Kaviar

a software to approximate viability kernels, capture basins and resilience values

Version 1.1

User Guide

Laetitia Chapel

October 13, 2009

1 Introduction

We developed software to approximate viability kernels and compute resilience values using Support Vector Machines (SVMs). In this version:

- you can use either the algorithm with a regular grid (see [Deffuant et al., 2007]) or the active learning algorithm (see [Chapel and Deffuant, 2007]);
- the capture basins and resilience values are computed in dimension d;
- you can define heavy (for kernel approximation) or optimal (for capture basin approximation) controllers;
- the results can be visualized in 2d

The software is written in the Java programming language. The software has two modes: a GUI mode and a batch mode.

2 Glossary

- **Viability kernel (Viab(K)):** set of all the points x for which there exists at least one control function $t \rightarrow u(t)$ such that the whole evolution starting from x always remains in the viability constraint set K.
- **Capture basin (Capt(K,C)):** set of states for which there exists at least one trajectory that reaches a target C in finite time, without leaving K.
- **Resilience values:** computed as the inverse of restauration cost. Inside the viability kernel; the cost is null, if there exists a trajectory that allows the system to come back to K, the cost is finite and its value is a function of the time needed to restore the system; if there is no control function that allows the restauration of the system, the cost is infinite.
- **Heavy controller:** keep the control constant as long as the system stays inside the viability kernel and change it only when the system reaches the boundary on the kernel, by choosing the first control that keeps the system inside $Viab(K)$.
- **SVMs:** classification procedures that define a separating hypersurface between examples x_i, associated with their label y_i. SVM training provides function:

  $$f(x) = \sum_{i=1}^{n} \alpha_i y_i k(x, x_i) + b.$$

  The points x_i for which $\alpha_i > 0$ are called support vectors. The sign of the function $f(x)$ gives the label of the point x. $k(x, x_i)$ is called kernel, for example the gaussian kernel has the following expression:

  $$k(x, x_i) = \exp\left(-\gamma \parallel x - x_i \parallel^2\right), \quad (1)$$

  where $\gamma > 0$ is a parameter.

Contents

3 Installation

3.1 Prerequisites

To run the java program, you need to install first:

- Java virtual machine (Sun's JRE environnement 5 or later compulsory), downloadable at
  `http://java.sun.com/javase/downloads/index.jsp`

The program can be run on Windows, GNU/Linux and Mac OS. The software version 1.1 is provided as a .jar file (**Kaviar-1.1.jar**) that allows to test the models already implemented in the version 1.1 and to implement additional model, and as a development kit archive (**Kaviar-Development-Kit-1.1.zip**) that provides a ready to use NetBeans developpement environment to add a new model.

3.2 Installation

The file **Kaviar-1.1.jar** is a java executable file. It runs the software with the Graphical User Interface.

