndim}-eps))/(2*eps);
79 end
80
81 function gLa=grad1La(f,g,lambda,c,varargin)%%vecteur de dimension ndim
       gradient du lagrangien
82 eps=0.01;
83 gLa=gradf(f,varargin{:})+lambda*gradf(g,varargin{:})+c*gradf(g,varargin
       {:})*g(varargin{:});
84 end
```

6 Système Lotka-Volterra

6.1 Existence de Solution

Soit $u \in \mathcal{U} := \{u : [0,T] \longrightarrow [0,\bar{u}]; u \ mesurable\}$
Pour $r > 0$ et $m > 0$ fixés, le système de Lotka-Volterra se définit ainsi :
$\forall t \in [0,T]$

$$\begin{cases} \dot{x}(t) &= rx(t) - x(t)y(t) \\ \dot{y}(t) &= -my(t) + x(t)y(t) - u(t)y(t) \\ x(0) &= x_0 \\ y(0) &= y_0 \end{cases}$$

Proposition 4. *(Existence de solution)*
Pour chaque $u \in \mathcal{U}$, le système de Lotka-Volterra admet une unique solution à valeurs dans $\mathbb{R}_+ \times \mathbb{R}_+$

Démonstration. On vérifie les hypothèses du théorème [1].
Posons

$$f(x,y,u) := \begin{pmatrix} rx - xy \\ -my + xy - uy \end{pmatrix}$$

- On a que f est continue en x, y, u, en tant que polynôme, de classe $\mathcal{C}^1$, en particulier de classe $\mathcal{C}^1$ par rapport à x.
- De plus comme f est $\mathcal{C}^1$ par rapport à (x,y) alors $\exists k > 0$ et $c > 0$ tel que $\forall X = (x,y), X_1 = (x_1, y_1) \in \mathbb{R}^2$

  $$\|f(X,u) - f(X_1,u)\|_2 \leq k\|X - X_1\|_2 + c$$

  En particulier prenons $X_1 = 0$ alors $f(X_1,u) = 0$ et

  $$\|f(X,u)\|_2 \leq k\|X\|_2 + c$$

- Soit $X = (x,y) \in \mathbb{R}^2$ fixé, comme $f(X,\cdot) : u \in U \longmapsto f(X,u) \in \mathbb{R}^2$ est continue et que U est compact alors

  $$F(X) := \{f(X,u) | u \in [0,\bar{u}]\}$$

  est compact, non vide.
  En ce qui concerne la convexité, $F(X)$ est convexe en temps que produit de convexe de $\mathbb{R}$.

□

Un autre théorème qui a sa place dans cette partie, mais dont nous nous servirons pas forcément, est le suivant :

Théorème 3. *(Périodicité)*
Les solutions du système de Lotka-Volterra, pour $u \equiv$ constante, sont périodique.

Démonstration. Nous renvoyons au travail de Jean-Pierre-Françoise sur le sujet, qui utilise le théorème de Poincaré-Bendixon, pour le démontrer. □

6.2 Méthode d'Euler explicite

Dans ce qui va suivre, on s'intéressera uniquement à la résolution numérique des équations de Lotka-Volterra, à partir d'ici nous pencherons plus sur des problèmes de résolutions numériques d'équations différentielles ordinaires.
Reprenons la fonction f définie plus haut :

$$f : ((x, y), u) \in \mathbb{R}^2 \times \mathbb{R} \longmapsto \begin{pmatrix} rx - xy \\ -my + xy - uy \end{pmatrix} \in \mathbb{R}^2$$

Une solution discrète approximative d'ordre 1, peut être définie de la sorte :

$$(x_{n+1}, y_{n+1}) = (x_n, y_n) + h_n f((x_n, y_n), u(t_n))$$

Où t_n est une subdivision de l'intervalle de temps $0 = t_0 < t_1 < \cdots < t_n = T$ et $h_n = t_{n+1} - t_n$
Il s'agit de la méthode explicite d'Euler. Nous allons voir que dans le cas du système de Lotka-Volterra, cette méthode conduit à des solutions non périodique. Or d'après un des théorèmes précédent, nous savons qu'elles doivent l'être. Lors des applications numériques nous prendrons une subdivision régulière telle que h_n =constante.

