

Utiliser des machines à vecteurs de support pour approcher les contours d'une fonction valeur dans des problèmes d'atteinte de cible

Laetitia Chapel, Guillaume Deffuant

▶ **To cite this version:**

Laetitia Chapel, Guillaume Deffuant. Utiliser des machines à vecteurs de support pour approcher les contours d'une fonction valeur dans des problèmes d'atteinte de cible. 12ieme Journée Scientifique de l'Ecole Doctorale SPI : Apprentissage statistique - Apprentissage symbolique, May 2008, Aubière, France. pp.5. hal-00499497

HAL Id: hal-00499497
https://hal.science/hal-00499497

Submitted on 12 Jul 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Utiliser des machines à vecteurs de support pour approcher les contours d'une fonction valeur dans des problèmes d'atteinte de cible

16 mai 2008

Laetitia Chapel Guillaume Deffuant
Laboratoire d'Ingénierie des Systèmes Complexes, Cemagref
24 avenue des Landais
63172 Aubière Cedex
04 73 44 06 00
prenom.nom@clermont.cemagref.fr

ABSTRACT
Nous proposons d'utiliser un algorithme d'apprentissage particulier, les SVMs, pour résoudre des problèmes d'atteinte de cible en temps minimal. La cible correspond à un état souhaité, en sachant que le système se détériore dans une certaine région de l'espace (lorsqu'il transgresse un ensemble de contraintes de viabilité). Le bassin de capture au temps t correspond à l'ensemble des états qui peuvent atteindre la cible en un temps inférieur ou égal à t, sans quitter l'ensemble des contrainte de viabilité. Les frontières d'un bassin de capture au temps t correspondent alors aux contours d'une fonction valeur au temps t. Les bassins de capture peuvent alors être utilisés pour définir une fonction de contrôle qui permet au système d'atteindre la cible en un temps minimal. Nous proposons une nouvelle approche, basée sur les SVMs, qui permettent d'approcher les bassins de capture successifs et définissons une procédure de contrôle qui permet de contrôler le système afin qu'il atteigne la cible. Nous illustrons cette méthode sur un exemple simple : le problème de la voiture sur la colline.

KEYWORDS
Systèmes dynamiques, théorie de la viabilité, machines à vecteurs de support, bassins de capture, contrôleur optimal de viabilité.

1 Introduction

Les machines à vecteurs de support (SVMs) [18] sont une méthode de discrimination qui montre de bonnes performances dans la résolution de problèmes variés tels que la reconnaissance de formes [6] ou la catégorisation de textes [13]. Le classifieur est alors un hyperplan optimal défini dans un espace « déployé », construit à l'aide d'un nombre réduit de points : les vecteurs de support. Dans cet article, nous montrons comment les SVMs peuvent être utilisées pour résoudre des problèmes d'atteinte de cible.

La théorie de la viabilité, initiée au début des années 1990 par Jean-Pierre Aubin et ses collaborateurs [1], se focalise sur le problème de viabilité : maintenir un système dynamique dans un ensemble de contraintes, noté K. Ce problème est fréquent en économie [5], robotique[18] ou en écologie [14], cite-Mullon2004. Le concept principal est le *noyau de viabilité*, noté $Viab(K)$, qui correspond à l'ensemble des points x pour lesquels il existe une fonction de contrôle $t \leftarrow u(t)$ telle que l'ensemble de la trajectoire partant de x reste indéfiniment dans K.

La théorie de la viabilité traite également du problème d'atteinte d'une cible. Un état *capture* la cible lorsqu'il existe au moins une fonction de contrôle qui permette au système d'atteindre la cible en temps fini, tout en restant à l'intérieur des contraintes de viabilité. L'ensemble des états capturant la cible est appelé *bassin de capture*. A chaque état contenu dans le bassin de capture, on définit son *temps minimal de capture*, qui correspond au temps minimal mis par la trajectoire pour atteindre la cible sans violer les contraintes. Cette fonction correspond à celle obtenue en résolvant les équations Hamilton-Jacobi- Bellman (HJB) en programmation dynamique [12]. Le bassin de capture d'un système correspond au noyau de viabilité du système, auquel on a ajouté à la dynamique une dimension représentant le temps qui s'écoule. Le bassin de capture permet de définir directement des politiques de contrôle, qui fournissent une trajectoire qui reste toujours dans l'espace des contraintes, tout en atteignant la cible en temps minimal.

