Project no: 043268, acronym: PATRES

PATTERN RESILIENCE

Thematic Priority: New and Emerging Science and Technology
Research Topic: Tackling Complexity

Deliverable D3.1: Report on methods and tools for computing resilience.

Start date of the project: 01/02/2007 End date: 31/01/2010
Revision: Version 0 Date: March 2010

1 Introduction.

The objective of this report is to summarise the main developments and conclusions of the project related to the methods and tools to compute resilience. One of the main motives of the project was to investigate the mathematical definition of resilience proposed by Martin [5], based on viability theory. In this report, we only rapidly scan the main points of the project achievements in this respect, referring for the details to the different chapters of the book derived from the project (to be published in 2010).

A part of our work has been devoted to the elaboration of a better expression of this viability based definition of resilience, more closely related to the existing literature about resilience. Moreover, we applied this definition to an example of savanna dynamics, to which a more usual definition of resilience had been already applied. Hence we could argue more directly about the advantages of the viability based definition. This work is presented in chapters 2 and 3 of the book. We summarise these points in sections 2 and 3 of this report.

In practice, this view of resilience relies on the possibility to compute viability kernels and resilience indices. A set of algorithms approximating viability kernels is available. Their principle is roughly the same: they build a series of smaller and smaller sets, converging to the viability kernel. At each step, the next set is built from the previous one by removing the parts for which no control can keep the points inside the set at the next time step. A particular direction of work that we developed in the project was related to the idea to represent the sets using machine learning techniques, and more particularly "'support vector machines"'. We developed a software prototype called "'Kaviar"' implementing the corresponding algorithms, which is a deliverable of the project. We synthesize the main points of this research in section 4 of this report, and chapter 10 of the book provides more details.

In section 5 of this report, we summarise the main methodological issues that we addressed in the case studies. They concern:

- the case study on savanna for which there was a problem in the definition of the variables representing the pair correlation.
- the case study on bacteria where we had to simplify the representation of correlation functions, in order to put them in a 2 dimension space.

Finally, in section 6, we report an ongoing work about the geometric robustness that can be derived from the viability based approach.

2 Background: Engineering and ecological resilience.

The literature about resilience is abundant and covers many disciplines, from physics to ecology and social sciences. We focus our review on the work of a particularly influential community called "resalliance", and we propose a critical assessment of some of the concepts and options proposed by this group. The chapter of the book presents some of these concepts which are related to resilience such as "adpative cycles" and "panarchy". But in this report, we focus on their work more directly about resilience, distinguishing between "engineering" and "ecological" resilience:

- 'Engineering resilience' is the "Rate and speed of return to preexisting conditions after disturbance" [3], which is sometimes called "elasticity". This approach supposes that the system can be defined as a dynamical system. Calculating resilience is based on linear stability analysis, as far as the dynamical system is defined by ordinary differential equations: calculate equilibria; apply infinitesimally small displacements, or disturbances, from the equilibrium; use Taylor expansions to obtain a set of linear equations describing the dynamics of the displacements; calculate the eigenvalue of the matrix of coefficients of the linearized set of equations. If the real part of the eigenvalue is smaller than zero, the disturbed system will return to its equilibrium. Thus, the sign of the eigenvalue's real part indicates whether the system is resilient, and the inverse of its absolute value is a measure of its elasticity, or engineering resilience. For individual-based system, the the computation of the engineering resilience is based on a simulation study as in [6] and [7] (see [2], [5] and chapter 2 of the book for reviews).

- In a highly influential review, Holling [4] suggested a more holistic definition. In the middle of the 1990s, a group of ecologists and social scientists founded the Resilience Alliance (RA; www.resalliance.org). The RA has the declared aim to promote and develop Holling's notion of ecological resilience and related concepts because they are considered essential for solving vital socio-ecological problems and for fostering sustainability. The resilience definition currently preferred by the RA was formulated by Holling [4] "Resilience is the capacity of a system to absorb disturbance and reorganize while undergoing change so as to still retain essentially the same function, structure, identity, and feedbacks."

