Geometric robustness of viability kernels and resilience basins

Isabelle Alvarez, Sophie Martin

Chapter 8

GEOMETRIC ROBUSTNESS OF VIABILITY KERNELS AND RESILIENCE BASINS

Isabelle Alvarez [1] [2] and Sophie Martin [1]

[1] *Cemagref - LISC, 24 av. des Landais 63172 Aubière, France*
isabelle.alvarez@cemagref.fr
sophie.martin@cemagref.fr

[2] *LIP6, UPMC, 5 place Jussieu 75005 Paris, France*

Abstract In chapter 2, the concepts of viability kernel and resilience basin are proposed to characterize the viability and resilience of social and ecological systems. These systems are submitted to strong disturbances and their models embed uncertainties and approximations. Therefore, it is essential to consider the stability and the robustness of their resilience. This is classically done with sensitivity analysis which measures the difference of size of the viability kernel and of the resilience basin under small deviations of parameters. In this chapter, we propose in addition to study the geometry of the viability kernel or resilience basin as subsets of the state space. We show that some indicators of the shape of these sets give information about how risky disturbances or uncertainties are. The distance to the set boundary gives information about the robustness of trajectories, which are safer far from the boundary. It is also an indicator of robustness for control policies, which aim at keeping the trajectories far from the boundary.

1. Introduction

The definition of resilience described in chapter 2 applies to dynamical systems whose dynamics are modelled by controlled differential equations and in which some properties of interest are defined by a subset of the state space (the constraint set). These dynamical systems, when they model environmental or socioeconomic systems, are subject to uncertainties. Moreover, the properties of interest are rarely known with absolute certainty and accuracy. Viability theory can take into account only a part of these uncertainties, considering a set of velocity vectors rather than a single vector. Therefore, performing sensitivity

analysis (like in Saltelli et al., 2000) appears as a good solution to assess the impact of the other parameters of the dynamics, and also the impact of slight modifications of the boundary of the constraint set on the viability kernel and its capture basin.

As seen in Chapter 2, the resilience value is infinite inside the viability kernel, but outside this set it can switch to a finite value or even to zero. If a perturbation leads the system to a state of zero resilience value, there is no hope of driving it back eventually to the viability kernel.

This is the reason why an explicit study of the robustness of states and trajectories is so important in viability studies. In this chapter, we propose a geometric method to appraise the robustness of the results given by a viability study. Intuitively, the robustness or the risk associated with a given state depends on its position relative to the boundary of the viability kernel or the resilience basin.

These geometric concepts and their interest according to the resilience issue are described in section 2, with a simple example. Prerequisite, correct use of the method and algorithms are described in section 3. The geometric study is illustrated on the PATRES case study of language competition in section 4 (see Chapter 3). It highlights the role the robustness information can play when controlling the system.

2. Geometric Criteria of Robustness in Viability and Resilience Analyses

2.1 Geometric description of relevant sets

As in Chapter 2, we consider a dynamical system and we suppose that a property of interest of this system is defined as a constraint set (subset of the system's state space). The viability kernel is formed by all the states belonging to the constraint set that are viable, that is, the subset of initial points from which it is possible to maintain the system inside the constraint set. The resilience basin is defined as the capture basin of the viability kernel, that is the set of all states from which it is possible to reach the viability kernel in finite time.

The volume of these sets is useful to qualify the robustness of the system. The smaller they are, the less robust the system is. For example, if the volume of the viability kernel is very small compared with the volume of the constraint set, then it will be difficult to maintain the system in this desired set. If the volume of the resilience basin is very small compared with the volume of the state space (when its size is finite), then the system itself is not very resilient.

The variation of these sets with small modifications of the parameters is also a valuable piece of information. In particular, if small modifications of the constraint set lead to catastrophic modifications of the viability kernel (empty

or very small set), then the viability study is not robust to uncertainties in parameters.

Besides this sensitivity analysis of the model, the geometric description provides more information concerning the robustness to uncertainty or measurement error of the state and control variables. Figure 8.1(a) shows two viability kernels with the same volume. Obviously a dynamical system that evolves in the top left kernel is less resistant to perturbation than a dynamical system that evolves in the top right kernel.

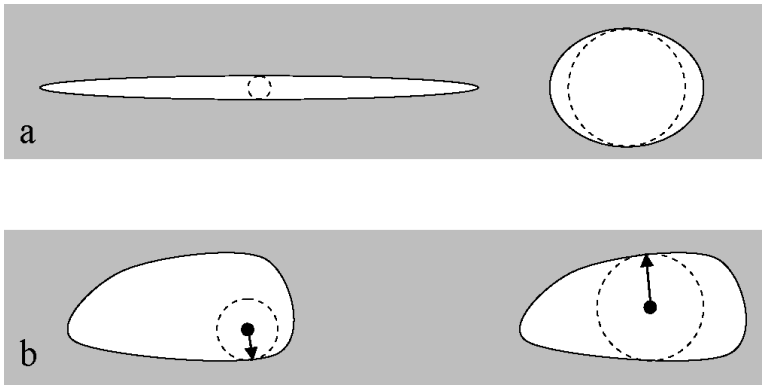

Figure 8.1: Two examples of situations where the geometric description brings useful information. (a) Two kernels with the same volume. A system evolving in the left kernel cannot resist even small disturbance along the vertical axis. Dot circles show the respective largest maximal balls of both sets. (b) Two points in the same kernel. The dashed circle shows the largest perturbation that keep the point inside the viability kernel.

A useful indicator of the shape of the viability kernel or of the resilience basin is the diameter of the largest maximal ball. Maximal balls inside a set are open balls that are not contained in any larger ball inside the set. Centres of maximal balls form the skeleton (see Serra, 1988, for more information about mathematical morphology concepts). The largest maximal ball is the largest ball inscribed in the set. The centre of the largest maximal ball is the farthest point from the boundary of the set. With Euclidean distance, the largest maximal ball is a sphere (it is the sphere itself). When the centre of the largest maximal ball is close to the boundary of the set, then every point is close to the boundary, as is the case in Figure 8.1(a) on the left.

The diameter of the largest maximal ball can be compared with two base characteristics of the viability study, as shown in Figure 8.2. When the diameter of the largest maximal ball is small compared with the diameter of the minimal bounding ball of the set, the system is very sensitive to even small

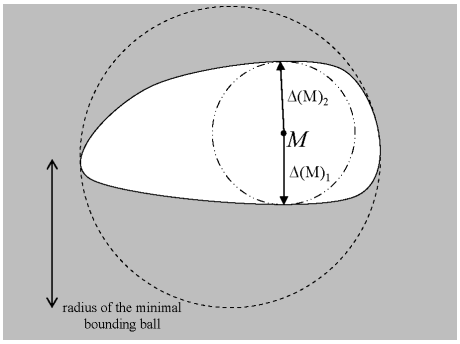

Figure 8.2: Geometric indicators for the viability kernel or the resilience basin: the radius of the kernel minimal bounding ball and the radius of the largest maximal ball inside the kernel. M is the centre of the maximal ball, and $\Delta(M)_1$ and $\Delta(M)_2$ are the sensitive disturbances at point M.

disturbances. In the case of a viability kernel or a resilience basin, this means that the system is not robust to the uncertainties in the state variables.

The size of the viability kernel can also be compared with the size of the constraint set. When the diameter of the minimal bounding ball of the viability kernel is small compared with the diameter of the minimal bounding ball of the constraint set, the dynamical system has to be restrained in a small area. This makes control much more difficult.

