Computing the viability kernel in large state dimension

Noël Bonneuil a,b,*

a Institut national des études démographiques, 133, bld Davout, 75980, Paris cedex 20, France
b École des hautes études en sciences sociales, France

Available online 28 December 2005
Submitted by Steven G. Krantz

Abstract

T-viable states in a closed set K under a certain set-valued dynamic are states from which there exists at least one solution remaining in K until a given time horizon T. Minimizing the cost to constraints lets us determine whether a given state is T-viable or not, and this is implementable in large dimension for the state-space. Minimizing on the initial condition itself lets find viable states. Quincampoix’s semi-permeability property helps find other states located close to the viability boundary, which is then gradually delineated. The algorithm is particularly suited to the identification of specific trajectories, such as the heavy viable solution, or to the computation of viability kernels associated with delayed dynamics. The volume of the viability kernel and its confidence interval can be estimated by randomly drawing states and checking their viability status. Examples are given.

Keywords: Viability theory; Set-valued analysis; Simulated annealing

1. Introduction

Consider a finite-dimensional vector space X, a compact subset K of X, and the dynamic:

\[
\begin{cases}
  x'(t) \in F(x(t)) := \{f(x(t), u(t)), u(t) \in U\} & \text{for all } t \geq 0, \\
  x(0) = x_0 \in K,
\end{cases}
\]

where F is a Marchaud map1 defined from K to X.

* Fax: +33 1 56 06 21 99.
E-mail address: bonneuil@ined.fr.

1 A set-valued map \( F: X \to K \) is a Marchaud map if the graph and the domain of F are closed and not empty; the values \( F(x) \) are convex, and F is non “explosive”: \( \exists c \forall x \in \text{Dom}(F), \| F(x) \| := \max_{y \in F(x)} \| y \| \leq c(\| x \| + 1) \).

© 2005 Elsevier Inc. All rights reserved.

A state \( x_0 \) is said to be *viable* if there exists a solution \( x(\cdot) \) to (1) that forever remains in \( K \):

\[
\forall t \geq 0, \quad x(t) \in K.
\] (2)

Aubin [1] defined a viability domain as a set of viable states, and showed that there exists a largest viability domain that includes all others. He called this maximal set the *viability kernel* of \( K \), denoted \( \text{Viab}_F(K) \).

Saint-Pierre [11] devised a clever algorithm to compute this viability kernel when \( F \) is Marchaud. He discretized (1) so that the sequence of discrete viability kernels of \( K \) converges to a subset contained in the viability kernel of \( K \) under \( F \). He showed that this sequence converges to the viability kernel if \( F \) is also Lipschitz (see also [8]).

Similarly, a viability kernel is associated with the discrete dynamic:

\[
\begin{cases}
  x_{j+1} \in \Phi(x_j) := \{ \phi(x_j, u_j), u_j \in U \} & \text{for all } j \in \mathbb{N}, \\
  x(0) = x_0 \in K,
\end{cases}
\] (3)

where the graph and the domain of \( \Phi \) are nonempty and closed. Euler discretization allows us to approximate the viability problem (1) by (3) [11].

The results and the algorithm implemented by Saint-Pierre apply theoretically in any dimension of the control- and state-space. However, in practice, as \( K \) is reduced to a discrete grid the algorithm must be able to update every cell of the grid at each time, which requires large amounts of computer memory. Viability kernels are routinely computed in three dimensions [2,3]. Nonetheless, the “barrier” of the discrete representation of the state-space remains. Here, I propose and explore a procedure to compute the viability kernel and its volume in larger dimension, using Quincampoix’s theorem on the semi-permeability of the viability boundary, together with an original combination of classical simulated annealing.

I will present the continuous-time case with its discretized scheme, because the application to the discrete-time case is straightforward.

2. Method

2.1. Characterizing states of the viability kernel

Let the state space be \( \mathbb{R}^p \), the control space \( \mathbb{R}^q \), and \( T > 0 \) the time horizon. We shall compute the viability kernel \( \text{Viab}_F^K(T) \) at time horizon \( T \).