The three main differences of the Resilience Alliance notion of resilience to engineering resilience are: (1) A shift from equilibrium to a dynamical regime

including feedbacks; an ecological system is rarely at its equilibrium, it is more often far from it, relating to external events (such as rainfalls for savanna for instance). (2) A shift in focus from numerical values of state variables to 'relationships', i.e. to the internal organization of ecosystems which give rise to their properties. (3) A shift in focus from the ability to recover after disturbance (engineering resilience) to the ability to 'absorb' the effect of disturbances, i.e. not to change essentially in the first place.

3 Viability based resilience: towards an operational definition of ecological resilience.

In chapter 3 of the book, we present in more details the definition of resilience which is at the centre of the project: the viability based definition of resilience [5]. We show how this definition can be seen as an extension of the previous "engineering" definition of resilience, towards some aspects of the ecological definition of resilience (without matching all its aspects). Indeed, in the engineering definition (that we also call "attractor based"), the system is defined by a differential equation ruling the dynamics in the state space, $x(t)$ being the state of the system at time t:

$$\begin{cases} x'(t) &= f(x(t)), \\ x(t_0) &= x_0. \end{cases} \tag{1}$$

Such a system generally has a set of stable equilibria, that can be called attractors of the dynamics, because the dynamics tends to be attracted towards these sets. The engineering definition of resilience is based on the analysis of such a dynamical system. A value should be associated to the different attractors: some are desirable, others are not. The resilient states are the ones which lie in the attraction basins of "good" attractors. The non-resilient states lie in the attraction basins of "bad" attractors. As explained previously, an intensity of resilience can be associated with a given attractor, by computing the "speed" of return to this attractor. Note that this value is the same for the whole attraction basin.

The viability based approach introduces not only the state of the system, but also possible actions on it. Hence we get a system of the following type:

$$\begin{cases} x'(t) &= f(x(t), u(t)), \\ x(t_0) &= x_0. \end{cases} \tag{2}$$

In this equation, $u(t)$ is a management action that is applied to the system at each time step. In such a system, the definition of an attractor or an equilibrium is more

difficult, because the trajectory depends on the choices of the management actions at each time step. Hence for a given starting point, there is an infinity of potential trajectories. Consequently, in this approach, the desired property that one wants the system to keep is defined without considering any attractor of the system. It is simply defined as a subset of the state space, in which one wants the system to remain.

Changing this definition of the desired properties of the system changes the mathematical framework for defining resilience, and viability theory is the appropriate one ([1]). Indeed, viability theory focuses on dynamical systems which cease to function or badly deteriorate when they cross the limits of a subset of their state space, called the viability constraint set. Hence the problem addressed is to keep the system within the limits of this viability constraint set. The problem is formally the same as the resilience one, simply, the viability constraint set is replaced by the desired set. The concepts and tools from viability theory can be used directly.

We use two main concepts derived from viability theory: "viability kernel and capture basin". The viability kernel ([1]) gathers all states from which there exists at least one action policy that keeps the system indefinitely inside K. More formally, if we denote $S_f(x, t, u(.))$ as the state reached after time t, starting from state x and applying action policy $u(.)$, with the controlled dynamical system (2), the viability kernel is defined by equation (3).

$$Viab_{f,U}(K) := \{x \in K \text{ such that } \exists u(.) \text{ with } \forall t > 0; S_f(x, t, u(.)) \in K\} \quad (3)$$

The capture basin of target set C, denoted $Capt_{f,U}(C)$ is the set of states from which there exists a management policy leading the system into target set C. More formally, the definition of the capture basin is:

$$\begin{aligned} Capt_{f,U}(C) \quad = \quad & \{x \in \mathbb{R}^n \text{ such that } \exists u(.) : t \to u(t) \in U \text{ measurable} \\ & \exists t^* > 0 \text{ with } S_f(x, t^*, u(.)) \in C\} \end{aligned} \quad (4)$$