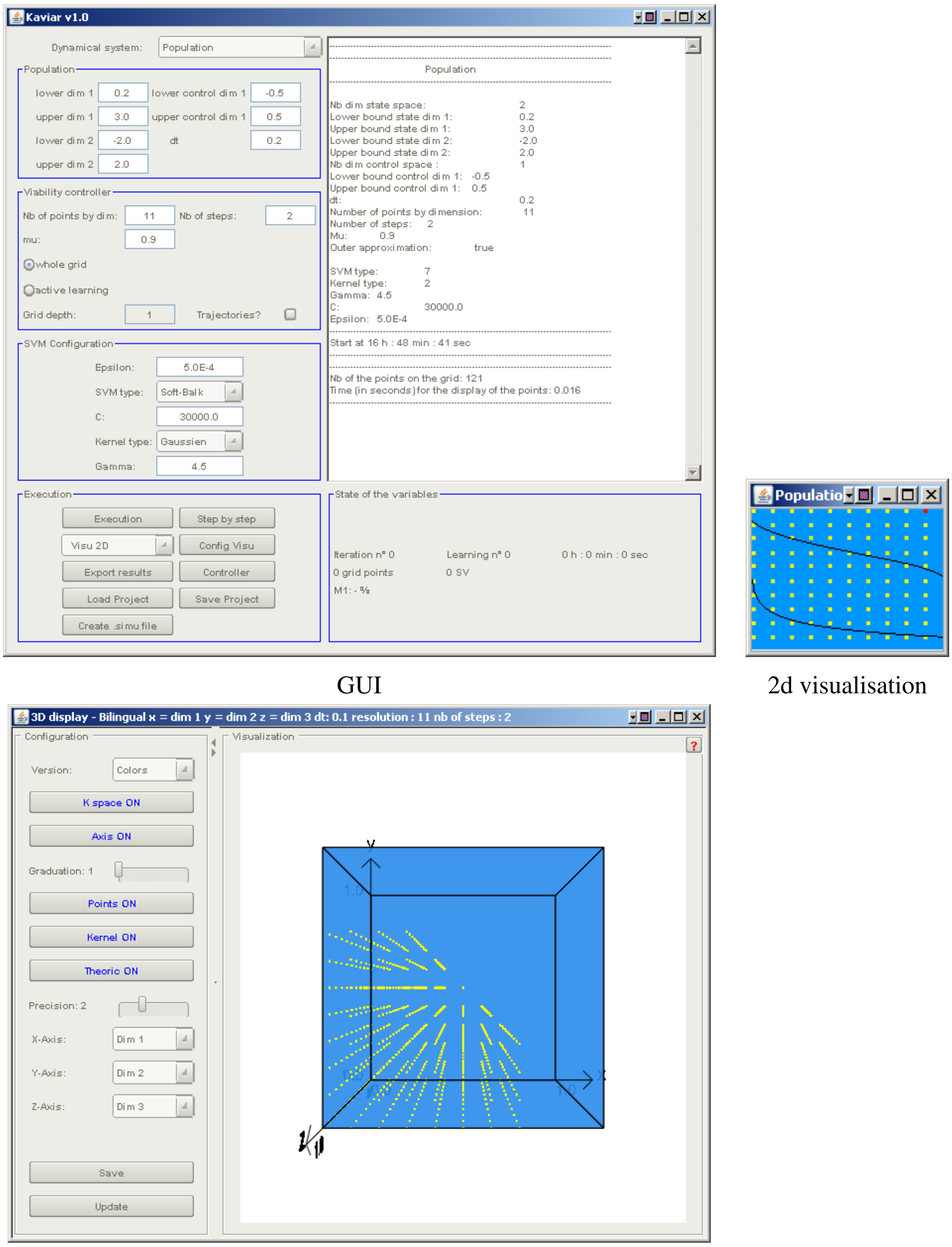
For users who want to implement their own model, you can either use the ready to use NetBeans project provided by the development kit archive **Kaviar-1.1.zip**, or use the file **Kaviar-1.1.jar** as a java dependency.

3.3 Run the program

Double click on the **Kaviar-1.1.jar** file. You should have the GUI represented in figure 1.

For users who wish to implement their own models, the easier way is to use the developpement archive that provides a NetBeans project. You have to download and install Netbeans at `http://www.netbeans.org/downloads/`, then unzip the development kit archive **Kaviar-1.1.zip** and open this directory as a project from NetBeans.

You can also develop your model with another tool (such as Eclipse IDE) by adding **Kaviar-1.1.jar** to your class-path.



GUI

2d visualisation

3d visualisation

Figure 1: Software in the GUI mode

4 Description of the models

Some models are already implemented in the software. This part gives a description of the dynamics of them, and their viability constraint set.

4.1 Population system [Aubin, 2002]

We consider a simple dynamical system of population growth on a limited space. The state $(x(t), y(t))$ of the system represents the size of a population $x(t)$, which grows or diminishes with the evolution rate $y(t) \in [d; e]$. The size of the population must remain in an interval $K = [a, b]$, with $a > 0$. The inertia bound c limits the derivative of the evolution rate at each time step. The system in discrete time defined by a time interval dt can be written as follows:

$$x(t + dt) = x(t) + x(t)y(t)dt \quad (2)$$
$$y(t + dt) = y(t) + u(t)dt \quad (3)$$

with $-c \leq u(t) \leq +c$. The viability constraint set is the set $K = [a; b] \times [d; e]$. For example, we can use the following parameters: $a = 0.2$, $b = 3$, $c = 0.5$, $d = -2$, $e = 2$.

4.2 Consumption system [Aubin, 1991]

The problem is a consumption problem, defined in 2 dimensions $x(t)$ and $y(t)$. Variable $x(t)$ represents the consumption of a raw material and $y(t)$ its price. The price evolution between two time steps is bounded. The viability constraint set is the set $K = [a; b] \times [d; e]$. The dynamics represent the consumption of the raw material, limited by prices:

$$x(t + dt) = x(t) + (x(t) - y(t))dt \quad (4)$$
$$y(t + dt) = y(t) + u(t)dt, \quad (5)$$

with $-c \leq u(t) \leq +c$. For example, we can use the following parameters: $a = 0$, $b = 2$, $c = 0.5$, $d = 0$, $e = 3$.