2.2 State robustness

When a system starts from a state inside the viability kernel, it is possible to maintain its evolution inside the constraint set with certainty, whereas if the system starts inside the resilience basin, it is possible to reach the viability kernel in finite time. These are the properties that characterize the viability kernel and the resilience basin. Nevertheless all states are not equal in regard to other respects: state utility, control cost, but also robustness to uncertainty or measurement error and robustness to perturbation. In Figure 8.1(b), the point on the left is far less robust than the point on the right.

DEFINITION 8.1 ***Robustness of a State and Sensitive Disturbance.***
Let K be a viability kernel or a resilience basin in the state space E (or in the extended phase space with control variables). Let x be a state in K. The robustness $s(x)$ of the state x is defined by:

$$s(x) = \max\{\alpha \geq 0;\ \forall y \in E,\ d(x,y) < \alpha \Rightarrow y \in K\}$$

. A minimal sensitive disturbance at point x, $\Delta(x)$ is defined by:

$$\Delta(x) = argmin_{\delta \in E}\{\|\delta\|,\ s(x+\delta) = 0\}$$

.

When a state is far from the boundary, it is actually tolerant to error in state determination, and in the same extent its robustness to perturbation is high. In Figure 8.1(b), the distance to the boundary is indicated by a dashed circle. It is obvious that the robustness value of the right state is greater than the robustness value of the left state. When a small perturbation is applied to the left point, it leaves the viability kernel. The point on the right can resist larger disturbances. Inside the viability kernel, where states are undifferentiated as to resilience, the concept of robustness is more appropriate to describe the situation, since the robustness of a state is the size of the largest perturbation that keeps the system in the viability kernel.

2.3 Trajectory Robustness

The distance map is used to define the robustness of a state, but it can also be used to propose a family of robustness definitions on trajectories. Since the distance map gives the distance to the boundary at each point, it is possible to use this information to define robustness indicators at the level of a trajectory. Several definitions can be proposed, depending on the risk perception of the manager.

DEFINITION 8.2 ***Geometric Robustness of a Trajectory.***
Let $x : t \longmapsto x(t)$ be a trajectory in the viability kernel or resilience basin K. We note Γ_K its boundary. Let f be a function from the set of continuous real-valued functions to $\mathbb{R}$. The f-geometric robustness value of u on the time interval $[0,T]$ is:

$$r_f(x) = f\left(\{t \in [0,T] \mapsto d(x(t),\Gamma_K)\}\right)$$

.

For example, the most risk-adverse indicator is the minimum of the robustness on the trajectory, with $f = \min$. But it has some drawbacks, since it does not take into account the time during which the robustness is low. So the function $f = \text{mean}$ can also be used as an average value to consider the robustness of a trajectory. (This definition has its own drawbacks, since a trajectory that leaves the set has a non-zero robustness).

DEFINITION 8.3 ***Min-robustness and Mean-robustness of a Trajectory.***
Let $x : t \longmapsto x(t)$ be a trajectory in the viability kernel or resilience basin K. We note Γ_K its boundary. The min-robustness value of x on the time interval T is:

$$r_-(x) = \min_{t\in[0,T]} \{d(x(t),\Gamma_K)\}$$

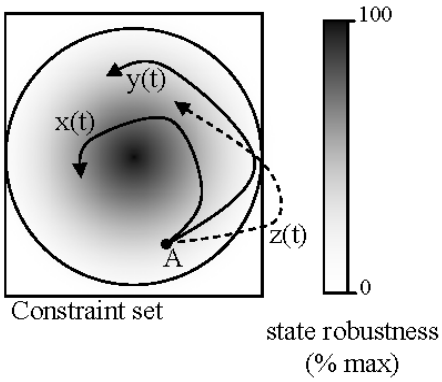

Figure 8.3: Geometric robustness. Trajectory $x(t)$ is more robust than trajectory $y(t)$. Trajectory $z(t)$ has a min-robustness value of 0, since it leaves the viability kernel.

The mean-robustness value of x on the time interval $[0,T]$ *is:*

$$r_m(x) = \frac{1}{T}\int_{t\in[0,T]} d(x(t),\Gamma_K)dt$$

.

Other definitions can be proposed with a discounting rate for future robustness values, in a similar way than is done for delayed rewards or payments (which are worth less than if they were paid at present time). For example, the control policy can take into account the fact that it is possible to modify a trajectory in the future. Therefore the robustness at the present time of a state reached in the future is less critical than the robustness of the present state. (The robustness value of a future state is considered to be less than the robustness value of the present state with a positive discount factor).

DEFINITION 8.4 ***Discounting Robustness.***
Let α *be a discount factor. The discounting robustness value of x on the time interval* $[0,T]$ *is:*

$$r_\alpha(x) = \frac{1}{T}\int_0^T d(x(t),\Gamma_K)e^{-\alpha t}dt$$

.

With these definitions, both trajectories x and y starting from A in Figure 8.3 have a strictly positive robustness value for all time intervals, and the robustness of x is greater than the robustness of y, for the min-robustness, the mean-robustness or the discounting robustness (actually all f-robustness with f monotone). The robustness of trajectory z is defined on $[0,T_1]$, where T_1

is the time it exits the viability kernel. Its min-robustness value on the time interval $[0, T_1]$ is zero.

All these definitions can be adapted to the discrete case. For example, the discounting robustness in the discrete case is:

$$r_\alpha(x) = \frac{1}{T}\sum_{t=0}^{T} d(x(t), \Gamma_K)\frac{1}{(1+\alpha)^t}$$

.

Trajectory robustness can be used to compare different evolutions of the system, starting from the same state point at some time t_0. For example, in Figure 8.3, a manager will disregard trajectory z, because it leaves the viability kernel. Since trajectory x dominates y for all robustness indicators, the manager should follow the sequence of control of trajectory x.

2.4 A Simple Example

The geometric approach can be better understood when it is applied to a simple example, like the problem of lake eutrophication (see Carpenter et al., 1999, for an extensive description).

Clear water or water in the oligotrophic state provides ecosystem services such as freshwater, irrigation supplies, etc. of much higher economic value than turbid water or water in the eutrophic state. But many lakes have experienced sudden shifts from oligotrophic to eutrophic states. Phosphorus is the most critical nutrient for the eutrophication of lakes. Excess phosphorus is imported by farms in the form of fertiliser and animal feed supplements. Most of the phosphorus accumulates in soil, and may then be transported to streams and lakes during runoff events associated with snow melt or rainstorms. Knowing the dynamics of lake eutrophication, the available regulatory laws, the present concentration of phosphorus in the lake and the present amount of inputs, the issue is to determine whether the lake can remain in an oligotrophic state or if it is doomed to become eutrophic in finite time. Simple models describe the lake time evolution with few variables: L, the amount of phosphorus inputs and P the phosphorus concentration in the lake.

Agriculture requires a minimum value for L, whereas oligotrophic state requires a maximum value for P. In this model, regulatory laws are constraints on $\frac{dL}{dt}$: this means that the law cannot set the maximum amount of phosphorus inputs, but rather imposes a decrease of the phosphorus inputs, for example by a percentage each year, with a maximum allowed.

With these simple assumptions, it is possible to compute the viability kernel for the lake eutrophication problem. The viability kernel (see Figure 8.4) is the subset of the (L,P)-plane that gathers all states (L,P) such that there exists at least one regulatory law that allows the oligotrophic state to be maintained (Martin, 2004).