Il existe plusieurs algorithmes d'approximation de noyaux de viabilité et de bassins de capture. Par exemple, Saint-Pierre [16] a développé un algorithme basé sur la discrétisation de l'espace d'état, qui fournit une approximation du noyau de viabilité comme un ensemble de points viables. [11] a proposé un algorithme qui approche le noyau de viabilité en utilisant une méthode de discrimination. En général, ces algorithmes souffrent de la malédiction de la dimensionnalité, qui limite leur utilisation pratique à des problèmes en petite dimension. En particulier, en les appliquant à des problèmes d'atteinte de cible, ajouter d'une dimension supplémentaire au problème peut avoir augmenter significativement le temps de calcul.

Cet article propose une nouvelle méthode pour résoudre des problèmes d'atteinte de cible, qui évite d'ajouter une dimension auxiliaire au système. Le principe est d'approcher itérativement les bassins de capture à différents temps successifs t. Pour approcher le bassin de capture au temps t, nous utilisons une grille de points qui couvre l'espace K, et étiquetons un point $+1$ lorsqu'il existe un contrôle qui lui permet d'atteindre le bassin de capture au temps $(t - \delta t)$, et -1 sinon. Nous utilisons ensuite les machines à vecteurs de support pour calculer une frontière entre les points de la grille en fonction de leur étiquette. L'algorithme fournit une approximation incluse dans les vrais bassins de capture et permet de définir un contrôleur optimal, qui garantit d'atteindre la cible.

Cet article est divisé en 3 parties : la section 2 rappelle les principaux concepts de la théorie de la viabilité et son application aux bassins de capture. Dans la section 3, nous proposons un nouvel algorithme d'approximation de bassins de capture et de fonction de temps minimal et définissons un contrôleur optimal qui garantit d'atteindre la cible. Nous appliquons les algorithmes sur un exemple d'application : le problème de la voiture sur la colline. Pour terminer, nous donnons quelques pistes de travail.

2 Capturabilité

2.1 Définitions

La théorie de la viabilité traite également le problème d'atteinte d'une cible fermée $C \in K$, sans que le système ne quitte jamais l'espace des contraintes K. Le système dynamique est défini par :

$$x'(t) \in F(x(t)) = \{\varphi(x(t), u(t)), u(t) \in U(x(t))\}. \tag{1}$$

Les évolutions $x(t)$ *capturant la cible* sont les évolutions viables dans K jusqu'à ce qu'elles atteignent la cible C en temps fini :

$$\exists \tau \geq 0 \text{ tel que } \begin{cases} x(\tau) \in C \\ \forall t \in [0,\tau], x(t) \in K. \end{cases} \tag{2}$$

Le *bassin de capture* $Capt_F(K,C)$ [2] du système pour F est l'ensemble de tous les états pour lesquels il existe au moins une évolution qui permette d'atteindre la cible sans quitter K (figure 1). Considérons la correspondance F_I, qui coïncide partout avec F sauf dans la cible :

$$F_I(x(t)) = \begin{cases} F(x(t)) & \text{si } x \notin C \\ \overline{Co}(F(x(t)) \cup \{0\}) & \text{si } x \in \partial C \\ \{0\} & \text{si } x \in int(C), \end{cases} \tag{3}$$

où $\overline{Co}$ est l'enveloppe convexe[1] de $(F(x(t)) \cup \{0\})$, ∂C est la frontière de la cible et $int(C)$ est l'intérieur de C. L'introduction de la correspondance F_I permet d'écrire la relation suivante :

$$Viab_{F_I}(K) = Capt_F(K,C) \cup Viab_F(K) \tag{4}$$

Lorsque K est un répulseur (i.e. $Viab_F(K) = \emptyset$), le bassin de capture coïncide avec le noyau de viabilité du système (3). Cependant, K n'est en général pas un répulseur et une autre approche doit être utilisée pour approcher le bassin de capture du système.