The set of constraints \( K \) is represented by a constraint on state \( x \in X \):

\[
K = \{ x \mid h(x) \leq 0 \}.
\] (4)

This constraint on \( h(x) \) can be either explicit, such as:

\[
h(x) = \max_{i=1, \ldots, m} g_i(x),
\] (5)

where \( g_i : \mathbb{R}^n \to \mathbb{R}, i = 1, \ldots, m \); or implicit:

\[
h(x) = d(x, K),
\] (6)

where \( d(x, K) \) is the distance of \( x \) to \( K \).

I denote \( S(x) \) as the set of all solutions to (1) starting from a given state \( x \), and define a cost at state \( x \) as:

\[
c(T, x) := \inf_{x(\cdot) \in S(x)} \sup_{t \in [0, T]} h(x(t)).
\] (7)
Lemma 2.1.

\[ x \in \text{Viab}^T_F(K) \iff c(T, x) \leq 0. \]  (8)

**Proof.** By definition of viability, a state \( x \in K \) is viable if there exists at least one trajectory remaining in \( K \) until time \( T \), or every constraint defining \( K \) is satisfied until time \( T \):

\[ \exists x(\cdot) \in S(x) \text{ such that } \sup_{t \in [0, T]} h(x(t)) \leq 0. \]  (9)

Hence the lemma. \( \square \)

**Remark.**

\[ \text{Epigraph}(c(\infty, \cdot)) = \text{Viab}(\ast)(\text{Epigraph}(h)), \]  (10)

where \( \ast \) represents the dynamic:

\[ \begin{cases} 
    x' \in F(x), \\
    y' = 0.
\end{cases} \]  (11)

The implementation of (7) through the discretization scheme under step \( \rho > 0 \):

\[ x^{(n+1)} = x^{(n)} + \rho F(x^{(n)}) \]  (12)

takes place in \( J := \frac{T}{\rho} \) dimensions, and requires a minimization routine in large dimension, such as simulated annealing [10], available in [6]. Note equivalently \( x^{(1)}, \ldots, x^{(J)} \) or \( x(\rho), \ldots, x(\frac{T}{\rho}) \).

- The first result is that I can determine whether a state \( x \) is viable or not under (3) without knowing the viability kernel, which was a prerequisite in [11].
- The second result is that this is implementable for any finite dimension \( p \) of the state space.
- The third result is that specific solutions, such as heavy viable solutions, come from the general case.
- The algorithm is adaptable to delayed dynamics.

A drawback is that the time horizon is limited, and each control spans over \( T \) time units, which is \( J = \frac{T}{\rho} \) values under the discretized scheme. Simulated annealing routinely works with 200 variables, so that we can deal with a time horizon of 200 time units with one control, or 100 time units and two controls, or 67 time units and three controls. In contrast, the procedure of [11] is adapted to infinite \( T \) (solutions remain in \( K \) forever). My approach handles large dimensions of state space in exchange for limitations on time horizon.

- If \( x \) belongs to the interior of \( \text{Viab}^T_F(K) \), then there is no need to go as far as the minimum of \( \sup_{t \in [0, T]} h(x(t)) \): simulated annealing stops as soon as one solution remaining in \( K \) is found. Its cost \( c(T, x) \) is \( \leq 0 \). If \( c(T, x) = 0 \), then \( x \) is viable and the solution \( x(\cdot) \) starting from \( x \) encounters \( \partial K \). Moreover, an element of the regulation map in the discretized scheme:

\[ R_K(x) := \left\{ (u^{(1)}, \ldots, u^{(J)}) \in U^J \mid x(\cdot) \in K, x(0) = x, \ x^{(n+1)} = x^{(n)} + \rho F(x^{(n)}) \right\} \]  (13)

is obtained as the sequence \( (u^{(1)}, \ldots, u^{(J)}) \) associated with \( c(T, x) \).
• If \( c(T, x) > 0 \), the solution starting from \( x \) leaves \( K \), and simulated annealing proceeds to its end. The question is whether \( c(T, x) > 0 \) close to 0 gives us useful information on the location of \( x \) with respect to the viability boundary.