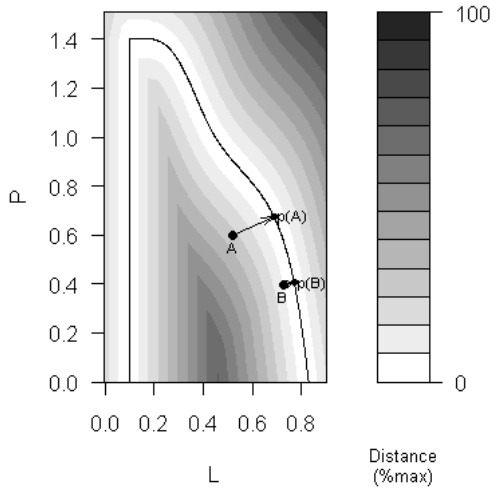

Figure 8.4: Distance map for the viability kernel boundary of the lake eutrophication problem, with the following constraints: $0 \leq P < 1.4$, $0.1 < L < 1$, $\left|\frac{dL}{dt}\right| \leq 0.09$. The black line is the viability kernel boundary. $p(A)$ and $p(B)$ show the direction and size of the minimal sensitive disturbance at point A and B, respectively.

The information about the distance to the viability kernel boundary is valuable because of measure uncertainties or exogenous disturbances (which, for instance, cause a sudden increase in phosphorus concentration). Figure 8.4 shows the viability kernel with the level curves of the distance to its boundary. The geometric study shows in what cases a state is dangerously close to the decision boundary. For example, point B stands very close to the boundary compare to A: it is less robust to perturbation.

The projection onto the boundary shows the direction of the most dangerous disturbance. Figure 8.4 shows that when the lake is in state B, a small increase of phosphorus input (L) combined with a very small change in the phosphorus concentration (P) can shift the state of the lake outside the viability kernel. This means that in a case of such a small disturbance, at some time in the future, whatever controls are applied, the lake will experience eutrophication.

The geometric study also gives global information concerning the problem. In Figure 8.5, point M is the centre of the largest maximal ball of the viability kernel. It is the farthest point from the boundary with $d(M,\Gamma) \approx 0.36$. The corresponding diameter $0.36 \times 2 = 0.72$ can be compared with the size of the

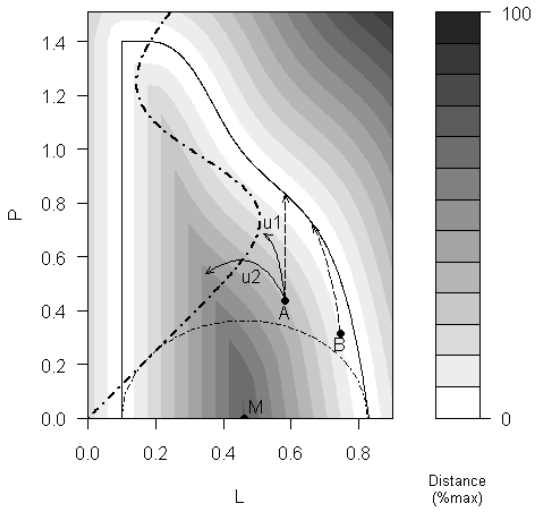

Figure 8.5: Trajectories for the lake eutrophication problem, corresponding to constant negative or null control. The dot dashed line is the set of equilibriums. Long dashed trajectories that leave the viability kernel are not robust. Point M is the centre of the largest maximal ball.

constraint set (0.9 along L). Because this viability kernel has a reasonable size, it should be easy to find an action policy that guarantees an oligotrophic state.

The geometric study gives information about trajectories. We consider controls of the type $\frac{dL}{dt} = -\alpha L(0)$, with $\alpha \geq 0$, $L(0)$ being the amount of phosphorus inputs at the initial time. Both trajectories u_1 ($\alpha = -5\%$) and u_2 ($\alpha = -15\%$) starting from A with a constant (negative) control have a strictly positive value, either for the min or the mean-robustness. Both min and mean-robustness values of u_2 are greater than the ones of u_1. The vertical trajectory coming from A is not robust for the min-robustness. It corresponds to the trajectory with no phosphorus input variation ($\alpha = 0$). If the initial state of the lake is at B, finding an efficient control policy is more difficult. The trajectory coming from B in Figure 8.5 has a null min-robustness value, since it leaves the viability set in twelve years. The control is already set to half the maximum allowed (-0.09/year), $\alpha = -6\%$ of the initial value $L(0)$ at B.

All these geometric indicators can also be used to define a particular action policy, for instance by maximizing the robustness as described in section 4. The most important information concerns the choice of a particular control among all possible viable controls. For example, at point B, a lot of control

values are still possible to keep the lake in a viable state. Generally a manager does not want to impose too strict regulation (for example, the maximum diminution allowed). In order to keep the robustness above a given threshold, the manager can anticipate and choose a viable control closer to the control available at the projection point $p(B)$. In the same way, if the manager wants to raise the robustness value of a lake at state A, he can propose a control close to the control available at the projection point $p(A)$.

In the next section we present the methods and tools that allow one to compute and use the distance map and projection to provide geometric information for a viability study.

3. Computing Geometric Robustness

The underlying concept of robustness is the distance to the boundary (of the significant sets of the viability study). It is used to define the robustness of states and trajectories, to describe the shape of the sets (with the largest maximal ball), to select appropriate control (through the projection point of a state). It is therefore essential to have at one's disposal an efficient method to compute the distance and projection. This is the subject of this section.

3.1 Practice of Geometric Method

In this section we develop the prerequisites that are often insufficiently taken into account when dealing with geometric methods. It is particularly important in the case of the study of viability since an efficient control policy depends on the result of the resilience and robustness study. Many choices are implicitly made before the computation of the distance, with impacts on the distance map. It is essential to underline the different options and their significance in order to really benefit from the geometric study. As for all geometric methods, the choice of the distance is a critical step of the geometric robustness study. A change of the distance can radically change the conclusion. For example, in Figure 8.1(a), a change from the Euclidean distance to:

$$d_2(x,y) = \sqrt{(3(x_1 - y_1))^2 + (\frac{1}{3}(x_2 - y_2))^2}$$

transforms the right circle shape into the left ellipse. Both shapes share the same area, but the radius of the largest maximal ball in the case of the ellipse is three times smaller than in the case of the circle. This problem arises frequently with multiple criteria weighting.

However, the geometric study takes place after the building stage of the model, therefore this issue is generally already fixed when the distance map algorithm is applied.

In any case, a reflection about the distance is essential to interact with the human experts in charge of the model.

Nondimensionalized variables. The viability kernel is a subset of $\mathbb{R}^n$, and the two main methods that are available to approximate the viability set generally perform homogenization and rescaling operation. So the distance map algorithm is generally applied to variables that are already nondimensionalized, and the distance induced by the inner product can be used directly. When it is not the case, model experts have to propose a suitable coordinate system in which the inner product is meaningful.

Two kinds of coordinate system transformation are basically useful, the Min-Max (MM) and the standard (s) transformation (see equation 8.1).

With the Min-Max transformation the model expert considers that a subset of the state space only has to be taken into account for the distance map transform. The range of every variable is reduced to the interval $[0,1]$ and the distance transform map applies to the unit hypercube. The main justification for the use of Min-Max transformation is that during the viability study, a constraint set in which the dynamical system should evolve is defined. This constraint set is a subset of the input state space. This set can be considered as the set of all possible initial states for the dynamical system, and then mapped to the unit hypercube for further use.

The Min-Max transformation is also particularly appropriate when the variables are in fact sensor measurements, since the accuracy of physical sensors is generally a function of the total range of values. For each attribute i the minimum and the maximum of the range are set to $Min(i)$ and $Max(i)$ and the new coordinate system is used to describe the state space.