Within this framework, in the viability based definition of resilience, the set of resilient states is the capture basin of the viability kernel. Indeed, this set contains the states for which there exists a management policy driving back the system into the viability kernel, where it is then possible to keep the desired property indefinitely (in the absence of perturbations). The property cannot be restored from the states outside this capture basin, and their resilience is therefore null. The

states belonging to the capture basin have a resilience index that is strictly positive and which may be quantified by the inverse of the time necessary to restore this property, or even with a more elaborate cost function ([5]).

We show that this viability approach based definition of resilience includes the attractor based definition as a particular case. Moreover, it offers new possibilities:

- The approach addresses the question of choosing management actions in order to keep or restore the desired property, and allows one to compute policies of action to keep or restore the desired property of the system,
- The desired property can be defined as a subset of the state space which does not include any attractor of the dynamical system. In this case, to keep the desired property, one should act regularly on the system. This corresponds to usual situations of ecological or social systems, which are impossible to address in the attractor based framework of resilience.

4 Algorithms for computing viability kernels and resilience indexes: Kaviar software

One interest of the viability based definition of resilience is the availability of algorithms for computing the viability kernels and resilience indexes. Actually, these algorithms compute only approximations of these values, which then are used to compute policies of actions. These approximations use a discrete set of points covering the state space (for instance a grid) and the resolution of this set of points rules the quality of the approximation. This general approach leads to the very general problem that these algorithms face: the dimensionality curse. Indeed, when the dimension of the state space increases, the global number of points increases exponentially to keep the same resolution (and the same accuracy of approximation). Hence, the limits of computational capacities are reached rapidly (for space of dimension more than 6 or 7).

4.1 Discretising in space and time to approximate viability kernels.

The first algorithm was developed by Saint-Pierre [8]. It provides an approximation of the viability kernel $Viab_{\varphi}(K)$ of when φ satisfies some conditions[1]. The principle of the algorithm is to define a new dynamical system which is discrete in time and space. To do this the principle is to consider a grid of points $K_h = (x_k)_k^h$ of resolution h covering K such that:

$$\forall x \in K, \exists x_i \in K_h(\text{ such that } \|x - x_i\| \leq h, \tag{5}$$

This discrete dynamical system is defined by function $G^h : (x_k)_k^h \times U \rightarrow (x_k)_k^h$:

$$\left\{ \ G^h(x_i, u) = x_j = argmin_{(x_k)_k^h} \|x_k - (x_i + \varphi(x_i, u(t))dt)\| \right. \tag{6}$$

By considering the sequence $(K_h^n)_n$, with $K_h^0 = K_h$:

$$K_h^{n+1} = \{x_k \in K_h^n \text{ such that } \exists u \in U(x_k) \text{ and } G_h(x_k, u) \in K_h^n\}, \tag{7}$$

we obtain $Viab_{G_h}(K) = \bigcap_{n=0}^{+\infty} K_h^n$, and there exists one integer p such that $Viab_{G_h}(K) = K_h^p$.

Saint-Pierre shows that, when φ is μ-Lipschitz, the discrete viability kernel tends to the viability kernel of the initial system when the resolution of the grid h tends to 0.

The algorithm is fast, but at each iteration, the current approximated viability kernel K_h^n is defined as a set of points, which is not very convenient to manipulate. In addition, the numerical scheme is diffusive, which gives overestimations of actual kernels.

Computing resilience values comes down to an approximation the viability kernel of an auxiliary system.