6.3 Méthode de Runge-Kutta d'ordre 4

Pour cette méthode, le schéma est bien différent, en effet pour chaque itération, on calcule les valeurs des poids de la quadrature :

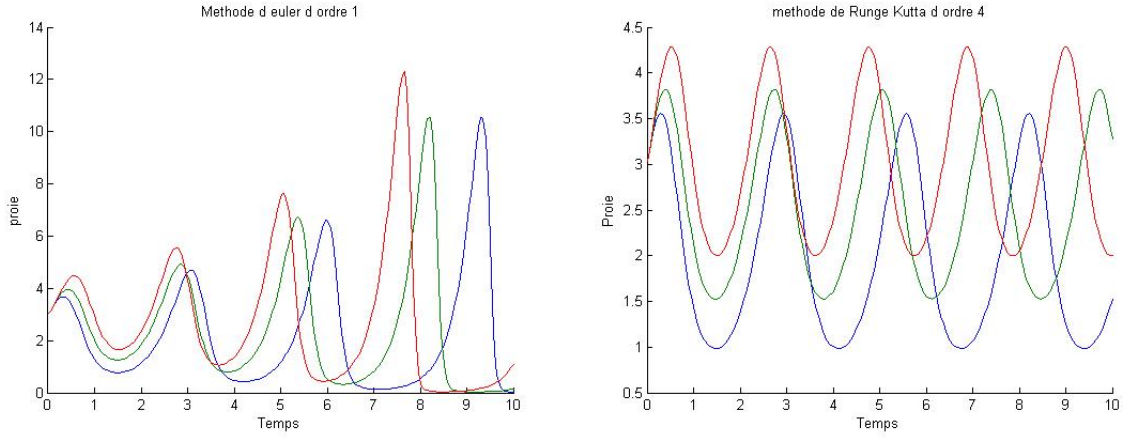
$$\begin{aligned} a_1 &= f((x_n, y_n), u(t_n)) \\ a_2 &= f((x_n, y_n) + \tfrac{h}{2} a_1, u(t_n + \tfrac{h}{2})) \\ a_3 &= f((x_n, y_n) + \tfrac{h}{2} a_2, u(t_n + \tfrac{h}{2})) \\ a_4 &= f((x_n, y_n) + h a_3, u(t_n + h)) \end{aligned}$$

$$(x_{n+1}, y_{n+1}) = (x_n, y_n) + \frac{h}{6} (a_1 + 2a_2 + 2a_3 + a_4)$$

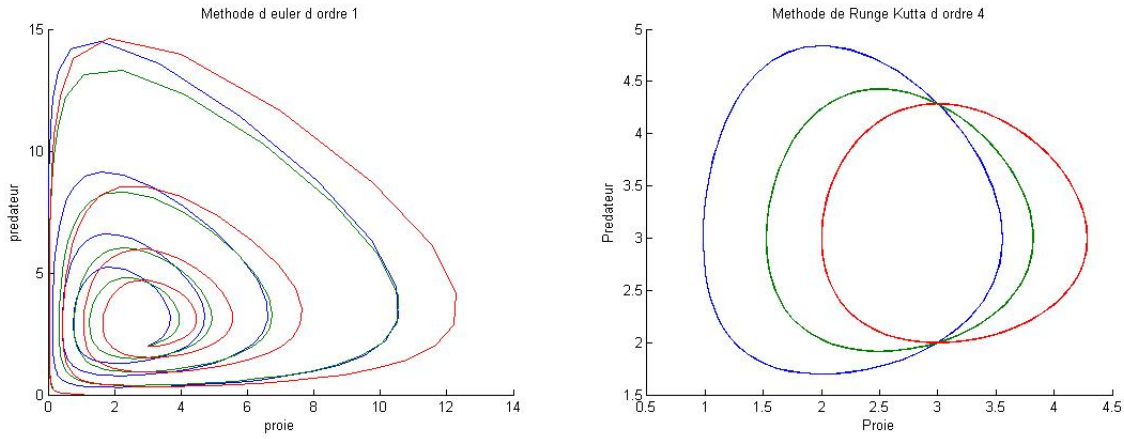
Ce schéma amène à une approximation de la solution, à l'ordre 4.

6.4 Résolution Numérique cas pratique

Nous proposons ci-dessous un algorithme visant à comparer ces deux schémas vus plus haut. En voici les résultats :



(6)



(7)

Dans les exemples ci dessus nous avons tracé l'évolution des proies (5) pour trois contrôles distincts, à gauche : résultat obtenu par la méthode d'Euler, à droite par la méthode de Runge-Kutta. En (6) les deux figures sont les trajectoires solution du système de Lotka, prise pour trois contrôles différents à savoir $u \equiv 0, u \equiv 0.5$ et $u \equiv 1$. Nous voyons clairement la non périodicité de la solution x, et des solutions x, y sur le schéma d'Euler.

ALGORITHME :

```
function LotkaVolterra
%initialisation
T=10;
N=200;
h=T/(N+1);
r=3;
mp=1;
m=2;
t=0:h:T;

%choix du controle
%u=input('entrez le controle R-->R\n');
function u=u(x)
    u=l;
    end
%methode d'euler
for l=0:0.5:1

     x(1)=3;
     y(1)=2;
         for j=1:N+1
             x(j+1)=x(j)+h*(r*x(j)-mp*x(j)*y(j));
             y(j+1)=y(j)+h*(-m*y(j)+x(j)*y(j)-u(t(j+1))*y(j));

         end
figure(1)
hold on;
plot(x,y)
title('Methode d euler d ordre 1');
xlabel('proie');
ylabel('predateur');
hold off;

figure(2)
hold all;
title('Methode d euler d ordre 1');
plot(t,x)
xlabel('Temps');
ylabel('proie');
hold off;
%methode de Runge-Kutta
X(:,1)=[x(1);y(1)];%vecteur colonne
for i=1:N+1

    k1=f(X(:,i),u(t(i+1)));
    k2=f(X(:,i)+(h/2)*k1,u(t(i+1)+(h/2)));
    k3=f(X(:,i)+(h/2)*k2,u(t(i+1)+(h/2)));
```

```
48      k4=f(X(:,i)+h*k3,u(t(i+1)+h));
49
50      X(:,i+1)=X(:,i)+(h/6)*(k1+2*k2+2*k3+k4);
51 end
52 figure(3)
53 hold all;
54 plot(t,X(1,:))
55 title('methode de Runge Kutta d ordre 4');
56 xlabel('Temps');
57 ylabel('Proie');
58 hold off;
59
60 figure(6)
61 hold all;
62 plot(t,X(2,:))
63 title('methode de Runge Kutta d ordre 4');
64 xlabel('Temps');
65 ylabel('Predateur');
66 hold off;
67
68 figure(4)
69 hold all;
70 plot(X(1,:),X(2,:))
71 title('Methode de Runge Kutta d ordre 4');
72 xlabel('Proie');
73 ylabel('Predateur');
74 hold off;
75
76 [s,v,w]=LotkaChangVar(@u,3,2);
77 figure(5)
78 hold on;
79 plot(v,w)
80 hold off;
81 end
82
83 function f=f(x,u)%vecteur colonne
84 f=[r*x(1)-mp*x(2)*x(1); -m*x(2)+x(1)*x(2)-u*x(2)];
85 end
86
87
88 end
```

7 Temps de crise lié au système de Lotka-Volterra

7.1 Première approche numérique

Ici le problème est de trouver un contrôle visant à minimiser le temps pendant lequel les proies sont en danger, clairement on dira qu'une espèce est en danger lorsque $x(t) < \bar{x}$.