The standard transformation is widely used in statistics to normalize sample data, using an estimate of the mean E_i and of the standard deviation s_i of each variable i. In the framework of viability study, $s(i)$ can be seen as a characteristic unit of measurement for variable i, and $E(i)$ a characteristic value for variable i. For example, $s(i)$ can be set to the range of the constraint set on variable i.

In practice, the choice of the transformation should be guided by the viability problem.

The new coordinate system is defined by (8.1).

$$y_i^{MM} = \frac{x_i - Min_i}{Max_i - Min_i} \quad \text{or} \quad y_i^s = \frac{x_i - E_i}{s_i}. \tag{8.1}$$

In this new coordinate system, the canonical inner product is meaningful.

Choice of a distance. When the input state space has an inner product, the Euclidean distance which is derived from the inner product is generally used to compute the distance map unless some particular property is required. But it is

not always the most appropriate. Other distances than the Euclidean distance can still be considered in order to compute geometric indicator for a viability kernel or a resilience basin. The use of a particular distance should be linked to the way the possible disturbances or uncertainties for each state variable can combine themselves.

$$\text{Euclidean distance} \quad d(x,y) = \|x-y\| = (\sum_i |x_i - y_i|^2)^{\frac{1}{2}} \tag{8.2}$$

$$L^1 \text{ distance} \quad d_1(x,y) = \|x-y\|_1 = \sum_i |x_i - y_i| \tag{8.3}$$

$$L^\infty \text{ distance} \quad d_\infty(x,y) = \|x-y\|_\infty = \sup_i |x_i - y_i| \tag{8.4}$$

With the Euclidean distance, the uncertainty (or error or possible perturbation) is distributed among all the variables: the amount of uncertainty sums over all the variables, following Pythagoras' theorem. The combined effect of a small disturbance a_i on every variable i is a larger disturbance a with size $\|a\| = (\sum_i |a_i|^2)^{\frac{1}{2}}$. With the L^1 (or Manhattan) distance, the combined effect of disturbances on different variables is simply their sum: $\|a\|_1 = \sum_i |a_i|$.

With the L^∞ (or sup norm) distance, the combination of perturbations of the same size on different variables does not change the size of the resulting perturbation: $\|a\|_\infty = \max_i |a_i|$. This means that every combination of disturbances are allowed, or, in other terms, the different sources of uncertainties or disturbances do not substitute for one another.

In the lake eutrophication problem, we used the Euclidean distance, since the two variables stands for a same chemical substance (coming from different sources). In other problems, with variables standing for different physical quantities, it can be preferable to use the sup norm distance. This is the case for example in the cheese ripening process (Mesmoudi et al., 2009), where state variables are rather different: temperature, mass, microorganisms respiration, etc.

3.2 Distance Map and Projection Algorithm

The method we use for computing the distance to the boundary was initially developed for classification systems (see Alvarez et al., 2010) for the general case and Alvarez, 2004 for a detailed illustration on machine learning decision trees). It applies well to viability kernels and resilience basins which can be seen as classifiers. (States inside the viability kernel belong to the class *viable*, whereas states outside belong to the class $\neg$*viable*. States inside the resilience basin belong to the class *resilient*, whereas states outside belong to the class $\neg$*resilient*).

The distance and projection maps are computed with an adapted version of a discrete algorithm coming from mathematical morphology (Meijster et al.,

2000). This algorithm is defined on a hyper rectangle of $\mathbb{N}^d$, and so the area of the state space for which the distance is computed has first to be mapped to a hyper rectangle in $\mathbb{N}^d$. This means in particular that two neighbours on any axis of the grid are at the same distance. The algorithm computes the exact distance of each point of the discretized space to a subset $\bar{S}$ of $\mathbb{N}^d$. We modified the original algorithm to compute the projection(s) onto $\bar{S}$. In our case, $\bar{S}$ is the set of non-viable or non-resilient states.

The algorithm 1 consists of two steps. The first step computes the distance and nearest point of the boundary on the first axis. It labels the points of the boundary if needed. The second step is called $d-1$ times and adds an axis to the previous subspace of dimension $k-1$. It updates the distance value and the list of projection points in the new subspace. To each norm is associated a function that specifies how the distance in dimension k subspace is computed from the distance in dimension $k-1$. For sake of simplicity, we consider that a unit hypercube of the input space is discretized onto a N points per axis d-dimensional grid G in $\mathbb{N}^d$.

For example, with the Euclidean distance, the square distance in dimension k subspace is given by a parabola: If $g_{k-1}(X)$ is the squared distance between X and some point of the boundary, computed on the first $k-1$ axes, and if u_k is the unit vector of axis k, then the squared distance in the dimension k subspace, $g_k(X)$ is at most $g_{k-1}(X)$. Applying the Pythagorean theorem, it is also at most $g_{k-1}(X - l.u_k) + (x_k - l)^2$ for $0 < l < N$, and the minimum value gives the result. But the computation and comparison of all these values is suboptimal.

The algorithm is optimal, since instead of computing all the distance values, it considers a set of distance functions and computes their lower envelope. For example, for the Euclidean distance, it considers the set F of parabolas:

$$\left\{F^X(i) = g_{k-1}(X) + (i - x_k)^2\right\} (0 \leq x_k < N)$$

The square distance for all $x_k, 0 \leq x_k < N$ is given by the lower envelope of F, that is, for each abscissa, the minimum value among all the values given by the different parabolas for this abscissa, as it can be seen in Figure 8.6.

The key point is the fact that two parabolas in F intersect at most once in $\mathbb{N}^2$, and that the intersection is very easy to compute: F^X intersects F^Y when $2(x_k - y_k)$ divides $(x_k^2 - y_k^2 + g_{k-1}(X) - g_{k-1}(Y))$. Instead of storing all the parabolas values, only the intersection points and the vertices are stored.

This is the reason why it is optimal: The computation of the envelope is reduced to a matrix searching problem, whose complexity is of $O(N)$ in this totally monotonic case (see Hirata, 1996 and Aggarwal et al., 1987 for details), so the overall complexity is in $O(N^d)$. The algorithm also works well for other distances, as long as this key point concerning the intersection of the building functions is maintained. The complementary set $\bar{S}$ of the viability kernel can be any subset of $\mathbb{N}^d$, so it contains $O(N^d)$ points. The complexity of algorithms

Algorithm 1 DistanceAndProjectionOnToSet.
Sketch of the distance map algorithm.

Require: a map from $G = [0,N-1]^d$ in $\mathbb{Z}^d$ to $\{0,1\}$. Points of the boundary are labeled by a function $label(x_1,...,x_d)$. A building function $F(X)(i)$ and an intersect function according to the *norm*.