4.3 Lake system [Carpenter et al., 1999]

The system under consideration encompasses a lake and the farming activities in its watershed. The model combines an ecosystem model of phosphorus dynamics and a controlled model for phosphorus input dynamics. Dynamic of phosphorus in water follows the model:

$$\frac{dP(t)}{dt} = -bP(t) + L(t) + r\frac{P(t)^q}{P(t)^q + m^q}, \quad (6)$$

where P is the mass of phosphorus in the lake water, b is the rate of phosphorus elimination at each time step, m is the P mass in the water for which recycling is half of the maximum rate r. Parameter q sets the steepness of the recycling versus P curve when $P \approx m$. $L(t)$ represents the inputs of phosphorus that come from human activities. We suppose that the lake manager can act directly on the time variation through control u, with u bounded because modifications take time:

$$\frac{dL(t)}{dt} = u, \text{ with } u \in [L_{min}, L_{max}]. \tag{7}$$

We also assume that an oligotrophic lake becomes eutrophic when the amount of phosphorus in the water increases over some fixed threshold P_{max}. We suppose that farmers' benefit depends linearly on the inputs of phosphorus. Consequently, profitability is reached when the value of phosphorus inputs is higher than a given threshold L_{min} and lower than the maximal legal value L_{max}.
The system is thus 2-dimensional:

$$\begin{cases} \frac{dP(t)}{dt} = -bP(t) + L(t) + r\frac{P(t)^q}{P(t)^q+m^q}. \\ \frac{dL(t)}{dt} = u, \end{cases} \tag{8}$$

and $K = [L_{min}; L_{max}] \times [0; P_{max}]$. For example, we can use the following parameters: $L_{min} = 0.1$, $L_{max} = 1$, $P_{max} = 0.5$, $c = 0.1$, $q = 8$, $r = 1$, $m = 1$, $b = 0.8$.

4.4 LakeResilience system [Martin, 2004]

We are interested in determining if the lake system can come back to the viability kernel. We attribute a cost for the system for each time step such that $P > P_{max}$ or $L < L_{min}$. This can be expressed as a 3 dimensional viability problem, where the state is (L, P, C), with C representing the cost. The equations ruling the system in discrete time are:

$$P(t + dt) = P(t) + dt.P'(t) \tag{9}$$

$$-c.dt \leq L(t + dt) - L(t) \leq c.dt \tag{10}$$

$$C(t + dt) = \begin{cases} C(t) \text{ if } P < P_{max} \text{ and } L > L_{min} \\ C(t) - \lambda(x)dt \text{ otherwise,} \end{cases} \tag{11}$$

with $\lambda > 0$. We use a cost function consisting of two weighted terms, the first term, which corresponds to the ecological cost, is the time spent in an eutrophic state, the second one, which is an economic cost, measures the duration of the period of negative profits weighted by the norm of these negative profits. The function that associates x with the minimal cost over all trajectories starting at x is then defined by:

$$\lambda(x) = \min_{x(.)}(c_1 \int \chi_{P \geq P_{max}} x(\tau)d\tau + c_2 \int (L_{min} - L)\chi_{L \leq L_{min}} x(\tau)d\tau) \tag{12}$$

with $\chi_{P \geq P_{max}}(x) = 1$ if $P \geq P_{max}$, $\chi_{L \leq L_{min}}(x) = 1$ if $L \leq L_{min}$ and 0 otherwise.
The state space is 3 dimensional $H = [0, L_{max}] \times [0, P_M] \times [0, C_M]$, with $P_M > P_{max}$.
The cost function $\lambda(x)$ equals 0 if $x \in K = [L_{min}, L_{max}] \times [0, P_{max}]$.