Nous allons nous restreindre à l'ensemble des contrôles admissibles :

$$\mathcal{U} = \{u : [0,T] \longrightarrow \{0,\bar{u}\} |\ u\ mesurable\}.$$

Pour un temps de saut, le problème s'écrit alors :

$$\min_{u\in\mathcal{U}} T - \tau \quad \text{telleque} \quad x_u(\tau) = \bar{x}$$

où (x_u, y_u) solution de

$$\begin{cases} \dot{x}(t) &= rx(t) - x(t)y(t) \\ \dot{y}(t) &= -my(t) + x(t)y(t) - u(t)y(t) \\ x(0) &= x_0 \\ y(0) &= y_0 \end{cases}$$

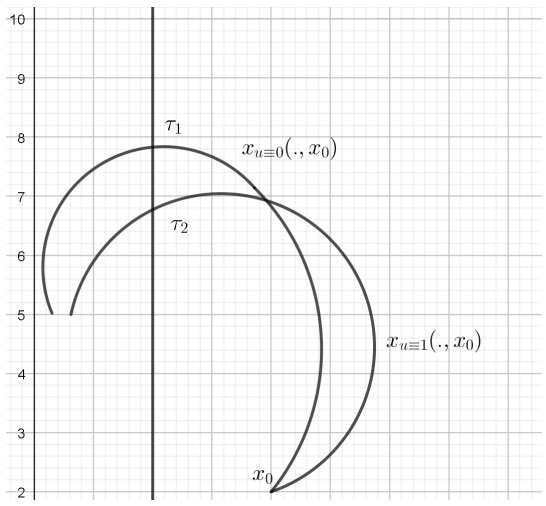
Existence de solution au problème
Nous avons vu dans la partie précédente que les conditions sur f et F sont réunies pour avoir une solution X_u au système. D'après la proposition [3] il nous reste à montrer que $U \neq \emptyset$ compact, convexe de $\mathbb{R}$, et comme $K = \{(x,y) \in \mathbb{R}^2 |\ x \geq \bar{x}\}$ est un fermé convexe de $\mathbb{R}^2$, de plus $U = \{0,1\}$ alors compact convexe non vide de $\mathbb{R}$. Ceci entraine qu'il existe bien un u^* tel que

$$\inf_{u\in\mathcal{U}} T - \tau = T - \tau_{u^*}$$

Dans la suite nous fixerons $\bar{u} = 1$, remarquons qu'en fait $\int_0^T \mathbf{1}_{K^c}(x_u(t,x_0))dt = T - \tau$, où τ est le point de croisement de K avec la trajectoire. Par exemple, pour le problème

$$\min_{u\in\{u_1:t\in[0,T]\longrightarrow 1; u_2:t\in[0,T]\longrightarrow 0\}} T - \tau$$

On a schématiquement :



L'optimum est atteint en $u \equiv 1$ et $\tau_1 = 1.112$

Une façon de procéder est de discrétiser l'ensemble de fonction $\mathcal{U}$, de telle sorte à obtenir un ensemble fini de contrôle, que nous noterons $\mathcal{U}_\epsilon$
Cet ensemble sera constitué des fonctions suivantes :
$u_1 \equiv 0$
$u_2 \equiv 1$
$u_3^i \equiv \begin{cases} 1 \text{ si } t \in]t_i, T] \\ 0 \text{ sinon} \end{cases}$ Pour $t_i = \frac{(i-1)}{N} T$

$$u_4^i \equiv \begin{cases} 0 \text{ si } t \in [0, t_i[\\ 1 \text{ si } t \in [t_i, t_{i+1}[\\ 0 \text{ sinon} \end{cases}$$

$$u_5^i \equiv \begin{cases} 1 \text{ si } t \in [0, t_i[\\ 0 \text{ sinon} \end{cases}$$

$$u_6^i \equiv \begin{cases} 1 \text{ si } t \in [0, t_i[\\ 0 \text{ si } t \in [t_i, t_{i+1}[\\ 1 \text{ sinon} \end{cases}$$

Algorithme

(1) pour chaque $u \in \mathcal{U}_\epsilon$

- calculer (x_u, y_u) en tout point solution du système Lotka-Volterra ;
- $\tau_k = x_u^{-1}(\bar{x})$ si $x_u^{-1}(\bar{x})$ existe ;
- k=k+1

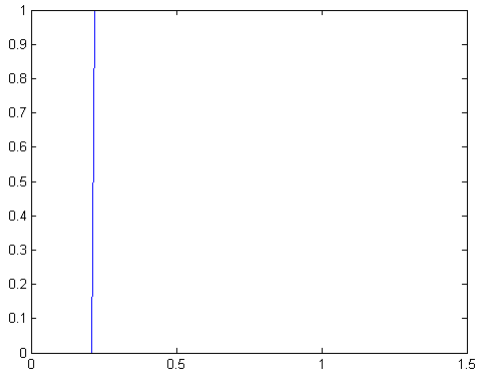
(2) Trouver (u^*, τ^*) tel que $\min_{\tau \in \{\tau_1, \ldots, \tau_k\}} T - \tau = T - \tau^*$ et $x_{u^*}(\tau^*) = \bar{x}$

Fin

Application Pratique :