Ensure: Distance of each point of $[0,N-1]^d$ to $\bar{S}$ and the corresponding nearest point of $\bar{S}$

Procedure **firstAxisDistance** `distance along the first axis`
for all $(x_2,...,x_d) \in [0,N-1]^{d-1}$ **do**
  for $i \leftarrow 0$ to $N-1$ **do**
    $d(i,l) = |i-l|$
    $A \leftarrow \{d(i,l),\ 0 \le l < N \text{ and } (l,x_2,...,x_d) \in \bar{S}\}$
    if $A = \emptyset$ **then**
      $g(i,x_2,...,x_d) \leftarrow \infty$
    else
      $j \leftarrow \text{argmin}\{A\}$
      $g(i,x_2,.,x_d) \leftarrow |i-j|$
      $p(i,x_2,.,x_d) \leftarrow \text{label}(j,x_2,.,x_d)$
    end if
  end for
end for
return g, p
end procedure **firstAxisDistance**
if *norm* = Euclidean **then**
  $g_1 \leftarrow g^2$ `post-processing for Euclidean distance`
end if
for $k \leftarrow 2$ to d **do**
  Procedure **AdditionalAxis** `example: Euclidean square distance`
  for $(x_1,...x_{k-1},x_{k+1},...,x_d) \in [0,N-1]^{d-1}$ **do**
    for $i \leftarrow 0$ to $N-1$ **do**
      `recruitment of building functions`
      $A \leftarrow \{F^{(x_1,.,l,.,x_d)}(i) = (d(i,l)^2 + g_{k-1}(x_1,.,l,.,x_d)\}$, with $0 \le l < N$
      if $\min_{0\le l<N}\{A\} < \infty$ **then**
        $j \leftarrow \text{argmin}_{0\le l<N} A$
        `assignment following the building functions envelope`
        $g_k(x_1,.x_{k-1},i,x_{k+1},.x_d) \leftarrow F^{(x_1,.,j,.,x_d)}(i)$
        $p(x_1,.x_{k-1},i,x_{k+1},.x_d) \leftarrow p(x_1,.x_{k-1},j,x_{k+1},.x_d)$
      end if
    end for
  end for
  end procedure **AdditionalAxis**
  if $k = d$ **then**
    if *norm* = Euclidean **then**
      $g_d \leftarrow \sqrt{g_d}$
    end if
  end if
  return g_d, p
end for

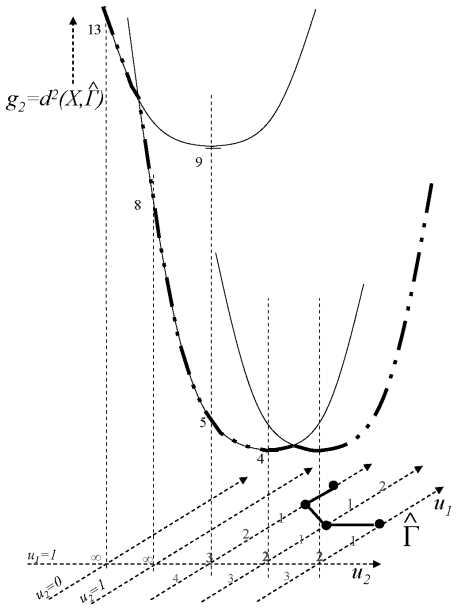

Figure 8.6: Illustration of Algorithm 1. The distance along the first axis (in grey) is computed first. Then the parabolas are built along the second axis. Here, for $u_1 = 1$, along the second axis with $0 \leq u_2 \leq 3$, two parabolas are recruited with respective vertices $(2,9)$ and $(3,4)$. The dot-dashed line is their lower envelope. The second parabola is lower than the first one from point $u_2 = 1$, since the intersection is between 0 and 1. At the next step, ($u_2 = 4$), another parabola will be recruited with vertex $(u_2 = 4, 2^2)$. Then the square distance will be assigned backwards, following the lower envelope: 4 at $u_2 = 4$, 4 at $u_2 = 3$, 5 at $u_2 = 2$, 8 at $u_2 = 1$ and 13 at $u_2 = 0$.

which consider the distance of the different points of the state space to each of $\bar{S}$ points is therefore much higher.

For the Euclidean distance, the building function and the intersect function are defined by:

$$F^X(i) = g_{k-1}(X) + (i - x_k)^2 \tag{8.5}$$

$$\text{intersect}(F^X, F^Y) = \text{truncate}(x_k^2 - y_k^2 + g_{k-1}(X) - g_{k-1}(Y)) \div 2(x_k - y_k). \tag{8.6}$$

For the sup norm, the distance in dimension k subspace is computed from the distance in dimension $k-1$ with a truncated V-shaped function:

$$F^X(i) = \max(g_{k-1}(X), |i - x_k|) \tag{8.7}$$

The V-shaped function has its vertex at abscissa x_k and is truncated at the value $g_{k-1}(X)$, since the distance of the sup norm in dimension k is the max of the distance for the first $k-1$ axes and of the distance along axis k, that is $|i-x_k|$ at step i. The intersection of truncated V-shaped functions F^X and F^Y is given by the following formula (with $x_k \leq y_k$):

$$\text{intersect}(F^X, F^X) = \begin{cases} \max((x_k + g_{k-1}(Y)), ((x_k + y_k) \div 2)) \\ \qquad\qquad\qquad\qquad \text{if } g_{k-1}(X) \leq g_{k-1}(Y) \\ \min((y_k - g_{k-1}(X)), ((x_k + y_k) \div 2)) \text{ otherwise} \end{cases} \tag{8.8}$$

This algorithm is very efficient, since the complexity is in $O(d.N^d)$, which is optimal. It can also be parallelized on up to N^{d-1} processors. For example, the computation of the distance map for 10^9 points (20 points per axis in 7 dimensions, or 1000 points per axis in 3 dimensions) takes about 3 hours on a 2.4 GHz processor (see Alvarez et al., 2010 for more details).

4. Robust Viability-Guided Management

We illustrate some viability and robustness guided policies using one of the language competition models, the bilinguals Minett-Wang model, presented in detail in chapter 3.

4.1 Language Competition Model Description

In the bilinguals Minett-Wang model, the population is made of three groups, the monolingual speakers of languages X and Y, and the bilingual speakers B. The model is two-dimensional, with the dimensions representing the proportions of X and Y speakers, on the x and y axis, respectively ($b = 1-x-y$). The evolution of these two variables are governed by the following equations:

$$\begin{array}{rcl} \frac{dx}{dt} & = & (1-x-y)(1-y)^a s - xy^a(1-s) \\ \frac{dy}{dt} & = & (1-x-y)(1-x)^a(1-s) - yx^a s \end{array} \tag{8.9}$$

where $s \in [0,1]$ denotes the prestige of language X compare to the language Y one, and a is a parameter that models how the attractiveness scales with the proportion of speakers (for more details see chapter 3).

The prestige measures the status associated with a language due to individual and social advantages related to the use of that language, being higher according to its presence in education, religion, administration and the media. We assume that public action can modify the prestige of a language, but that its variation at each time step is bounded:

$$\begin{array}{rcl} \frac{ds}{dt} & = & u \\ u & \in & U := [-\bar{u}; \bar{u}]. \end{array} \tag{8.10}$$

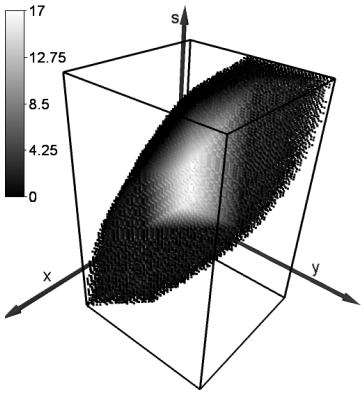

Figure 8.7: Distance map of the language viability domain. A grid of 100 points per axis is mapped on the unit hypercube. Only the points of the viability domain are drawn. Their colour is a function of the Euclidean distance to the boundary. A black bounding box encloses the viability domain.

The problem of maintaining a given level of monolingual speakers in both languages can be described by a subset of the state space, the constraint set, K, in the viability theory terminology:

$$K := [x_{min}; 1] \times [y_{min}; 1] \tag{8.11}$$

with $x_{min} > 0$ and $y_{min} > 0$.

It is possible to exhibit viability domains (see Bernard and Martin, submitted) associated with the viability problem described by (8.9), (8.10) and (8.11). (A viability domain is a subset of the viability kernel, similar to it, but it is not maximal).

With this model, the variables are already normalized and the dot product is obvious. Since the boundary of the viability domain is computed directly (Bernard and Martin, submitted), the distance algorithm computes an approximation of the distance to the boundary on a grid with no further conditions. Figure 8.7 shows a transparency view of the distance map inside the viability domain.