**Theorem 2.2.** For \( F : X \rightarrow K \) \( \lambda \)-Lipschitz, \( K \) a closed set defined by \( l \)-Lipschitz constraints \( h \), and \( c(T, x) = \inf_{x(t)} \sup_{t \in [0, T]} h(x(t)) \), then:

\[
\forall x, y \in K, \quad c(T, y) \leq c(T, x) + le^{\lambda T} \| y - x \|.
\] (14)

**Proof.** Consider a solution \( x(\cdot) \) starting from \( x(0) = x \) such that \( c(T, x) = \sup_{t \in [0, T]} h(x(t)) \). Then, by Filippov’s theorem [7], for all \( T > 0 \), there exists a solution \( y(\cdot) \) starting from \( y \) such that

\[
\forall t \leq T, \quad \| y(t) - x(t) \| \leq e^{\lambda T} \| y - x \|.
\] (15)

Hence,

\[
h(y(t)) \leq h(x(t)) + le^{\lambda T} \| y - x \|.
\] (16)

The same holds true for extrema, so (14) holds true. \( \square \)

Consequences:

- If \( x_0 \in \text{Viab}_F(K) \), then \( c(T, y) \leq le^{\lambda T} \| y_0 - x_0 \| \);
- \( \forall x_0 \in K, c(T, y) \leq le^{\lambda T} d(y, \text{Viab}_F(K)) \);
- If \( x_0 \in \partial \text{Viab}_F(K) \), then \( \forall \eta \geq 0, \exists y_0 \in B(x_0, \eta) \) such that \( c(T, y_0) \leq \eta le^{\lambda T} \).

Thus, the closer a state \( y \) is to the viability kernel, the smaller is \( c(T, y) \).

However, the converse is not true: a state may have a small cost without being close to the viability kernel. The remedy, once a small cost \( c(T, y) \) is obtained in a state \( y \), is to search for the existence of a viable state \( x \) in a small ball \( B(y, \epsilon) \) around \( y \). If one exists, then \( y \) is not farther away than \( \| y - x \| \leq \epsilon \) from the viability kernel.

In the case where \( h(x) = d(x, K) \),

\[
\inf_{x(t) \in S(x), t \in [0, T]} \sup_{\text{Viab}_F(K)} d(x(t), K) \leq \inf_{x(t) \in S(x), t \in [0, T]} \sup_{\text{Viab}_F(K)} d(x(t), \text{Viab}_F(K)) \leq e^{\lambda T} d(x, \text{Viab}_F(K))
\] (17)

so that the cost associated with a given state \( x \) is bounded by product of the distance of \( x \) to the viability kernel and an exponential in \( T \). This may be large.

Having located a state \( x \) “close enough” to the viability boundary, the part of the trajectory located in the interior of \( K \) (from \( x \) to the first encounter of the solution of (7) with \( \partial K \)) travels close to the viability boundary, without entering into the viability kernel. This results from Quincampoix’s [9] theorem of semi-permeability for \( F \):

**Theorem 2.3 (Semi-permeability property, Quincampoix [9, p. 414]).** For \( F \), a Lipschitz map with convex compact nonempty values and a closed set \( K \), if \( x_0 \) is a state of \( \partial \text{Viab}_F(K) \setminus \partial K \), there exists a viable solution in \( K \) starting from \( x_0 \) which remains in the boundary of \( \text{Viab}_F(K) \) as long as it does not encounter \( \partial K \), and this is the case for every viable solution starting from \( x_0 \).

This theorem was adapted to the discrete case by [4]:
Theorem 2.4 (Discrete-time semi-permeability property). For $K$ a closed set and $G: X \rightarrow X$ a locally Lipschitz correspondence with nonempty compact convex values, a solution $(x_j)_{j=1,\ldots,J}$ reaching the boundary of the viability kernel remains on it and is extremal (but not necessarily unique) until it reaches the boundary of $K$.

In the discretized scheme, if $(n_1 + 1)\rho \leq T$ is the first time when the solution stemming from a given state $x$ encounters or crosses $\partial K$, then the states $x(0), x(\rho), \ldots, x(n_1\rho)$ (or in discrete notation $x(0), x(1), \ldots, x(n_1)$) are good candidates to be close to the viability boundary. However, $x$ can be viable at $T$ and $x(i)$ only viable at $T - \rho i$, $i \in \{1, \ldots, J\}$. Then the viability of $x(i)$, $i = 1, \ldots, n_1$ must be checked by again calculating the minimum cost $c(T, x(i))$. For $x(1)$, a possible shortcut is to search to prolong the sequence $x(2), \ldots, x(n_1)$ by a state in $K$, and so on for all successive $x(j)$. Otherwise, the viable trajectory stemming from $x(1)$ may not contain the following $x(j)$, $j = 2, \ldots, n_1$, and the cost $c(T, x(1))$ must be computed again to check whether $x(1)$ is $T$-viable.