4.5 LakeMud system [Janssen and Carpenter, 1999]

The phosphorus cycle of the lake is described by a model that distinguishes the fast and slow dynamics. The model includes the mass of phosphorus in the lake water (P), that

loads from farming activities and process by outflow and sedimentation. We suppose that farmers use phosphorus that goes directly into the lake and increases the mass of phosphorus in the water. The phosphorus recycling rate depends on the sedimented phosphorus, which is called mud (M). Hence, the mass of phosphorus in the lake (fast dynamics) is driven by an other state variable, the mass of phosphorus in the mud (slow dynamics). One part of the mass of phosphorus contained into the mud converts to burial each year.
The dynamic equation representing the phosphorus in sediments and water (both in $g.m^{-2}$) is given by:

$$\begin{aligned} \frac{dP}{dt} &= -(s+h)P + L + rMf(P) \\ \frac{dM}{dt} &= -kM + sP - rMf(P), \end{aligned} \tag{13}$$

where L is the input of phosphorus (in $g.m^{-2}.y^{-1}$) that comes from human activities and

$$f(P) = \frac{P^q}{P^q + m^q}. \tag{14}$$

We suppose that the lake manager can act directly on the dynamics of phosphorus inputs, by modifying their time variation. Like in [Martin, 2004], we suppose that the derivative of L depends on the control u chosen by the manager, with u bounded because modifications take time:

$$\frac{dL}{dt} = u \text{ with } u \in [-VL_{max}, VL_{max}]. \tag{15}$$

We assume that the lake ecosystem objective is reached when the positive variable P satisfies:

$$P \in [0, P_{max}]. \tag{16}$$

We also suppose that farming activities are profitable and legal when:

$$L \in [L_{min}, L_{max}]. \tag{17}$$

The system is thus 3-dimensional (L, P, M) and $K = [L_{min}; L_{max}] \times [0; P_{max}] \times [0; M_{max}]$. For example, we can use the following parameters: $L_{min} = 0.1$, $L_{max} = 2$, $P_{max} = 2$, $s = 0.7$, $h = 0.15$, $r = 0.019$, $k = 0.001$, $m = 2.4$, $q = 8$.

4.6 LakeMudResilience system [Chapel et al., 2007]

We are interested in determining if the lake system that includes mud dynamics can come back to the viability kernel. The parameters of this model are the same that those of the lakeResilience model (there is no cost for having a large amount of phosphorus in the mud).

4.7 Language1d system [Abrams and Strogatz, 2003]

We consider the Abrams-Strogatz model of language competition. Two languages A and B are in competition and it is supposed that individuals speak only one language.

σ_A (resp σ_B, with $\sigma_B = 1 - \sigma_A$) is the density of speakers of language A (resp B). The language dynamics is given by:

$$\sigma'_A = (1 - \sigma_A)\sigma_A(\sigma_A^{a-1}s - (1 - \sigma_A)^{a-1}(1 - s)), \tag{18}$$

where s is a parameter which is in $[0, 1]$. It measures the social status, the prestige of language A or the politics in favor of language A. The prestige of language B is $1 - s$. For instance, if $s = 0$, the prestige of language A is null and the prestige of language B is maximal and if $s = 1$, this is the opposite. a is a parameter that represents the volatility of language A.
We want to examine the conditions in which the coexistence of speakers of A and B is possible. More precisely, we would like to keep the minority language above the density b. We suppose that the prestige s is a control parameter $s = u$ and that it can have 2 values: s_{min} and s_{max}. Therefore, we get the following system:

$$\begin{cases} \sigma'_A = (1 - \sigma_A)\sigma_A(\sigma_A^{a-1}u - (1 - \sigma_A)^{a-1}(1 - u)) \\ u = s_{min} \text{ or } u = s_{max} \\ b \leq \sigma_A \leq 1 - b \\ 0 \leq s \leq 1. \end{cases} \tag{19}$$

and $K = [b; 1 - b]$. For example, we can use the following parameters: $a = 1.31$, $b = 0.2$.

4.8 Language system [Abrams and Strogatz, 2003]

We consider the Abrams-Strogatz model of language competition but we suppose here that the prestige s can change and we include it as a state variable. The state space is therefore bidimensionnal (σ_A, s). Moreover, we suppose that the change of prestige during a given time step is limited: $-c \leq s' \leq c$. Therefore, we get the following system:

$$\begin{cases} \sigma'_A = (1 - \sigma_A)\sigma_A(\sigma_A^{a-1}s - (1 - \sigma_A)^{a-1}(1 - s)) \\ -c \leq s' \leq c \\ b \leq \sigma_A \leq 1 - b \\ 0 \leq s \leq 1. \end{cases} \tag{20}$$

and $K = [b; 1-b] \times [0; 1]$. For example, we can use the following parameters: $a = 1.31$, $b = 0.2$, $c = 0.1$.