The distance map is then used to propose viable and robust policies.

4.2 Robustness to Perturbation and Uncertainty

Besides the viability kernel, viability theory gives at each state a list of controls that ensures that a state can stay in the viability kernel one step ahead. Nevertheless, all states in viability kernel are not equivalent, since a system

in a viable state near the boundary can switch outside the viability kernel if subjected to unexpected perturbation. The distance map can help to define a more robust control policy than the standard viability control policy, which is based on the inertia-based avoidance control strategies. The same situation occurs in the resilience basin, and the distance map can help to take into account unexpected perturbations.

The following examples illustrate robust control policy in the case developed in section 4.1. For simplicity we consider the viability domain, but the same approach would apply in a resilience basin (when several controls are available).

Inertia-based Avoidance Control Strategies.

Heavy Trajectories. Heavy trajectories (Aubin, 1991) correspond to the choice at each time step of the control that minimizes the norm of the control rate of change. This means that the control stays the same until it is necessary to change it to avoid leaving the viability kernel. Figure 8.8 shows an example of several heavy trajectories with random input state and control in the viability kernel. In general, with the control based on the Saint-Pierre algorithm, these trajectories follow the flow of the dynamic system with constant control until they reach the boundary of the viability kernel. On the boundary of the viability kernel, when it does not coincide with the boundary of the constraint set, all the viable velocities belong to its tangent space. Consequently, the trajectories are stuck to the boundary until they encounter an area where the viability kernel coincides with the boundary of the constraint set. This is the reason why these trajectories stay a long time on the boundary, as shown in Figure 8.9. These trajectories are different from the ones observed in the case studies of this book because the tool we used to compute the control actions prevents the system to get too close to the boundary (see Chapter 7).

Slow Trajectories. Slow Trajectories (Aubin, 1991) correspond to the choice at each time step of the control with the smallest possible norm. This means that the system is not controlled except when it is necessary. Figure 8.10 shows a set of slow trajectories. In general these trajectories follow the flow until they reach the boundary of the viability kernel. As with heavy trajectories, they are then stuck to the boundary until they encounter an area where the viability kernel coincides with the boundary of the constraint set. Slow trajectories, like heavy trajectories, stay a lot of time on the boundary, as shown in Figure 8.11.

Standard control policies lead to trajectories that can stay for a long time very close to the boundary of the viability kernel. This problem cannot be easily solved by strengthening the constraints: being inside the viability kernel

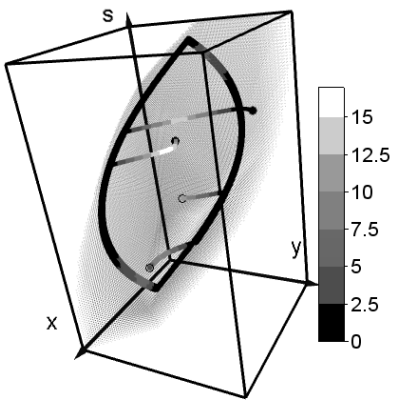

Figure 8.8: State robustness of 4 heavy trajectories with random initial state (black circles). The colour of the points depends on the robustness value. The viability kernel is shown as a cloud of small points. The black curved lines correspond to trajectories on the boundary: Trajectories often stay on the boundary.

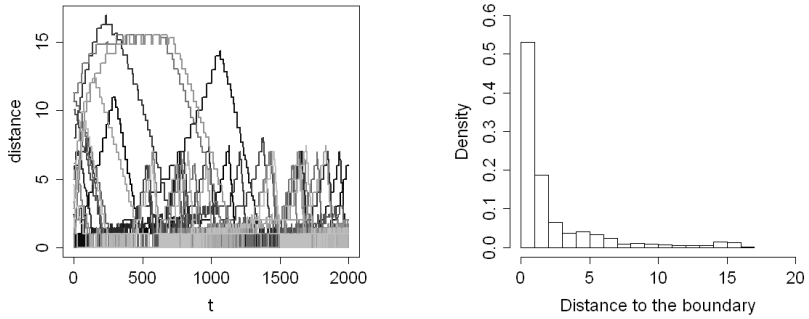

Figure 8.9: Heavy trajectories for the language competition model. Left: Distance to the boundary as a function of time. Right: Mean frequency (density) of the distance to the boundary. Heavy trajectories stay on the boundary more than half the time.

of a smaller constraint state ensures that, for given perturbation strengths, the system state remains inside the initial constraint set, but not in the viability kernel. So, without a complete study of the resilience basin of the new viability kernel, there is no guarantee that a trajectory, even in the more constrained viability kernel, will resist unexpected perturbation.

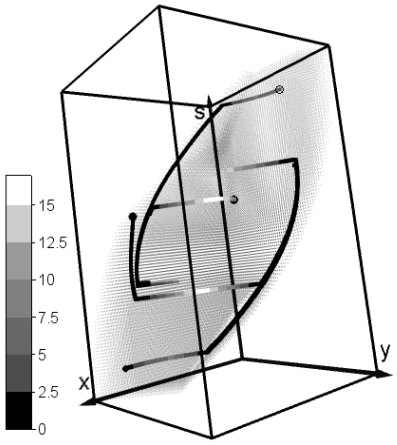

Figure 8.10: State robustness of slow trajectories. Black lines are on the boundary of the viability domain: Slow trajectories stay very often on the boundary.

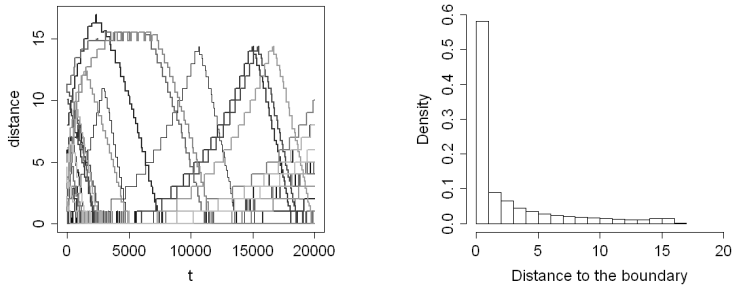

Figure 8.11: Slow trajectories for the language competition model. Left: Distance to the boundary as a function of time. Right: Mean frequency (density) of the distance to the boundary.

Geometric-based Control Strategies. The main objective of a geometric-based strategy is to propose control policies that are robust to unexpected perturbation or uncertainties in the state variables. The same concern is addressed in chapter 7. The principle is then not only to follow viable (or resilient) strategies but also to remain far from the boundary of the viability kernel (or resilience basin) if possible.

For this purpose we design a strategy we call “Avoidance with Threshold”. This implies a modification of the heavy strategy in the viability kernel. The control remains constant until the state robustness comes below a given threshold (for the heavy strategy, the value of the threshold is zero). When the ro-

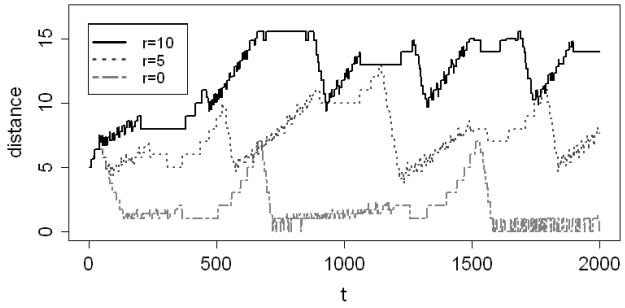

Figure 8.12: Trajectories from the same initial state point with different robustness threshold values (r). The avoidance with threshold strategy keeps the system away from the boundary.

bustness reaches the threshold, the control is still viable, but for sake of anticipation, the control corresponding to the projection point of the current state on the viability kernel boundary is applied. The idea behind this heuristic is that the control that applies on the boundary takes the flow into account.