If $x(i)$ is viable and the dimension of the state space is greater than two, then there may exist more than one trajectory on the viability kernel starting from $x(i)$. In this case, starting the simulated annealing with orthogonal initial condition in $u(1), \ldots, u(J)$ may yield a different solution, adding new candidates of viable states along the new trajectory stemming from $x(i)$.

In reverse time, solutions $x(-\cdot)$ starting from $x$ under:

$$x'(t) \in -F(x(t))$$  \hspace{1cm} (18)

may enter into the viability kernel, because of the semi-permeability property: in forward time, once outside the viability kernel, it is not possible to re-enter it; but it is possible to start from inside the viability kernel, go out, then travel along the boundary. Hence, in reverse time, a trajectory of low cost $c(T, x) > 0$ may have crossed the viability kernel, and does not help locate the viability boundary.

2.2. Finding viable states in the discrete viability kernel

We can now determine the viability of a given state. To delineate the viability kernel, it would be too time-consuming to draw states $x$ at random from $K$, and it would take too much memory to test every state of a given discrete grid. Exploring only $\partial K$ in the case when $\text{Viab}_T^F(K) \cap \partial K \neq \emptyset$ could work, but the dimension to explore remains large with $p$ large.

The procedure I propose is to search for an initial state $x_0$ close to the viability boundary by simulated annealing on $x_0$ itself. Because the cost $c(x_0)$ is positive for $x_0$ outside the viability kernel and $\leq 0$ for $x_0 \in \text{Viab}_T^F(K)$, the simulated annealing on initial state $x_0$ works with a modified cost:

$$\tilde{c}(x_0) = \begin{cases} c(x_0) & \text{if } c(x_0) > 0, \\ c(x_0) + \alpha & \text{if } c(x_0) \leq 0, \end{cases}$$  \hspace{1cm} (19)

with $\alpha > \text{Sup}_{x \in K} h(x)$. At each trial of $x_0$ during simulated annealing, a new simulated annealing is conducted in the discretized scheme on $(u(1), \ldots, u(J))$ at fixed $x_0$ to calculate the cost $c(x_0)$ to determine if this trial of $x_0$ is viable. During the search, if $c(T, x_0) \leq 0$ then I retain that $x_0$ belongs to the viability kernel. When $c(T, x_0)$ is positive and small, I explore the ball $B(x_0, \epsilon)$ for $\epsilon$ sufficiently small to know whether there exists a viable state close to $x_0$, and to determine if $x_0$ is located near the viability boundary. Then, the third result is that we can identify viable states in $K$. 

2.2.1. Examples

Aubin [1] gives the analytical expressions of the viability boundary in the continuous-time dynamic:

\[
\begin{align*}
    x'(t) &= x(t) - y(t), \\
    y'(t) &= u(t) \in [-c, c], \\
    K &= [0, 1] \times [0, \infty). \\
\end{align*}
\]  

Figure 1 shows the result of the double simulated annealing. The search was confined to the boundary of \( K \). During the search, each viable state \( x \in \partial K \) is retained, which appears on Fig. 1.

In dimensions above three, viability kernels are seldom easily computable. I suggest the following simple example, with which computations can be tested:

\[
\begin{align*}
    x_i'(t) &= x_p(t), & i = 1, \ldots, p - 1, \\
    x_p'(t) &= u(t) \in [-c, c], \\
    K &= [-1, 1]^p. \\
\end{align*}
\]  

The viability kernel of this constrained dynamic has a simple projection onto each \((x_i, x_p)\), \( i = 1, \ldots, p - 1 \), with the \( x_i \)-axis as axis of symmetry and with \( x_i = 1 - x_p^2/(2c) \) as the boundary in the top-right quadrant from \( x_i = 1 - 1/(2c) \) to \( x_i = 1 \). Figure 2 presents the result for two dimensions and 50 trials.

Figure 3 presents the triangulation of viable states in three dimensions (1000 trials). The result is similar to the representation obtained by Saint-Pierre’s viability algorithm. Figure 4 presents the projection onto \( x_{10} \) and one of the other \( x_i \) of the result obtained in 10 dimensions.