La *fonction de temps minimal* est la fonction qui associe à un état $x_0 \in K$ le temps minimal d'atteinte de la cible C :

$$\vartheta_C^K(x_0) = \inf \{\tau \geq 0 | \exists x(.) \text{ vérifiant } (1), x(0) = x_0, x(\tau) \in C \text{ et } \forall t \in [0,\tau], x(t) \in K\}. \tag{5}$$

Cette fonction est la fonction valeur obtenue lorsque l'on résout les équations Hamilton-Jacobi-Bellman (HJB) en programmation dynamique [12, 7] (pour plus de détails sur la fonction valeur et son approximation dans la théorie du contrôle, le lecteur peut se référer à [4] ou [9]). Elle prend des valeurs dans $\mathbb{R}^+ \cup +\infty$, en particulier $\vartheta_C^K(x_0) = 0$ si $x_0 \in C$ et $\vartheta_C^K(x_0) = +\infty$ s'il n'existe aucune trajectoire qui permette à x_0 d'atteindre la cible sans quitter K. Le bassin de capture de C viable dans K peut être alors défini en utilisant la fonction de temps minimal :

$$Capt_F(K,C) = \left\{x | \vartheta_C^K(x) < +\infty\right\}. \tag{6}$$

On peut également définir le bassin de capture en temps fini T :

$$Capt_F(K,C,T) = \left\{x | \vartheta_C^K(x) \leq T\right\}. \tag{7}$$

Afin de résoudre un problème de capture de cible dans une perspective de viabilité, on considère le système dynamique auxiliaire suivant :

$$(x(t)', y'(t)) \in F_C(x(t), y(t)) = \begin{cases} F(x(t)) \times \{-1\} & \text{si } x \notin C \\ \overline{Co}((F(x(t)) \times \{-1\}) \cup (\{0\} \times \{0\})) & \text{si } x \in \partial C \\ \{0\} \times \{0\} & \text{si } x \in int(C). \end{cases} \tag{8}$$

La variable $y(t)$ représente le temps qui s'écoule et prend ses valeurs dans $\mathbb{R}^+$. [12, 3] prouvent que, avec $\mathcal{H} = K \times \mathbb{R}^+$, l'épigraphe[2] de la fonction valeur $\vartheta_C^K(.)$ est le noyau de viabilité de $\mathcal{H}$ pour l'inclusion différentielle F_C :

$$Viab_{F_C}(\mathcal{H}) = \varepsilon pi(\vartheta_C^K), \tag{9}$$

où $\varepsilon pi(\vartheta_C^K) \in \mathcal{X} \times \mathbb{R}^+$ est l'ensemble $\left\{(x,y) \in \mathcal{X} \times \mathbb{R}^+ | \vartheta_C^K(x) \leq y\right\}$.

[1] L'enveloppe convexe d'un ensemble de points $X \in \mathbb{R}^d$ est l'ensemble convexe minimal contenant X.

[2] L'épigraphe de la fonction f est défini par $\varepsilon pi(f) = \{(x,y) \in I \times \mathbb{R} | f(x) \geq y\}$.

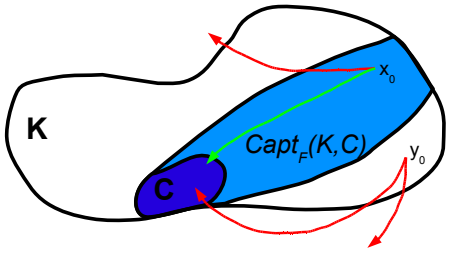


FIG. 1 – Exemple de bassin de capture (représenté en bleu). La cible C à atteindre est représentée en bleu foncé et l'espace des contraintes est délimité par la ligne continue. x_0 et y_0 sont deux états initiaux, et quelques évolutions possibles sont représentées par les flèches. x_0 est un point qui capture la cible car il existe au moins une évolution qui lui permet d'atteindre la cible sans quitter K (évolution capturant la cible en vert), tandis que y_0 est non viable : il n'existe aucune fonction de contrôle qui permette au système d'atteindre la cible tout en restant dans K (évolutions non viables en rouge).