A variant of this method is applied in chapter 7, selecting the control that optimizes the value of the SVM several steps ahead.

The robustness-based strategy, in order to be efficient, needs a fast and reliable way to access to the distance and projection on the boundary. This is provided by the distance map algorithm described in section 3.2.

Figure 8.12 displays the distance to the boundary of the viability kernel as a function of time for three trajectories that start at the same initial point but follow avoidance strategies with different threshold values. It shows that the avoidance strategy is very efficient: The heavy strategy (avoidance strategy with threshold equal to 0) governs a trajectory whose distance to the boundary of the viability kernel is often equal to 0. The avoidance strategies with strictly positive thresholds govern trajectories whose distance to the boundary of the viability kernel never crosses the threshold once this threshold is reached.

To underline the efficiency of the avoidance strategy, we perform different numerical experiments. We choose randomly different initial points and compute the trajectories governed by avoidance strategies with different thresholds including zero. For each trajectory, we compute the distance to the boundary of the viability kernel as a function of time and then the mean relative frequencies of this distance over all trajectories for each different strategy. Figure 8.13 shows the results computed over 24 trajectories. Most of the time, the trajectories are above the distance threshold. The frequency of the distance values smaller than the threshold are not null because the randomly chosen ini-

tial point may have a distance to the boundary smaller than the threshold. But once the distance to the boundary passes over the threshold it never returns below it. The avoidance strategy with threshold is very effective for the language competition model.

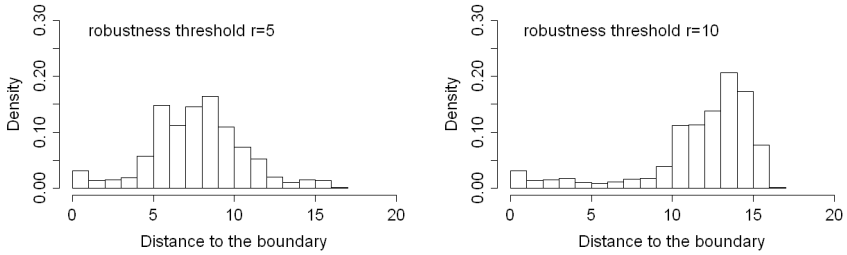

Figure 8.13: Mean frequency of the distance to the boundary with different robustness threshold values.

Finally, we go back to the trajectory robustness definitions proposed in section 2.3, and we compute the different robustness values of trajectories starting at the same point but following avoidance strategies with different thresholds. Figure 8.14 shows the results for different starting points. Except for the min-robustness, where both values can be equal to 0, the robustness of the trajectories with threshold is always strictly above the corresponding robustness of the standard heavy trajectory.

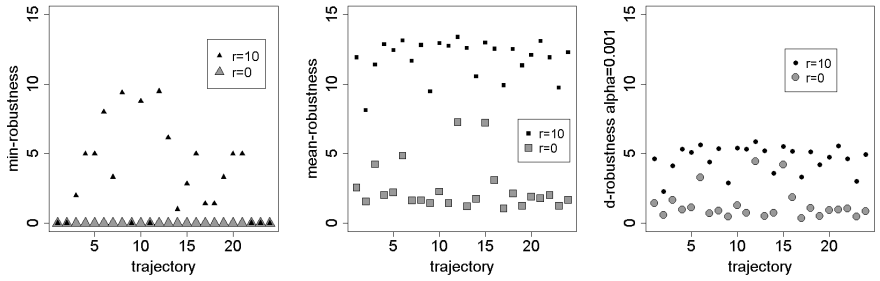

Figure 8.14: Comparative robustness of trajectories stemming from the same input point, with and without distance threshold.

Remarks. It is worth noting that the same approach can be followed for slow trajectories.

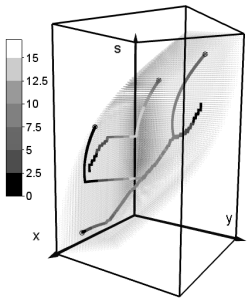

Figure 8.15: Trajectories with control directed towards the centre of the largest maximal ball. Points of the viability kernel are drawn as small dots. Trajectories are thick lines. Colour is a function of the distance to the boundary.

Geometric information can also be used to propose directly a control that aims at maintaining the system far from the decision boundary. For example, in the language competition model, it is possible to propose a control directed towards the centre of the largest maximal ball, as shown in Figure 8.15. This type of control changes radically the pattern of the trajectories, which can stay much longer in the white area (the farthest area from the boundary). However, this strategy can be unsuccessful depending on the flow. Some trajectories have a low robustness value despite the attempt to pull the trajectory towards the centre of the largest maximal ball. They cannot evolve anymore and stay in a low robustness state.

5. Conclusion

The definition of the viability kernel ensures the existence of an action policy that keeps the state of the system inside the constraint set. Analogously, for states inside the resilience basin, there exists at least one policy that allows the system to reach the viability kernel in finite time. However, what happens if a perturbation occurs and causes a jump of a given length of the system state? Does it remain in the viability kernel or in the resilience basin? The response to these questions is given by the distance to the boundary of these sets.

To compute an approximation of this distance on a regular grid, we have used a distance transform algorithm. This information associates each state belonging to the viability kernel or the resilience basin with its distance to the boundary and gives information about the robustness of this system state to perturbations. Besides, geometric robustness of the states inside the viability kernel or resilience basin can also be used to propose several definitions of trajectory robustness, which can be used in turn as new optimization criteria.

Heavy or slow control policies minimize at each time step the control norm or rate of change (which can be seen as control costs). These standard control policies lead to strategies lacking in robustness: trajectories can stay for a long time very close to the boundary of the viability kernel.

Geometric information, taking into account the distance to the set boundary, can be proposed to adapt the standard control strategy, in order to maintain the system away from the viability kernel or resilience basin boundary. We have proposed a geometric-based strategy: when the state robustness comes below a threshold, the control corresponding to the projection point of the current state is applied, even if the present control is still viable (or resilient). To implement such a strategy, we have modified the distance transform algorithm to include the approximation of the projection onto the boundary. And we have shown that it does indeed lead to more robust trajectories in the language competition model.

Other strategies could also be proposed, taking into account more local geometric information, such as the distance to the skeleton. In the case of a resilience basin, the robustness could be taken into account as well as the resilience value, to select the appropriate control to drive the system back to the viability kernel. (The geometric robustness could be taken into account even more directly in the definition of the cost function that is used to define the resilience value).

6. References

Abrams, D.M. and Strogatz, S.H. (2003). Modeling the dynamics of language death. *Nature*, 424:900.

Aggarwal, A., Klowe, M., Moran, S., Shor, P., and Wilber, R. (1987). Geometric applications of a matrix-searching algorithm. *Algorithmica*, 2:195–208.

Alvarez, I. (2004). Sensitivity analysis of the result in binary decision trees. In *Proceedings of the 15th European Conference on Machine Learning*, volume 3201 of *Lecture Notes in Artificial Intelligence*, pages 51–62. Springer-Verlag.

Alvarez, I., Martin, S., and Mesmoudi, S. (2010). Describing the result of a classifier to the end-user: Geometric-based sensitivity. In *proceedings of the 22d European Conference on Artificial Intelligence*, pages 835–840. IOS Press.

Aubin, J.P. (1991). *Viability Theory*. Birkhauser, Basel.

Bernard, C. and Martin, S. (Submitted to). Building strategies to ensure language coexistence in presence of bilingualism. *Applied Mathematics and Computation*.