Pour $N = 200$, résultat de l'optimisation du temps de crise, basé sur l'ensemble des contrôles décrits ci-dessus :

$$\inf_{u \in \mathcal{U}_\epsilon} T - \tau \simeq T - 1.20 = 0.3,$$

atteint pour le contrôle $u \in \mathcal{U}_\epsilon$ ci dessous :



defini par :

$$u^*(t) := \begin{cases} 0 & \text{si } t \in [0, 0.23] \\ 1 & \text{sinon} \end{cases}$$

Remarques :
On remarque alors que cette méthode n'est applicable qu'en nombre fini de contrôles admissibles, ce qui pose problème quant au temps de calcul de l'algorithme. En effet supposons que nous ayons eu à générer des contrôles tels que :

$$\begin{gathered} u(t_0) = u_0 \in \{0, \cdots, \tfrac{(i-1)}{N}\bar{u}, \cdots, \bar{u}\}, \\ \vdots \\ u(t_N) = u_N \in \{0, \cdots, \tfrac{(i-1)}{N}\bar{u}, \cdots, \bar{u}\}, \end{gathered}$$

pour $N = 200$, nous aurions du générer $(N+1)^{N+1}$ contrôles, ce qui vaut approximativement 10^{460} possibilités. Il est clair que nous n'aurions pas pu terminer le calcul.
Cependant par le principe du maximum de Pontryagin, on sait que $u^*(t) \in \{0, 1\}$ pour tout $t \in [0, T]$ (sauf d'éventuels arcs singulier) et que le nombre de commutation est peu élevé (les instants où $u(.)$ passe de 0 à 1 ou de 1 à 0, ce qui pourrait réduire la complexité.

7.2 Seconde Approche numérique

Tout d'abord nous considérons toujours le cas avec un temps de saut, ainsi réécrivons le problème (P).
Soit le changement de temps π_τ, pour $\tau \in [0, T]$, définie comme suit :

$$\pi_\tau(s) := \begin{cases} \tau s \quad \text{si } \ s \in [0,1], \\ (T-\tau)s + 2\tau - T \quad \text{si } \ s \in [1,2]. \end{cases}$$

$\Rightarrow$

$$\frac{d\pi_\tau}{ds}(s) = \begin{cases} \tau \quad \text{si } \ s \in [0,1], \\ (T-\tau) \quad \text{si } \ s \in [1,2]. \end{cases}$$

Posons, pour $u \in \mathcal{U}$

$$\begin{cases} \tilde{u}(s) = u(\pi_\tau(s)), \\ \tilde{x}(s) = x(\pi_\tau(s)), \\ \tilde{y}(s) = y(\pi_\tau(s)), \end{cases}$$

et

$$\tilde{\mathcal{U}} = \{u : [0,2] \longrightarrow U, u \ \text{ mesurable}\}, \quad \tilde{X}(s) = \begin{pmatrix} \tilde{x}(s) \\ \tilde{y}(s) \end{pmatrix}.$$

Ainsi le nouveau problème de Cauchy s'écrit :

$$(\tilde{C}) \quad \begin{cases} \frac{\tilde{X}}{ds}(s) = \frac{d\pi_\tau}{ds}(s) f(\tilde{X}(s), \tilde{u}(s)) \quad s \in [0,2], \\ \tilde{X}(0) = X(0). \end{cases}$$

Le problème de contrôle optimal devient :

$$(\tilde{P}) \quad \inf_{\tilde{u} \in \tilde{\mathcal{U}}} T - \tau \quad \text{sous la contrainte } \ \tilde{x}_{\tilde{u},\tau}(1) = \bar{x},$$

où $(\tilde{x}_{\tilde{u},\tau}, \tilde{y}_{\tilde{u},\tau})$ satisfait $(\tilde{C})$.