2.2 Contrôleur optimal pour atteindre la cible

Le bassin de capture d'un système est déterminant pour définir des politiques d'actions qui capturent la cible. A partir d'un point appartenant à $Capt(K, C)$, on sait qu'il existe une suite de contrôles qui va permettre au système d'atteindre la cible, tout en restant dans K. La question est : comment choisir ces contrôles ?

[12, 3] caractérisent l'épigraphe d'une fonction valeur V d'un problème de contrôle optimal comme le noyau de viabilité d'un système auxiliaire. Approcher un bassin de capture permet d'estimer la fonction valeur du problème discrétisé. La fonction valeur permet de déduire un contrôleur optimal : en tout état x, le contrôle optimal u^* est le contrôle qui permet d'atteindre le minimum de la fonction valeur :

$$u^* = \arg \min_{u \in U(x)} V(x). \tag{10}$$

Cette caractérisation du bassin de capture permet d'approcher la fonction de temps minimal et de contrôler ainsi le système afin qu'il atteigne la cible le plus rapidement possible.

En général, il n'existe pas de formule explicite pour déterminer le bassin de capture d'un système. Dans la section suivante, nous proposons un algorithme utilisant les SVMs qui permet d'approcher le bassin de capture d'un système, en travaillant directement dans l'espace d'état initial (système (1)).

3 Approximation de bassins de capture en utilisant des SVMs

3.1 Machines à vecteurs de support (SVMs)

Le but général de la discrimination est de construire une fonction qui permet de classer des exemples. Il existe de nombreux algorithmes de discrimination, les SVMs [18] en sont une méthode particulière qui montre de bonnes performances dans de nombreux domaines. Nous rappelons brièvement le principe des SVMs. Pour plus de détail, le lecteur pourra se référer à [10] ou [17].

Considérons une base d'exemples $S = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$ où x_i est un vecteur dans un espace $\mathcal{X} \in \mathbb{R}^N$ et $y_i \in \{-1, +1\}$. Dans le cas le plus simple, lorsque les données sont linéairement

séparables, on construit un hyperplan qui sépare les exemples positifs des exemples négatifs :

$$f(x) = w.x + b = 0 \tag{11}$$

où $w \in \mathbb{R}^N$, $x \in \mathcal{X}$, $b \in \mathbb{R}$ et « . » signifiant produit scalaire.

La propriété remarquable des SVMs est que cet hyperplan est optimal, c'est-à-dire qu'il maximise la distance minimale (marge) entre les exemples et le classifieur. Les points les plus proches, qui seuls sont utilisés pour la détermination de l'hyperplan, sont appelés vecteurs de support (SV).

Dans le cas non linéairement séparable, on utilise une fonction noyau[3]. $K(t, u)$, satisfaisant la condition de Mercer[4], qui permet de projeter les exemples dans un espace de redescription $\mathcal{F}$ de grande dimension et de se ramener ainsi à un cas linéairement séparable.

Lorsque les données sont bruitées, on introduit un paramètre de régularisation C qui permet d'autoriser les erreurs de classification. L'hyperplan séparateur est alors obtenu en résolvant le problème suivant :

$$\begin{array}{l} \max \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \\ \\ \text{Sous contraintes :} \\ \left\{ \begin{array}{l} 0 \leq \alpha_i \leq C \text{ , } i = 1 \text{ à } n \\ \sum_{i=1}^{n} \alpha_i y_i = 0 \end{array} \right. \end{array} \tag{12}$$

La résolution de ce problème quadratique permet la définition de la fonction de décision :

$$f(x) = signe(\sum_{i=1}^{n} \alpha_i y_i K(x_i, x) + b) \tag{13}$$

avec $\alpha_i > 0$ SV.