First, we introduce the cost function $\gamma(x, u)$ that associates a cost to a point x of the state space when applying action u, and function $C_K(x_0)$ that associates to x the minimal cost on all trajectories starting at x_0:

$$C_K(x_0) = \inf_{u(.)} \int_0^{\infty} \gamma(x(t), u(t))dt. \tag{8}$$

[1]The associated set valued map F map is Marchaud: it is upper semicontinuous, with compact convex values and verifies $|F(x)| = \sup\{|y| \, | y \in F(x)\} \leq c(1 + |x|), \forall x \in \mathbb{R}^n$

Moreover, we suppose that the cost of being in the viability kernel is null, because by hypothesis, this is the set of states that is desired. On the contrary, we suppose that being outside the viability kernel has a given cost (because from the states outside the viability kernel, we are sure that no action policy prevents from violating the constraints).

$$\gamma(x,u) > 0 \text{ when } x \notin Viab(K) \tag{9}$$

$$\tag{10}$$

The resilience problem is thus to determine action policies driving back the system into the viability kernel, at minimum cost. This is typically a problem of optimal control. Yet, it can be shown ([5]) that viability theory can be used to solve it. One needs to modify the state space by adding an axis for the cumulated cost over a trajectory. This new state space includes couples (x, C), where x is in the initial state space, and C represents the cost associated with this point. The constraint set of the extended viability problem is $x \in Viab(K))$ or $C > 0$. The dynamics of the system are given by:

$$(x(t), c(t))' = (\varphi(x(t), u(t)), -\gamma(x(t), u(t))) \tag{11}$$

$$u(t) \in U(x(t)) \text{ with the constraint: } if(x \notin Viab(K)), c(t) > 0. \tag{12}$$

4.2 Discretising only in time and using machine learning techniques.

The main motive behind using other classification procedures in the viability algorithm was that if these classification procedures have good properties, maybe we could use less points of the grid to define them, with the same accuracy, and therefore progress in the fight against the dimensionality curse. We now introduce this approach, which we used in practice on the case studies. The algorithm is based on the viability algorithm and works roughly as follows: at each iteration, we use a classification procedure to define a continuous approximation of the discrete set K_h^n and we remove the points of the grid which "boldy" leave this approximation. The other main difference with the viability algorithm is that we discretise the map F in time (with map G) but not in space (map G_h). This algorithm is described in details in chapter 10 of the book.

Then, one can define different management strategies to keep the system in its desirable set of states. One strategy is called "'heavy"'. Its principle is to keep

the same choice of action while the system does not cross the boundary of the viability kernel at the next time step. If the system crosses this boundary at the next time step, then choose one action that keep the system inside the viability kernel (such an action always exists). Another strategy is called "'lazy"'. Its principle is similar to the heavy strategy except that by default there is no action applied on the system. This means that "'no action"' should be one option in the set of possible actions which is not always the case. The particularity of this action strategy is that it tends to go to the attractors of the dynamics (without control), if such attractors are present in the constraint set.

In practice, in the software tool that we developed in the project (deliverable 3.2), we use "'support vector machines"'as a particular classification procedure. Support Vector Machines (SVM) are a set of supervised methods used for classification or regression. We consider here only the binary classification task: given a set of data points with associated labels $+1$ or -1, the goal is to construct an linear hyperplane which allows one to decide which class a new data point will be in. SVM construct the best possible hyperplane as the one that provides the largest separation, or margin, between the two classes. However, in most cases, the linearly separable assumption doesn't hold, but the idea of a linear model can be kept, while obtaining non-linear hyperplane, by using the "kernel trick" [**?**] (this kernel is totally different from the viability kernel). The function of the SVM has the following form:

$$f(x) = \sum_{i=0}^{N} \alpha_i y_i k(x, x_i) + b = 0. \tag{13}$$

Where x_i represent the points of the grid, k is a function called a kernel and must satisfy very general properties, α_i and b are scalar values computed by the learning algorithm. The points x such that $f(x) \geq 0$ are in class $+1$, the others in class -1. The SVMs are generally parsimonious because many of the values α_i are null. For more details, one can refer to [**?**] or [**?**].