Maintenant posons :

$$\tilde{\nu} = (\tilde{x}_0, \cdots, \tilde{x}_N, \tilde{y}_0, \cdots, \tilde{y}_N, \tilde{u}_0, \cdots, \tilde{u}_N, \tilde{\bar{u}}_0, \cdots, \tilde{\bar{u}}_N),$$

et

$$\nu = (x_0, \cdots, x_N, y_0, \cdots, y_N, u_0, \cdots, u_N, \bar{u}_0, \cdots, \bar{u}_N).$$

Ensuite définissons $c' : \mathbb{R}^{4(N+1)} \longrightarrow \mathbb{R}^{2N+2}$, définie par :

$$c'(\nu) := \begin{pmatrix} \tilde{x}_0 - x_{\text{initial}} \\ \tilde{y}_0 - y_{\text{initial}} \\ \tilde{x}_1 - \tilde{x}_0 - \frac{\tau h}{6}\left(a_{1,0}^{(1)} + 2a_{2,0}^{(1)} + 2a_{3,0}^{(1)} + a_{4,0}^{(1)}\right) \\ \vdots \\ \tilde{x}_p - \tilde{x}_{p-1} - \frac{\tau h}{6}\left(a_{1,p-1}^{(1)} + 2a_{2,p-1}^{(1)} + 2a_{3,p-1}^{(1)} + a_{4,p-1}^{(1)}\right) \\ \tilde{x}_{p+1} - \tilde{x}_p - \frac{(T-\tau) h}{6}\left(a_{1,p}^{(1)} + 2a_{2,p}^{(1)} + 2a_{3,p}^{(1)} + a_{4,p}^{(1)}\right) \\ \vdots \\ \tilde{x}_N - \tilde{x}_{N-1} - \frac{(T-\tau) h}{6}\left(a_{1,N-1}^{(1)} + 2a_{2,N-1}^{(1)} + 2a_{3,N-1}^{(1)} + a_{4,N-1}^{(1)}\right) \\ \tilde{y}_1 - \tilde{y}_0 - \frac{\tau h}{6}\left(a_{1,0}^{(1)} + 2a_{2,0}^{(1)} + 2a_{3,0}^{(1)} + a_{4,0}^{(1)}\right) \\ \vdots \\ \tilde{y}_p - \tilde{y}_{p-1} - \frac{\tau h}{6}\left(a_{1,p-1}^{(1)} + 2a_{2,p-1}^{(1)} + 2a_{3,p-1}^{(1)} + a_{4,p-1}^{(1)}\right) \\ \tilde{y}_{p+1} - \tilde{y}_p - \frac{(T-\tau) h}{6}\left(a_{1,p}^{(1)} + 2a_{2,p}^{(1)} + 2a_{3,p}^{(1)} + a_{4,p}^{(1)}\right) \\ \vdots \\ \tilde{y}_N - \tilde{y}_{N-1} - \frac{(T-\tau) h}{6}\left(a_{1,N-1}^{(1)} + 2a_{2,N-1}^{(1)} + 2a_{3,N-1}^{(1)} + a_{4,N-1}^{(1)}\right) \end{pmatrix}$$

Où pour tout $k \in [\![0, N-1]\!]$:

$$\begin{cases} a_{1,k} = f((\tilde{x}_k, \tilde{y}_k), u_k), \\ a_{2,k} = f((\tilde{x}_k, \tilde{y}_k) + \frac{h}{2} a_{1,k}, \bar{u}_{k+1}), \\ a_{3,k} = f((\tilde{x}_k, \tilde{y}_k) + \frac{h}{2} a_{2,k}, \bar{u}_{k+1}), \\ a_{4,k} = f((\tilde{x}_k, \tilde{y}_k) + h a_{3,k}, u_{k+1}). \end{cases}$$

Ensuite définissons la contrainte $c : \mathbb{R}^{4(N+1)} \longrightarrow \mathbb{R}^{2N+2}$, définie par :

$$c(\nu) := \begin{pmatrix} x_0 - x_{\text{initial}} \\ y_0 - y_{\text{initial}} \\ x_1 - x_0 - \frac{h}{6}\left(a_{1,0}^{(1)} + 2a_{2,0}^{(1)} + 2a_{3,0}^{(1)} + a_{4,0}^{(1)}\right) \\ \vdots \\ x_N - x_{N-1} - \frac{h}{6}\left(a_{1,N-1}^{(1)} + 2a_{2,N-1}^{(1)} + 2a_{3,N-1}^{(1)} + a_{4,N-1}^{(1)}\right) \\ y_1 - y_0 - \frac{h}{6}\left(a_{1,0}^{(2)} + 2a_{2,0}^{(2)} + 2a_{3,0}^{(2)} + a_{4,0}^{(2)}\right) \\ \vdots \\ y_N - y_{N-1} - \frac{h}{6}\left(a_{1,N-1}^{(2)} + 2a_{2,N-1}^{(2)} + 2a_{3,N-1}^{(2)} + a_{4,N-1}^{(2)}\right) \end{pmatrix}$$

Où pour tout $k \in [\![0, N-1]\!]$:

$$\begin{cases} a_{1,k} = f((x_k, y_k), u_k), \\ a_{2,k} = f((x_k, y_k) + \frac{h}{2} a_{1,k}, \bar{u}_{k+1}), \\ a_{3,k} = f((x_k, y_k) + \frac{h}{2} a_{2,k}, \bar{u}_{k+1}), \\ a_{4,k} = f((x_k, y_k) + h a_{3,k}, u_{k+1}). \end{cases}$$

Le travail n'est pas encore fini, en effet si l'on décidait de faire tourner le code avec seulement cette contrainte, il ne s'arrêterait jamais. En effet, il faut encore une contrainte sur τ, sans quoi à chaque

itérations il ne cesserait d'augmenter pour diminuer $T-\tau$. Cependant, d'après la section [3] :
Si $u^* \in \mathcal{U}$ est optimal et que le problème est contraint à un temps de saut alors :

$$\inf_{u\in\mathcal{U}} \int_0^T \mathbf{1}_{K^c}(x_u(t,x_0))dt = \int_0^T \mathbf{1}_{K^c}(x_{u^*}(t,x_0))dt = T-\tau^*.$$

Ainsi il en découle la condition d'optimalité sur τ :

$$\int_0^T \mathbf{1}_{K^c}(x_{u^*}(t,x_0))dt - (T-\tau^*) = 0.$$

De là posons :

$$\sigma(x_0,\cdots,x_N) := \sum_{i=0}^{N+1} h\mathbf{1}_{\{a\in\mathbb{R}|\ a\leq\bar{x}\}}(x_i).$$

Il est clair que $\sigma(x_0,\cdots,x_N)$ est une approximation de $\int_0^T \mathbf{1}_{K^c}(x_u(t,x_0))dt$. La contrainte sur τ s'écrit alors :

$$\sigma(x_0,\cdots,x_N) - T + \tau = 0.$$

Or une condition d'optimalité du Lagrangien nous dit que $\nabla_\nu \mathcal{L}(\nu^*,\lambda,\mu) = 0$ ce qui signifie que σ doit être de classe $\mathcal{C}^1$, ce qui n'est pas le cas.