3.2 Notations

De la même façon que pour l'algorithme d'approximation d'un noyau de viabilité en utilisant une méthode d'apprentissage supervisé, on discrétise le système dynamique dans le temps mais pas dans l'espace. Considérons la correspondance F :

$$F(x) = \left\{ \begin{array}{ll} \varphi(x, u) & \text{si } x \notin C \\ 0 & \text{si } x \in C \\ u \in U(x), & \end{array} \right. \tag{14}$$

et la correspondance $G = 1 + F.dt$ la correspondance discrète associée à F :

$$\left\{ \begin{array}{l} x^{n+1} \in G(x^n) \\ x^0 = x_0. \end{array} \right. \tag{15}$$

On suppose que G est μ-Lipschitz. Soient K et C deux ensembles de $\mathcal{X}$, $C \subset K$, on recherche le bassin de capture du système. On discrétise K avec une grille K_h telle que :

$$\forall x \in K, \exists x_h \in K_h \text{ tel que } \|x - x_h\| \leq \beta(h), \tag{16}$$

[3]Il faut souligner que le terme noyau est employé ici avec une signification différente de celle utilisée dans la théorie de la viabilité

[4]Une fonction $K(t, u)$ respecte la condition de Mercer si pour toute fonction $g(x)$ telle que $\int g(x)^2 \partial x$ est fini, on a $\int \int K(x, y) g(x) g(y) \partial x \partial y \geq 0$

et $\beta(h) \to 0$ lorsque $h \to 0$. Une telle grille existe car K est compact.

A chaque itération n, on définit un ensemble discret $C_h^n \subset C_h^{n-1} \subset K_h$, et un ensemble continu $L(C_h^n)$ qui est une généralisation de cet ensemble discret, et qui constitue l'approximation du bassin de capture au temps $n.dt$.

3.3 Algorithme d'approximation par l'intérieur

Dans cet algorithme, le bassin de capture au temps $n.dt$ inclut son approximation. A chaque itération, l'algorithme ajoute les points tels qu'il existe un contrôle qui permette de rentrer « franchement » à l'intérieur de l'approximation précédente. La convergence de l'approximation par l'intérieur requiert une condition sur la dynamique : un point x à l'intérieur du bassin de capture $n+1$ doit être défini tel qu'il existe $y \in G(x)$ qui appartient à l'intérieur du bassin de capture n. De plus, les conditions sur la procédure de discrimination sont plus strictes à l'intérieur de l'approximation.

A chaque itération, on constitue une base d'apprentissage $\mathcal{S}$ de l'ensemble des points x_h étiquetés $+1$ si le point capture la cible et -1 sinon. On calcule une fonction SVM à partir de $\mathcal{S}$ pour obtenir la fonction de classification f^n et on définit l'ensemble L_h^n comme suit :

$$L_h^n = \{x \in K \text{ tel que } f^n(x) \geq 0\}. \tag{17}$$

On définit itérativement les ensembles L_h^n :

Procédure 3.1 (Algorithme d'approximation d'un bassin de capture par l'intérieur)
On définit itérativement les ensembles L_h^n tels que :

$$\begin{array}{l} L_h^0 = C, \\ C_h^{n+1} = \{x_h \in K_h \textit{ tels que } \exists x \in G(x_h), d(x, K \backslash L_h^n) > \mu\beta(h)\}, \\ L_h^{n+1} = L(C_h^{n+1}) \end{array} \tag{18}$$