Computation time depends both on the parameters of the simulated annealing, and on the nonlinearity of the dynamics. As an example, for dynamic (21), on my Pentium IV D810 (Intel Pentium MP750, 1.86 GHz, 533 MHz FSB, RAM 512 MB), with discretization step 0.03, total
number of steps $J = 36$, and 200 iterations for the simulated annealing, testing the viability of one given point takes 0.066 seconds. Searching for a point close to the viability boundary requires double simulated annealing. For the same steps and dynamic, with 200 iterations for each simulated annealing, it took 22 minutes on average per initial point. This initial point however helps find others. The trajectory starting from this initial point yields other points close to the viability kernel thanks to the semi-permeability property. From each point of this trajectory, other trajectories are launched which travel along the viability boundary. These trajectories are produced at a rate of 131 per 22 minutes. Adding the 67 points per 22 minutes found at $\partial K \cap \text{Viab}_F(K)$ for which the cost is null, the total number of points found per 22 minutes amounts to 224. This was obtained for 3 dimensions. For 10 dimensions, the same dynamic, and the same parameters, the performance is approximately the same.
2.3. Capture basins

Let $K \subset X$ be a set of constraints and $C \subset K$ a target. The subset $\text{Capt}_S(K, C)$ of initial states $x \in K$ such that at least one solution $x(\cdot) \in S(x)$ starting from $x$ remains in $K$ until it reaches $C$ in finite time is called the capture basin of $C$ viable in $K$ [3].

We again define the set of constraints $K$ and the target $C$ through constraints:

$$
\begin{align*}
K &= \{ x \mid h(x) \leq 0 \}, \\
C &= \{ x \mid \gamma(x) \leq 0 \}.
\end{align*}
$$

For example, $h(x) = d(x, K)$ and $\gamma(x) = d(x, C)$.

In the discrete-time case (the continuous-time being straightforward), define:

$$
c(J, x) := \inf_{(x_j)_{j=1}^J \in S(x)} \inf_{j \leq J} \max_{i \leq j} \left( \gamma(x_j), \sup_{i \leq j} h(x_i) \right).
$$

If $c(J, x) \leq 0$, then $\exists (x_j)_{j=1}^J \in S(x)$, $\exists j_* \leq J$ such that $\gamma(x_{j_*}) \leq 0$ and $\forall i \leq j_*, x_i \in K$.

Hence,

$$
x \in \text{Capt}^J_F(K, C) \iff c(J, x) \leq 0.
$$

We apply the same minimization on $u(1), \ldots, u(J)$ to determine whether a given state $x$ is viable, and to further delineate the capture basin of $C$ viable in $K$.

2.4. Heavy solutions

Denote $U : X \sim \mathcal{U}$ a set-valued map associating the set $U(x)$ of controls admissible in $x$. Aubin [3] studied specific systems where controls $u$ are prevented from fluctuating too quickly, or $\|u'(t)\| \leq r$ with $r > 0$:

$$
\begin{align*}
\begin{cases}
x'(t) = f(x(t), u(t)), \\
u'(t) \in rB \quad \text{with } r > 0,
\end{cases}
\end{align*}
$$
under constraints on state $x$ and control $u$:

$$\begin{cases}
x(t) \in K, \\
u(t) \in U(x(t)).
\end{cases}$$

(26)

Define the constraint function as:

$$h(x, u) = \max(d(x, K), d\left(u, U(x)\right)).$$

(27)

Under the discretized scheme, with time $j = 1, \ldots, J$:

$$\begin{cases}
x^{(j+1)} = x^{(j)} + \rho f(x^{(j)}, u^{(j)}), \\
u^{(j+1)} = u^{(j)} + \rho v^{(j)},
\end{cases}$$

(28)

with $v^{(j)} \in rB$, the sign of the cost

$$c(J, x, u) = \inf_{v^{(j)} \cdots v^{(J)}} \sup_{j = 1, \ldots, J} \max(d(x^{(j)}, K), d\left(u^{(j)}, U(x^{(j)})\right))$$

(29)

associated with a given $(x, u)$ tells us whether $(x, u)$ is viable. Moreover, the $(v^{(j)})_{j=1, \ldots, J}$ for which both $c(J, x, u)$ is obtained and with largest norm yields the inertia function at time horizon $J$ [3].