Définition 13. (Enveloppe de Moreau)
Soit $V \subset \mathbb{R}^n$ convexe fermé non vide et $\epsilon > 0$.
On appelle enveloppe de Moreau la fonction $e_\epsilon : \mathbb{R}^n \longrightarrow \mathbb{R}_+$ définie par :

$$e_\epsilon(x) := \frac{1}{2\epsilon} d(x,V)^2.$$

Remarque :
Ici $d(\cdot,V) : x \in \mathbb{R}^n \longmapsto \inf_{y\in V} \|x-y\|$, dans notre cas $V = [\bar{x},+\infty[$ et :

$$d(\cdot,V) : x \in \mathbb{R} \longmapsto \inf_{y\in V} |x-y|.$$

A présent considérons :

$$\begin{cases} \gamma : \mathbb{R} \longrightarrow \mathbb{R} \\ \quad v \longmapsto 1-e^{-v} \end{cases}$$

Soit $x \in \mathbb{R}^n$, on a :
si $x \in V$ alors :

$$e_\epsilon(x) = 0 \quad \Rightarrow \quad \lim_{\epsilon\to 0} e_\epsilon(x) = 0,$$

et si $x \notin V$:

$$\lim_{\epsilon\to 0} e_\epsilon(x) = +\infty.$$

Dès lors comme γ est continue :

$$\lim_{\epsilon\to 0} \gamma(e_\epsilon(x)) = \mathbf{1}_{V^c}(x).$$

Maintenant que nous avons construit une fonction qui régularise l'indicatrice d'un ensemble, appliquons cela à notre ensemble $\{a \in \mathbb{R} | a \geq \bar{x}\}$, et régularisons ensuite σ. Ainsi, pour $\epsilon > 0$, posons :

$$\sigma_\epsilon(x_0,\cdots,x_N) = \sum_{i=0}^{N} h\gamma(e_\epsilon(x_i)),$$

on a par sommation finie de limite finie :

$$\sigma_\epsilon(x_0, \cdots, x_N) \underset{\epsilon \to 0}{\longrightarrow} \sigma(x_0, \cdots, x_N).$$

Enfin, pour $\epsilon > 0$ fixé, σ_ϵ est de classe $\mathcal{C}^1$ comme sommation finie de composition de fonction $\mathcal{C}^1$. Définissons $\mathbf{v} = (\nu, \tilde{\nu}, \tau)$ et la nouvelle contrainte :

$$C(\mathbf{v}) = \begin{pmatrix} c(\nu) \\ c'(\tilde{\nu}) \\ \tilde{x}_p - \bar{x} \\ \sigma_\epsilon(x_0, \cdots, x_N) - (T - \tau) \end{pmatrix}$$

Ainsi considérons le problème $(\tilde{P}_1)$, défini comme suit :

$$\inf_{\mathbf{v} \in \mathbb{R}^{8(N+1)+1}} T - \tau \quad \text{sous la contrainte} \quad C(\mathbf{v}) = 0$$

Remarque : On s'est ramené à un problème classique d'optimisation sous contrainte d'égalité.

Proposition 5. *(Existence de solution)*
Le problème $(\tilde{P}_1)$ admet une solution $(\nu^*, \tilde{\nu}^*, \tau^*)$.

Démonstration. Soit la fonction : $(\nu, \tilde{\nu}, \tau) \in \mathbb{R}^{8(N+1)+1} \longrightarrow T - \tau \in \mathbb{R}$ est continue en tout point de l'espace de départ donc semi-continue inférieurement et coercive. De plus, on a que c et c' sont continues et $\tau \in [0, T]$ alors l'ensemble de contrainte est fermé. D'après un théorème d'existence de solution en dimension finie, le problème admet une solution. □

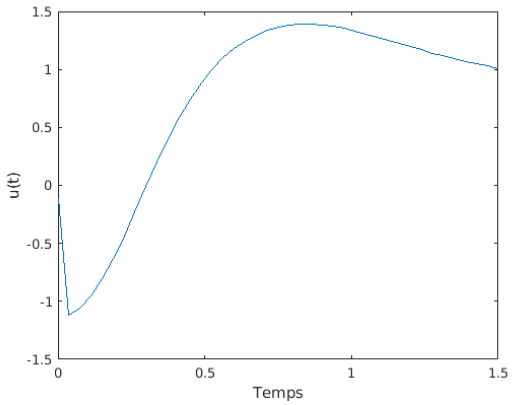
De là,comme nous l'avons vue dans la partie 5 (Lagrangien augmenté), la méthode numérique s'écrira :

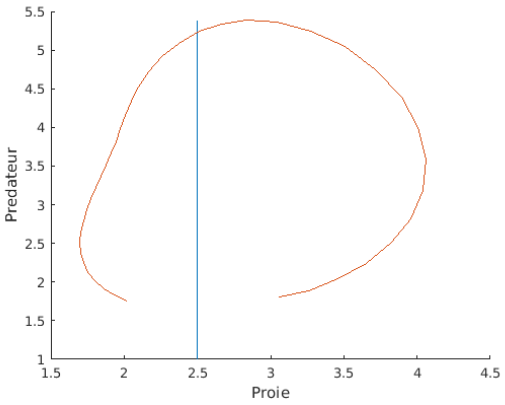
Début Algorithme Soit $(x_0, y_0) \notin Viab_K(f)$
ϵ fixé
$\lambda_0 \in \mathbb{R}^{4(N+1)+2}$
$\mu_0 \in \mathbb{R}_+^*$
$\mathbf{v}_0 \in \mathbb{R}^{8(N+1)+1}$
$k = 0$

Tant que $\|C(\mathbf{v})\| > \epsilon$

- Trouver :
$$\mathbf{v}_{k+1} = \text{argmin}_{\mathbf{v}_k \in \mathbb{R}^{8(N+1)+1}} \mathcal{L}(\mathbf{v}_k, \lambda_k, \mu_k).$$
- Si $\mathcal{L}(\mathbf{v}_{k+1}, \lambda_k, \mu_k) > \mathcal{L}(\mathbf{v}_k, \lambda_k, \mu_k)$ alors :
$$\mu_{k+1} = \epsilon \mu_k,$$
sinon
$$\mu_{k+1} = \mu_k$$
fin si.