Si la procédure de discrimination respecte les conditions suivantes :

$$\forall x \in L_h^n,\, d(x, C_h^n) \leq \beta(h) \tag{19}$$

$$\exists \lambda \geq 1 \textit{ tel que } \forall x \in K \backslash L_h^n,\, d(x, K_h \backslash C_h^n) \leq \lambda\beta(h), \tag{20}$$

et si la dynamique est définie telle que :

$$\forall x \in int(Capt_G(K, C, n+1)), \exists y \in G(x) \textit{ tels que } d(y, K \backslash Capt_G(K, C, n)) > 0, \tag{21}$$

alors

$$\forall n, L_h^n \subset Capt_G(K, C, n), \tag{22}$$

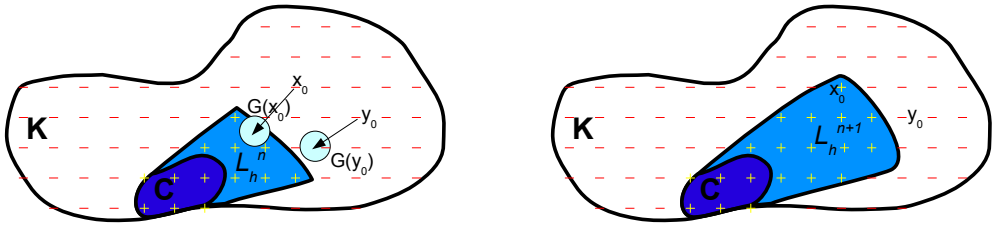
$$\forall n, L_h^n \to Capt_G(K, C, n) \textit{ lorsque } h \to 0. \tag{23}$$

La figure 2 illustre le passage de l'itération n à l'itération $n+1$. La preuve de la convergence de l'algorithme et les conditions que la fonction SVM doit respecter sont donnés dans [8].

A la première itération, la fonction f_0 n'est pas définie analytiquement. En pratique, on pose :

$$C_h^0 = \{x_h \in K_h \text{ tel que } x_h \in C\}, \tag{24}$$

et on calcule la première fonction SVM à partir de l'ensemble d'apprentissage $\mathcal{S}$, constitué des points de K_h avec une étiquette $+1$ si $x_h \in C_h^0$ et une étiquette -1 sinon.



(a) (b)

FIG. 2 – Exemple de passage de l'itération n (figure (a)) à l'itération $n+1$ (figure (b)). Les approximations du bassin de capture courant L_h^n et L_h^{n+1} sont représentées en bleu. L'espace des contraintes est délimité par la ligne continue et la cible C est représentée en bleu foncé. Les ensembles C_h^n et C_h^{n+1} sont représentés par les points « + » en jaune et les ensembles $K_h \backslash C_h^n$ et $K_h \backslash C_h^{n+1}$ par les points « - » en rouge. Les ensembles L_h^n et L_h^{n+1} sont des généralisations des ensembles C_h^n et C_h^{n+1}. A l'itération n, on teste pour tous les points $x_h \in K_h$ si $\exists x \in G(x_h)$ tel que $d(x, K \backslash L_h^n) > \mu\beta(h)$. Si oui, alors $x_h \in C_h^{n+1}$ (point x_0 par exemple sur la figure), et sinon, $x_h \notin C_h^{n+1}$ (point y_0 par exemple).

Utiliser des SVMs pour définir l'approximation courante d'un bassin de capture permet d'utiliser une méthode d'optimisation pour chercher un contrôle viable. On évite ainsi l'augmentation exponentielle du temps de calcul lorsque la dimension de l'espace des contrôles augmente. On recherche ici s'il existe au moins un contrôle qui va permettre à un point d'atteindre l'approximation du bassin de capture au temps précédent. Par construction, on a à l'itération $n+1$, $C_h^{n+1} \subset C_h^n$ (les points qui peuvent atteindre la cible avant le temps $n.dt$ sans quitter K vont pouvoir également l'atteindre avant le temps $(n+1).dt$). Ainsi, il suffit de tester uniquement les points $x_h \in K_h \backslash C_h^n$ pour définir l'ensemble C_h^{n+1} : maximiser $f^n(x_h)$ fournit un contrôle qui permet à la trajectoire $x_h(.)$ d'atteindre L_h^n, si un tel contrôle existe. A partir d'un point x, on réalise une descente de gradient jusqu'à trouver la suite de contrôle $(u_j)_j$ qui maximise la fonction $f_n(x)$:

$$u^* = \arg \max_{u \in U(x)} f_n(x + \varphi(x,u)dt). \tag{25}$$

3.4 Contrôleur optimal en utilisant les SVMs

Le but du contrôleur optimal est de choisir une fonction de contrôle qui permette au système d'atteindre la cible en un temps minimal, sans violer les contraintes de viabilité. L'idée est de choisir le contrôle qui pilote le système afin qu'il traverse les différents ensembles L_h^n dans un ordre décroissant, en suivant la direction de la plus grande pente.