To obtain a heavy solution starting from a given state $x_0$, I propose a double optimization, the first by dichotomy on the time $j_0$ until which $d(u^{(j)}, U(x^{(j)}))$ is minimal, then for each $j_0$, by simulated annealing to calculate $\tilde{c}(x^{(j_0)})$. Thus, the entire solution has $j_0 + J$ time steps.

Figure 5 shows two examples of heavy solutions, one joining the upper viability boundary, the other the lower viability boundary.

2.5. Computing the volume of the discrete viability kernel

Studying the variations of the volume of the viability kernel with respect to the set of control $U$ has meaningful interpretations in applications [4,5].

To compute the volume of the viability kernel, I classify the $i$th trial as viable, $X_i = 1$, or not viable $X_i = 0$, $i = 1, \ldots, N$ for a certain (large) number $N$ of initial states $x_0$ drawn at random.
in the state space. The random variable $X$ is then a Bernoulli of parameter $\theta = \frac{\text{vol}(\text{Viab}_F(K))}{\text{vol}(K)}$, and $\hat{\theta} = \frac{1}{N} \sum_{i=1}^{N} X_i$ is an unbiased convergent estimate of $\theta$. Multiplying the proportion by the volume of $K$ yields the volume of $\text{Viab}_F(K)$. This requires running the simulated annealing on $(u^{(1)}, \ldots, u^{(J)})$ for each randomly drawn $x_0$. This is an easy procedure which has the advantage of being operable in any state dimension. In the 2D example of (21), direct calculation of the volume gives a proportion of 0.83 times the volume of $K$, while random trial and viability testing by simulated annealing gives 0.85 with confidence interval $[0.76; 0.90]$ for 100 trials; 0.82 with confidence interval $[0.79; 0.87]$ for 300 trials.

2.6. Differential inclusions with delay

The procedure consisting of computing trajectories and retaining them as the optimum with a certain probability makes it possible to deal with dynamics delayed with a lag $0 < \tau \ll T$:

$$\begin{align*}
\begin{cases}
x'(t) &= f(x(t - \tau), u(t)) \quad \text{for } t \in [\tau, T], \\
u(t) &\in U \quad \text{for } t \in [0, T].
\end{cases}
\end{align*}$$

(30)

In the algorithm, from each candidate state $x_0$ and each sequence of discretized values $u(np)$, $n \in [0, \tau/\rho]$, trajectories are launched to search for the minimum cost. Figure 6 shows the example of the simple 2D delayed dynamic:

$$\begin{align*}
\begin{cases}
x'(t) &= x(t - 0.5) - y(t), \quad t \in [0.5, T], \\
y'(t) &= u \in [-c, c], \quad t \in [0, T], \\
K &= [0, 1] \times [0, T].
\end{cases}
\end{align*}$$

(31)

Fig. 6. Estimated viability kernel of continuous-time dynamics $x'(t) = x(t - 0.5) - y(t); y'(t) \in [-c, c]$ and comparison with the theoretical viability kernel of the nondelayed dynamics $x'(t) = x(t) - y(t); y'(t) \in [-c, c]$ (same $c$ for both kernels).
3. Conclusion

In computing the cost in (7), I can determine whether a state is viable without needing to compute the whole viability kernel as in [11]. The state space dimension is no limitation, because no discrete grid, which is memory consuming, is necessary. The only limitation is in time horizon. Moreover, by double simulated annealing on modified cost \( \tilde{c}(J, x) \), I can reach a state close to the viability boundary. By relying on the semi-permeability property, I can locate solutions traveling along this viability boundary, and thereby determine the boundary. Launching trajectories minimizing the cost from many initial states allows the eventual delineation of the viability kernel. In three dimensions, triangulation procedures help represent its shape.

Heavy solutions, which are noteworthy in viability theory, are obtained by dichotomizing time followed by simulated annealing for \( \tilde{c}(J, x) \). As before, this is obtained without prior knowledge of the viability kernel.

As each trajectory is temporarily followed, the algorithm is well suited to dynamics with delay.

Computing the volume of the viability kernel in large state dimension follows from counting the proportion of viable states in \( K \) for a certain number of initial states drawn at random.

References