-
$$\lambda_{k+1} = \lambda_k + \mu_{k+1} C(\mathbf{v}_{k+1}),$$

fin tant que.
Fin algorithme.
Résultat numérique Voici le résultat trouvé pour N=40, $\epsilon = 0.01$: $\tau^* = 1.26$ et u optimal tracée à gauche ci dessous et la trajectoire nominale tracé à droite.





8 Conclusion

En conclusion, l'approche du temps de crise par Lagrangien augmenté est très couteuse en temps de calcul, que ce soit la première (section 7.1) ou la seconde méthode utilisée (section 7.2). De plus, il semblerait que la seconde méthode nous donne un meilleurs temps de crise 0.3 pour la méthode 1 et 0.24 pour la seconde méthode.

Quoiqu'il en soit, dans la référence [3], l'étude numérique du modèle proie prédateur y est réalisée, mais cette fois ci la méthode numérique s'appuie sur les conditions d'optimalités dont nous avons parlé dans la section 3. Par ailleurs on a vu qu'il était nécessaire d'avoir une hypothèse de temps de saut transverse pour ces conditions d'optimalités. Cependant nous n'avons pas eu recours à une telle hypothèse pour garantir la solution du problème par Lagrangien, il semblerait que se soit un avantage. En rafinant les codes à la fois en utilisant les informations du PMP pour la méthode 1, ou en améliorant la prise en compte de contraintes sur la méthode 2, il serait possible de traiter le cas plus général de $n \geq 2$ points de croisement.

9 Annexe

Algorithme méthode 1

```
function Methode1P
[sortie,arr]=geneU(2);
topt=zeros(1,arr);

while (k<=arr)
    [U(:,k),rien]=geneU(k);
        %calcul de la solution x_u
        X(:,1)=[x0;y0];
        x(1,k)=X(1,1);
                y(1,k)=X(2,1);
```

```
11
12          for i=1:N
13                  k1=f(X(:,i),U(i,k));
14                  k2=f(X(:,i)+(h/2)*k1,U(i,k));
15                  k3=f(X(:,i)+(h/2)*k2,U(i,k));
16                  k4=f(X(:,i)+h*k3,U(i,k));
17
18                  X(:,i+1)=X(:,i)+(h/6)*(k1+2*k2+2*k3+k4);%h*f(X(:,i),u(i))
                        ;
19                  x(i+1,k)=X(1,i+1);
20                  y(i+1,k)=X(2,i+1);
21
22          end
23          %calcul de x^-1(xbar)
24          q=1;
25          while (abs(x(q,k)-xbar)>eps*5)&&(q<N+1)
26              q=q+1;
27          end
28
29          if q==N+1
30                  k=k+1;
31          else
32              topt(k)=t(q);
33              k=k+1;
34          end
35 k
36 end
37 disp('top=')
38 to=max(topt)
39 q=0;
40
41 for i=1:arr
42      if (to-topt(i)==0)
43          i0=i;
44          q=q+1;
45          figure(q)
46          plot(t,U(:,i0))
47      end
48 end
49
50 function f=f(x,u)%vecteur colonne
51 f=[r*x(1)-mp*x(2)*x(1); -m*x(2)+x(1)*x(2)-u*x(2)];
52 end
53 end
```

Algorithme méthode 2

```
1 function methode2P
2 %methode de Runge-Kutta pour la definition de la contrainte
```

```
3 function c=c(xt,yt,ut,ubt,to,x,y,u,ub)
4       c=zeros(1,2*N+4);
5       Xt=[xt;yt];%vecteur colonne
6       X=[x;y];
7       for i=1:N
8           if i<=p
9                   k1=to*f(Xt(:,i),ut(i));
10                  k2=to*f(Xt(:,i)+(h1/2)*k1,ubt(i+1));
11                  k3=to*f(Xt(:,i)+(h1/2)*k2,ubt(i+1));
12                  k4=to*f(Xt(:,i)+h1*k3,ut(i+1));
13
14                  c(i)=(Xt(1,i+1)-Xt(1,i)-(h1/6)*(k1(1)+2*k2(1)+2*k3(1)+k4
                        (1)));
15                  c(i+p)=(Xt(2,i+1)-Xt(2,i)-(h1/6)*(k1(2)+2*k2(2)+2*k3(2)+
                        k4(2)));
16          else
17              k1=(T-to)*f(Xt(:,i),ut(i));
18                  k2=(T-to)*f(Xt(:,i)+(h1/2)*k1,ubt(i+1));
19                  k3=(T-to)*f(Xt(:,i)+(h1/2)*k2,ubt(i+1));
20                  k4=(T-to)*f(Xt(:,i)+h1*k3,ut(i+1));
21
22
23                  c(i+p)=(Xt(1,i+1)-Xt(1,i)-(h1/6)*(k1(1)+2*k2(1)+2*k3(1)+
                        k4(1)));