The algorithm defined for computing viability kernels with a machine learning procedure implies to compute the distance from a point the boundary of the current approximation of the set. In the case when this boundary is approximated by a SVM, this distance is difficult to compute. To simplify the computations, we use a proxy to evaluate this distance, using directly the function f. This allows us to use optimisation techniques in order to find the control values that keep the system in the current approximation of the viability kernel.

Moreover, during the project, we developed an algorithm which computes re-

silience values directly in the state space of the system, whereas in the usual approach, it is necessary to add an auxiliary dimension for the cumulated cost. In this case, the SVMs approximate successive surfaces with increasing cost for going back to the viability kernel.

Similarly to the viability kernel approximation, the SVM are a particularly relevant classification function to approximate restoration cost (and hence resilience values). It enables one to use standard optimization techniques to find an appropriate control u^*:

$$u^* = \arg \max_{u \in U(x)} f_n(x + \varphi^*(x, u)dc) \tag{14}$$

that lead to the "optimal" state x^*. We can also extend the procedure to several time steps optimization at each iteration.

4.3 Kaviar.

Kaviar is a software under a GPL V3 license written in Java. In function of the chosen dynamical system, the application gives a viability kernel, capture basin or resilience values approximation. Once the approximation is obtained, the user can use a adequate controller (heavy or optimal one). The software and an user guide are downloadable at `http://trac.clermont.cemagref.fr/projets/Kaviar/`. Kaviar works in 2 modes: a batch mode and a GUI one. Figure **??** shows a snapshot of the graphical interface and figure **??** an example of a viability kernel with an heavy trajectory, and the controller panel associated. For a quick start with the software, there exists a java executable file which allows a new user to make his first step with Kaviar. The source code is available in order to implement new models. On the user guide, one can find the prerequisites and the installation steps, how to run the program, a description of the already implemented models, how to run the program in a batch mode and how to implement a new model. In the handbook, we give concrete examples of viability kernel, resilience values and associated action policy computations. In particular, we give some clues to define the values of the main parameters.

5 Methodological lessons from the case studies.

We used the approach more particularly in three case studies: languages, savanna and bacteria. In the case study about the resilience of languages, we could use

Kaviar without difficulties and it gave rather accurate approximations of the viability kernels. Indeed, the problem is in 2 dimensions, and the constraint set is easy to determine.

In the case study about savanna, the situation was similar when considering only the mean field model. The problem is in 2 dimensions, and the different options for the constraint set are easy to determine. It was not the same when we applied the approach to the pair approximation model. Then, we had 2 additional variables, which led to a problem in 4 dimensions. But the main problem was that the bounds of variation of these new variables depended on the value of another one. Hence we had a complex shape for the state space, which was not easy to express directly in Kaviar. We finally solved the problem by choosing other variables which were equivalent and had the pleasant property to have fixed bounds. With this new choice of variables, we could apply Kaviar directly. This type of practical problem is general, and the lesson is that it can be very efficient to look for changing the variables into ones with fixed bounds.

In the case study about bacterial biofilms, we had a state space given by the moment approximation which was in 10000 dimensions (2D correlations of 100×100). This is of course impossible to run Kaviar with in such a state space. In this case, we changed again the variables by considering the integral of the values above one and the integral of the values below one for each pattern. It turned out that these descriptors were sufficient to characterize uniquely each pattern.

6 Geometric robustness.

Viability-based resilience is supported firstly by the computation of the viability kernel of the system constraint set and then by the computation of the basin of attraction of the viability kernel. Standard control policies (inertia or slow control) are usually proposed to maintain the dynamical system in this appropriate subset of state space. Unfortunately, following these policies often leads the system to evolve part of the time on the boundary of the viability kernel or capture basin. This is an unsafe place to stay in because of possible unaccounted uncertainties but also because on the boundary the control is not robust in the classical sense: only very few control values keep the system viable. In chapter eleven of the book we present methods to define and take into account the robustness of a state and a trajectory.