Procédure 3.2 (Algorithme de contrôle optimal)
Considérons $x \in L_h^T$. Notons $n(x)$ défini comme suit :

$$n(x) = \arg\max_n x \in L_h^n. \tag{26}$$

et $u^*(x)$ est tel que :

$$u_i^* = \arg\max_{u \in U(x_i)} f^{n(i)-1}(x_i + \varphi(x_i, u)dt). \tag{27}$$

La fonction de contrôle $u^*(x)$ converge vers une politique de contrôle qui minimise le temps maximal d'atteinte de la cible, lorsque h et dt tendent vers 0.

La preuve que le système atteindra la cible sans jamais quitter K est directe : lorsque h et dt tendent vers 0, les frontières ∂L_h^n tendent vers les contours de la fonction valeur associée à la solution HJB du problème et la procédure choisit le contrôle qui maximise cette fonction valeur.

4 Exemple d'application : la voiture sur la colline

4.1 Définition du problème

On considère ici le fameux problème de la voiture sur la colline (en anglais « car on the hill »). La description du problème est tirée de [15].

Le système est défini en deux dimensions : la position de la voiture x et sa vitesse x'. La voiture peut être contrôlée à l'aide d'une variable de contrôle continue en une dimension : l'accélération a. Le but est de garder la voiture dans un ensemble de contraintes données, tout en atteignant la cible aussi vite que possible.

Le but est d'atteindre le haut de la colline, avec une vitesse faible : $C \in [0,5; 0,7] \times [-0,1; 0,1]$. La voiture doit rester dans l'ensemble de contraintes $K = [-1; 1] \times [-2; 2]$. L'accélération est limitée $a \in [-4; 4]$.
L'évolution de la vitesse x' est fonction de la position x :

$$x'' = \frac{a}{\sqrt{1 + (H'(x))^2}} - \frac{gH'(x)}{1 + H'(x)^2}, \tag{28}$$

avec $g = 9,81$ et :

$$H'(x) = \begin{cases} x^2 + x & \text{si } x < 0 \\ x/\sqrt{1 + 5x^2} & \text{si } x \geq 0. \end{cases} \tag{29}$$

On considère le système dynamique en temps discret suivant :

$$\begin{cases} x(t + dt) = x(t) + x'(t)dt \\ x'(t + dt) = x'(t) + x''(t)dt. \end{cases} \tag{30}$$

4.2 Approximation des bassins de capture et contrôle optimal

La figure 3 présente l'approximation du bassin de capture du système et des contours de la fonction valeur. Si la vitesse est trop faible lorsque la voiture est dans la côte de la colline, même l'accélération maximale ne suffit pas pour remonter la pente. La voiture doit redescendre pour accumuler assez de vitesse et remonter ensuite jusqu'en haut de la colline. La figure 3 présente une trajectoire qui permet d'atteindre la cible en un temps minimal, à partir du point $x_0 = (-0,1; 0,4)$.

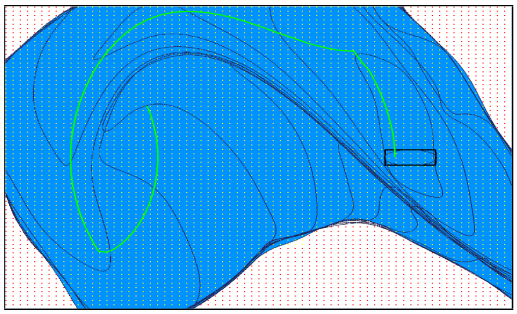


FIG. 3 – Approximation du bassin de capture du problème de la voiture sur la colline. L'axe horizontal représente la position $x(t)$ et l'axe vertical la vitesse $x'(t)$. K est le rectangle qui limite les points de la grille. Le bassin de capture est représenté en bleu. Les différentes lignes de niveaux représentent les contours de la fonction valeur approchée. La grille contient 71 points par dimension.