Carpenter, S.R., Ludwig, D., and Brock, W.A. (1999). Management of eutrophication for lakes subject to potentially irreversible change. *Ecological Applications*, 9:751–771.

Castello, X., Eguíluz, V.M., and San Miguel, M. (2006). Ordering dynamics with two non-excluding options: bilingualism in language competition. *New Journal of physics*, 8:308–322.

Crystal, D. (2000). *Language Death*. Cambridge University Press, Cambridge.

Deffuant, G., Chapel, L., and Martin, S. (2007). Approximating viability kernels with Support Vector Machines. *IEEE Transactions on automatic control*, 52(5):933–937.

Hirata, T. (1996). A Unified Linear-Time Algorithm for Computing Distance Maps. *Information Processing Letters*, 58(3):129–133.

Martin, S. (2004). The cost of restoration as a way of defining resilience: a viability approach applied to a model of lake eutrophication. *Ecology and Society*, 9(2).

Matheron, G. (1988). *Examples of topological properties of skeletons*, pages 217–257. Image Analysis and Mathematical Morphology. Academic Press.

Meijster, A., Roerdink, J., and Hesselink, W.H. (2000). A General Algorithm for Computing Distance Transforms in Linear Time. *Morphology and Its Applications to Image and Signal Processing*, pages 331–340.

Mesmoudi, S., Alvarez, I., Martin, S., Sicard, M., and Wuillemin, P.-H. (2009). Geometric analysis of a capture basin: Application to cheese ripening process. In *European Conference on Complex Systems*.

Minett, J. and Wang, W. (2008). Modelling endangered languages: the effects of bilinguism and social structure. *Lingua*, 118:19–45.

Saint-Pierre, P. (1994). Approximation of the viability kernel. *Applied Mathematics & Optimisation*, 29:187–209.

Saltelli, A., Chan, K., and Scott, M. (2000). *Sensitivity Analysis*. Wiley.

Serra, J. (1988). *Image Analysis and Mathematical Morphology*. Academic Press.

Wang, W.S.-Y. and Minett, J.W. (2005). The invasion of language: emergence, change and death. *Trends in Ecology and Evolution*, 20(5):263–269.

Appendix: Mapping the exact or approximate viability kernel or resilience basin onto a discrete grid

The DistanceAndProjectionOnToSet algorithm (1) provides the exact distance map to a discrete subset $\bar{S}$ of $\mathbb{N}^d$. It is relevant to use it to compute an approximation of the distance to a viability kernel or a resilience basin with some general hypotheses. Depending on the method that is used to define explicitly the viability kernel or the resilience basin, an approximation of its boundary can be computed. In the other cases, an approximation of the set itself is computed

by the viability or the SVM approximation algorithm. The approximation of the distance is not the same in this case.

Case Where the Boundary is Known. When the subset $\bar{S}$ of $\mathbb{N}^d$ is an approximation $\hat{\Gamma}$ of the true boundary Γ, the algorithm computes directly the exact distance of the points of the grid $G = [0, N-1]^d$ to $\hat{\Gamma}$.

For simplicity, we assume that Γ is included in the unit hypercube H of E. We define G_N the regular grid with N points per dimension: G_N has N^d elements $\hat{x} = (k_1, \ldots, k_d)$, $k_i \in \{0, N-1\}$ corresponding to $x = (k_1/(N-1), \ldots, k_d/(N-1))$ in E. The discretized boundary $\hat{\Gamma}_N \subset G_N$ is defined as follows:
Let $\hat{x} \in G_N$ and let x be its corresponding point in E,

$$\hat{x} \in \hat{\Gamma}_N \text{ if and only if } d(x, \Gamma) \leq \frac{\sqrt{d}}{N-1} \tag{8.A.1}$$

($\sqrt{d}$ is the diagonal length of the unit hypercube and $(N-1)$ the rescaling coefficient).

THEOREM A.1 *Let $y \in H$ and $\hat{x}_N$, the nearest point of y in G_N. Then the exact distance of $\hat{x}_N$ to the discretized boundary $\hat{\Gamma}_N$, approximates the distance of y to the viability kernel boundary Γ in H, $\Gamma \cap H$ as N goes to infinity.*

Proof:

Let $P(y)$ be the projection of $y \in H$ onto $\Gamma \cap H$ and $\hat{P}(\hat{x}_N)$ be the projection of $\hat{x}_N \in G_N$ onto $\hat{\Gamma}_N$. For the Euclidean distance (but similar inequalities exist for other distances), let $\varepsilon_N = \frac{\sqrt{d}}{N-1}$. By construction of $\hat{\Gamma}_N$, there exists $\hat{P}(y) \in \hat{\Gamma}_N$ such that $d(\hat{P}(y), P(y)) \leq \varepsilon_N$ and $P(\hat{x}_N) \in \Gamma \cap H$ such that $d(\hat{P}(\hat{x}_N), P(\hat{x}_N)) \leq \varepsilon_N$. Then, thanks to triangular inequality,

$$\begin{aligned} d(y, \Gamma \cap H) &\leq d(y, P(\hat{x}_N)) \\ &\leq d(y, \hat{x}_N) + d(\hat{x}_N, \hat{P}(\hat{x}_N)) + d(\hat{P}(\hat{x}_N), P(\hat{x}_N)) \\ &\leq d(\hat{x}_N, \hat{P}(\hat{x}_N)) + 2\varepsilon_N \end{aligned}$$

For the same reason, $d(\hat{x}_N, \hat{P}(\hat{x}_N)) \leq d(y, \Gamma \cap H) + 2\varepsilon_N$ ∎

When we consider the Euclidean distance, it is also possible to approximate the Euclidean projection onto the viability kernel boundary.

THEOREM A.2 *Let $y \in H$ and $\hat{x}_N$, the nearest point of y in G_N. Then the projection of $\hat{x}_N$ onto the discretized boundary $\hat{\Gamma}_N$, approximates the Euclidean projection onto the viability kernel boundary in H, $\Gamma \cap H$, outside the skeleton[1] of the connected parts of the class areas, as N goes to infinity.*

Proof:

Let $P(y)$ be the projection of $y \in H$ onto $\Gamma \cap H$ and $\hat{P}(\hat{x}_N)$ be the projection of $\hat{x}_N \in G_N$ onto $\hat{\Gamma}_N$. $(\hat{P}(\hat{x}_N))_N$ is a bounded sequence, so it has a convergent subsequence toward a point Q. Since $d(\hat{P}(\hat{x}_N), \Gamma) \leq \frac{\sqrt{d}}{N-1}$, $Q \in \Gamma$. Moreover, from Theorem A.1, $d(y, Q) = d(y, P(y))$. Since y doesn't belong to the skeleton, $P(y) = Q$.∎

Case Where Only the Viability Set is Known. In this case the viability algorithm or the SVM approximation computes an approximation S of the viability set. The boundary of the viability set Γ is not explicitly defined. Nevertheless, instead of defining the discretized boundary, we can defined the discretized complementary set of S, $\hat{\bar{S}}$ in G:

$$\hat{x} \in \hat{\bar{S}}_N \text{ if and only if } d(x, \bar{S}) \leq \frac{\sqrt{d}}{N-1} \tag{8.A.2}$$

Then, for the same reason as above,

THEOREM A.3 *Let $y \in H$ and $\hat{x}_N$, the nearest point of y in G_N. Then the exact distance of $\hat{x}_N$ to the discretized set $\hat{\hat{S}}_N$, approximates the distance of y to the viability kernel boundary ∂S in H, $\partial S \cap H$ as N goes to infinity.*