6.1 Geometric definition of the robustness of a state and algorithms.

Robustness of a state is classically defined by its sensitivity as in classification system, that is the distance to the viability kernel or capture basin boundary. This geometric definition allows to define sensitive disturbance as the biggest perturbation that keeps the state of the system in the viability kernel or capture basin.

With this definition of the robustness of a state inside the viability kernel or the capture basin, it is possible to propose several definitions for the robustness of a trajectory. For example, the most risk-adverse indicator is the $\min$ distance to the boundary over the trajectory. The average distance can also be used as an average value to define the robustness of a trajectory. Other definitions can be proposed with a discounting rate for future distance, to take into account the fact that it is possible to modify a trajectory in the future, therefore the distance of future states to the boundary is less critical than the present one. All these definitions are illustrated on a simple example, the lake eutrophication problem, which was originally used to introduce the concept of viability-based resilience [5].

An optimal algorithm coming from mathematical morphology was adapted to compute an approximation of the distance (for several metrics) to the viability kernel or capture basin boundary, and the projection in the case of the Euclidean distance. This distance algorithm computes the exact distance on the discretized boundary.

The projection (possibly projections) on the kernel boundary defines the sensitive disturbance, it shows the direction where uncertainty or error in state determination is the most risky. The important issue of the choice of the metric is also discussed.

6.2 Geometric robustness-based strategy: Application to the language competition model.

Then, the concept of robustness inside the viability kernel or capture basin is used to define control strategies that lead to more robust trajectories.

These strategies were tested on the language competition application, which is one of the case studies of the project. We used here a tree-dimensional version of the language competition model, including bilingualism and the relative attractiveness of one language versus the other.

With this model, experiments show that classical control strategies, such as heavy or slow trajectories, and the classical St Pierre algorithm, lead the system

to evolve more than half of the time on the viability kernel boundary. This means that the robustness value of the trajectories is either null or very low (depending on the definition of the robustness). With the approach based on support vector machines, the control can be more robust, because it is possible to define relevant controls a few time steps before reaching the boundary. However, the computation of the distance to the boundary with the SVMs is not direct, and we use the more or less satisfactory proxy of the SVM function to access it. Hence the robustness is not completely managed.

Geometric-based control strategies consist in taking into account the robustness of the current state to choose the control. With these strategies, when the robustness decreases to a given threshold, the classical heavy or slow strategy is no longer followed. Instead, the control that would be applied at the projection point corresponding to the current point is applied. Experiments show that these geometric-based control strategies are much more robust than the classical ones.

References

[1] J.P. Aubin. *Viability Theory*. Birkhauser, Basel, 1991.

[2] V. Grimm and C. Wissel. Babel, or the ecological stability discussions: an inventory and analysis of terminology and a guide for avoiding confusion. *Oecologia*, 109:323–334, 1997.

[3] L. Gunderson and C.S. Holling. *Panarchy : understanding transformations in human and natural systems*. Island Press, Washington (DC), 2002.

[4] C.S. Holling. Resilience and stability of ecological systems. *Annual Review of Ecology and Systematics*, 4:1–24, 1973.

[5] S. Martin. 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), 2004.

[6] Y. G. Matsinos and A. Y. Troumbis. Modeling competition, dispersal and effects of disturbance in the dynamics of a grassland community using a cellular automaton. *Ecological Modelling*, 149:71–83, 2002.

[7] M. Ortiz and M. Wolff. Dynamical simulation of mass-balance trophic models for benthic communities of north-central chile: assessment of resilience time

under alternative management scenarios. *Ecological Modelling*, 148:277–291, 2002.

[8] P. Saint-Pierre. Approximation of viability kernel. *Applied Mathematics and Optimization*, 29:187–209, 1994.