24                  c(i+N)=(Xt(2,i+1)-Xt(2,i)-(h1/6)*(k1(2)+2*k2(2)+2*k3(2)+
                        k4(2)));
25          end
26      end
27          for i=1:N
28                  k1=f(X(:,i),u(i));
29                  k2=f(X(:,i)+(h/2)*k1,ub(i+1));
30                  k3=f(X(:,i)+(h/2)*k2,ub(i+1));
31                  k4=f(X(:,i)+h*k3,u(i+1));
32
33                  c(i+2*N)=X(1,i+1)-X(1,i)-(h/6)*(k1(1)+2*k2(1)+2*k3(1)+k4
                        (1));
34                  c(i+3*N)=X(2,i+1)-X(2,i)-(h/6)*(k1(2)+2*k2(2)+2*k3(2)+k4
                        (2));
35
36          end%for
37           c(4*N+1)=xt(1)-x0;
38           c(4*N+2)=yt(1)-y0;
39
40           c(4*N+3)=x(1)-x0;
41           c(4*N+4)=y(1)-y0;
42
43           c(4*N+5)=phi(x)-T+to;
44           c(4*N+6)=xt(p)-xbar;
45
```

```
46 end%function contrainte
47 function phi=phi(x)
48     phi=0;
49     for i=1:N+1
50         phi=phi+h*(1-exp(-dK(x(i))^2/(2*eps)));
51     end
52 end
53
54     function d=dK(a)
55         if a>=xbar
56             d=0;
57         else
58             d=(a-xbar);
59         end
60     end
61
62 function L=Lag(xt,yt,ut,ubt,to,x,y,u,ub,lambda,mu)
63         L=phi(x)+lambda*c(xt,yt,ut,ubt,to,x,y,u,ub)'+(mu/2)*norm(c(xt,yt,
               ut,ubt,to,x,y,u,ub),2)^2;
64 end
65 nu=[xt,yt,ut,ubt,to,x,y,u,ub];
66 z(1,:)=nu;
67 k=0;
68 bo=1;
69 a=0.2;
70
71 while bo
72     q=1;
73
74     k=k+1
75     z(k,:)=nu;
76
77         while (norm(gradLa(xt,yt,ut,ubt,to,x,y,u,ub,lambda,mu),2)>
               tolerence)
78         df=gradLa(xt,yt,ut,ubt,to,x,y,u,ub,lambda,mu);% 1  3*N+4
79         norm(df,2)
80
81         nu=nu-a*df;
82
83      xt=nu(1:N+1);
84      yt=nu(N+2:2*N+2);
85      ut=nu(2*N+3:3*N+3);
86      ubt=nu(3*N+4:4*N+4);
87      to=nu(4*N+5);
88      x=nu(4*N+6:5*N+6);
89      y=nu(5*N+7:6*N+7);
90      u=nu(6*N+8:7*N+8);
91      ub=nu(7*N+9:8*N+9);
92
```

```
93
94            end
95
96            if Lag(xt,yt,ut,ubt,to,x,y,u,ub,lambda,mu)> Lag(z(k,1:N+1),z(k,N
                  +2:2*N+2),z(k,2*N+3:3*N+3),z(k,3*N+4:4*N+4),z(k,4*N+5),z(k,4*
                  N+6:5*N+6),z(k,5*N+7:6*N+7),z(k,6*N+8:7*N+8),z(k,7*N+9:8*N+9)
                  ,lambda,mu)
97                  mu=mu-epsmu;
98            end
99
100        lambda=lambda+mu*c(xt,yt,ut,ubt,to,x,y,u,ub);
101
102
103       bo=(norm(c(xt,yt,ut,ubt,to,x,y,u,ub),2)>tolerence*9);
104 end
105 disp('tout est ok, le controle vaut ')
106 figure(3)
107 plot(x,y)
108
109 figure(1)
110 plot(s,ubt)
111 xlabel('Temps');
112 ylabel('u(t)');
113
114 disp('to calcule=')
115 to
116 end
```

Références

[1] J.-P.Aubin, *Viability Theory* , Birkhäuser, 1991.

[2] T. Bayen, L. Pfeiffer, *Second order analysis for the time crisis problem*, submitted to Journal of Convex Analysis, 2019 https ://arxiv.org/abs/1902.05290

[3] T. Bayen, Alain Rapaport, *Minimal time crisis versus minimum time to reach a viability kernel : a case study in the prey-predator model*, Optimal Control Appl. Methods, vol. 40, 2, 2019, pp. 330–350.

[4] T. Bayen, Alain Rapaport,*About Moreau-Yosida regularization of the minimal time crisis problem*, Journal of Convex Analysis 23 (2016), No. 1, pp. 263-290.

[5] J. Nocedal, S-J. Wright,*Numerical Optimization*, Second edition, Springer.

[6] E. Trélat, *Contrôle optimal : théorie et application*, Vuibert, 2005.

[7] J. F. Bonnans, *Optimisation continue*, Dunod, 2006.

[8] P. Azerad, M. Cuer, *Analyse numérique des équations différentielles*, cours L3.

[9] T. Bayen, *Optimisation numérique*, cours M1.

[10] B. Mohammadi, J-H. Saiac, *Modélisation et optimisation numériques*,Faculté des Sciences de Montpellier et Conservatoire des Arts et Métiers.

[11] John T. Betts, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*, Second edition, SIAM, 2010.