5 Conclusion

Nous avons proposé un algorithme d'approximation de bassins de capture et de fonction de temps minimal. L'utilisation de SVMs fournit une approximation parcimonieuse des bassins de capture et permet d'utiliser une méthode d'optimisation pour trouver les contrôles qui capturent la cible. Ce dernier point est particulièrement important pour traiter des problèmes dont l'espace des contrôles est en grande dimension. De plus, nous obtenons une approximation par l'intérieur, ce qui garantit d'atteindre la cible, même si l'approximation n'est pas très fine.
Cette méthode pourrait être étendue au-delà des problèmes d'atteinte de cible. En effet, les problèmes de programmation dynamique, dans lesquels il s'agit de trouver une politique d'action qui maximise une récompense cumulée, peuvent être traité comme des problèmes de viabilité, en ajoutant une dimension supplémentaire qui représente la récompense cumulée. Il est donc maintenant important d'identifier les conditions dans lesquelles notre approche, évitant d'ajouter cette dimension supplémentaire, peut s'appliquer à de tels problèmes.

REMERCIEMENTS
Ce travail a été partiellement financé par le contrat Européen PATRES (FP6 NEST n˚043268).

Références

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

[2] J.-P. Aubin. Viability kernels and capture basins of sets under differential inclusions. *SIAM Journal on Control and Optimization*, 40(3) :853–881, 2001.

[3] J.-P. Aubin and H. Frankowska. The viability kernel algorithm for computing value functions of infinite horizon optimal control problems. *Journal of mathematical analysis and applications*, 201(2) :555–576, 1996.

[4] R. Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[5] C. Béné, L. Doyen, and D. Gabay. A viability analysis for a bio-economic model. *Ecological Economics*, 36(3) :385–396, 2001.

[6] C. Burges. A tutorial on Support Vector Machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2) :121–167, 1998.

[7] P. Cardaliaguet, M. Quincampoix, and P. Saint-Pierre. Optimal times for constrained nonlinear control problems without local optimality. *Applied Mathematics & Optimization*, 36 :21–42, 1997.

[8] L. Chapel. *Maintenir la viabilité ou la résilience d'un système : les machines à vecteurs de support pour rompre la malédiction de la dimensionnalité ?* PhD thesis, Université Blaise Pascal, 2007.

[9] M.-G. Crandall and P.-L. Lions. Viscosity solutions of hamilton-jacobi equations. *Transactions of the American Mathematical Society*, 277(1) :1–42, 1983.

[10] N. Cristianini and J. Shawe-Taylor. *Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, 2000.

[11] G. Deffuant, L. Chapel, and S. Martin. Approximating viability kernels with support vector machines. *IEEE transactions on automatic control*, 52(5) :933–937, 2007.

[12] H. Frankowska. Optimal trajectories associated with a solution of the contingent hamilton-jacobi equation. *Applied Mathematics and Optimization*, 19(1) :291–311, 1989.

[13] T. Joachims. Text categorization with support vector machines : learning with many relevant features. In *Proceedings of ECML-98, 10th European Conference on Machine Learning*, 1998.

[14] V. Martinet and L. Doyen. Sustainability of an economy with an exhaustible resource : A viable control approach. *Resource and Energy Economics*, 29(1) :17–39, 2007.

[15] A. Moore and C. Atkeson. The parti-game algorithm for variable resolution reinforcement learning in multidimensional state-spaces. *Machine Learning*, 21 :199–233, 1995.

[16] P. Saint-Pierre. Approximation of the viability kernel. *Applied Mathematics & Optimization*, 29(2) :187–209, 1994.

[17] B. Scholkopf and A. Smola. *Learning with Kernels : Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2002.

[18] V. Vapnik. *The nature of statistical learning theory*. Springer Verlag, 1995.