4.9 LanguageResilience system [Bernard et al., 2008]

We are interested in determining if the Abrams-Strogatz language system (without bilinguals) can come back to the viability kernel. We attribute a cost for the system for each time step such that $\sigma_A < b$ or $\sigma_A > 1 - b$. This can be expressed as a 3 dimensional viability problem, where the state is (σ_A, s, C), with C representing the cost. The equations ruling the system in discrete time are:

$$\sigma_A(t + dt) = \sigma_A(t) + dt.\sigma'_A(t) \tag{21}$$

$$-c.dt \leq s(t + dt) - s(t) \leq c.dt \tag{22}$$

$$C(t + dt) = \begin{cases} C(t) \text{ if } \sigma_A < b \text{ and } \sigma_B < b \\ C(t) - \lambda dt \text{ otherwise,} \end{cases} \tag{23}$$

with $\lambda > 0$. We also specify a maximal cost C_M. For $\lambda = 1$, the cost represents the time the system is outside $K = [b; 1 - b] \times [0; 1]$. The state space is 3 dimensional $H = [0, 1] \times [0, 1] \times [0, C_M]$.
In this version of the algorithm, it is possible to work only in 2 dimensions $H = [0, 1] \times [0, 1]$. In this case, the resilience values are represented by level lines.

4.10 Bilingual system [Minett and Wang, IP]

In this model, we consider the existence of bilingual individuals. We note: σ_A is the density of speakers of language A, σ_B is the density of speakers of language B, σ_{AB} is the density of speakers of language A and B. The language dynamics is given by this model proposed by Minett-Wang (eliminating variable σ_{AB}):

$$\sigma'_A = (1 - \sigma_A - \sigma_B)\sigma_A^a s - \sigma_A \sigma_B^a (1 - s) \tag{24}$$

$$\sigma'_B = (1 - \sigma_A - \sigma_B)\sigma_B^a (1 - s) - \sigma_B \sigma_A^a s. \tag{25}$$

As in the previous language model, we suppose we want to keep both languages above a threshold density b, and that the absolute value of the prestige derivative cannot be larger than c. The viability constraint set is thus $\sigma_A + \sigma_{AB} \geq b \equiv sigma_B < b$ and $\sigma_B + \sigma_{AB} \geq b \equiv sigma_A < b$ So we have the following system:

$$\begin{cases} \sigma'_A = (1 - \sigma_A - \sigma_B)\sigma_A^a s - \sigma_A \sigma_B^a (1 - s) \\ \sigma'_B = (1 - \sigma_A - \sigma_B)\sigma_B^a (1 - s) - \sigma_B \sigma_A^a s \\ -c \leq s' \leq c \\ 0 \leq \sigma_A \leq b \\ 0 \leq \sigma_B \leq b \\ \sigma_A + \sigma_B \geq 1 \\ 0 \leq s \leq 1. \end{cases} \tag{26}$$

The state space is now 3 dimensional (σ_A, σ_B, s), and $K = [0; b] \times [0; b] \times [0; 1]$.

4.11 BilingualResilience system [Bernard et al., 2008]

We are interested in determining if the Minett-Wang language system (without bilinguals) can come back to the viability kernel. We attribute a cost for the system for each time step such that $\sigma_A > b$ or $\sigma_B > b$. This can be expressed as a 4 dimensional viability problem, where the state is $(\sigma_A, \sigma_B s, C)$, with C representing the cost. The equations ruling the system in discrete time are:

$$\sigma_A(t + dt) = \sigma_A(t) + dt.\sigma'_A(t) \tag{27}$$

$$-c.dt \leq s(t + dt) - s(t) \leq c.dt \tag{28}$$

$$C(t + dt) = \begin{cases} C(t) \text{ if } \sigma_A < b \text{ and } \sigma_B < b \\ C(t) - \lambda dt \text{ otherwise,} \end{cases} \tag{29}$$

with $\lambda > 0$. We also specify a maximal cost C_M. For $\lambda = 1$, the cost represents the time the system is outside K. The state space is 4 dimensional $H = [0, 1] \times [0, 1] \times [0, 1] \times [0, C_M]$.
In this version of the algorithm, it is possible to work only in 3 dimensions $H = [0, 1] \times [0, 1] \times [0, 1]$. In this case, the resilience values are represented by level lines.

4.12 Madagascar system [Carpenter et al., 1999]

The system represents the evolution of surface area converted s

$$s'(t) = -as(t) + u(t) \tag{30}$$

and the capital c of the population.

$$c'(t) = -ac(t) + \left(\min\left(1, \frac{s}{\gamma}\right) - \alpha\right) - \beta u(t) + \tau c(t) \tag{31}$$

. The control is the positive variation of the surface area:$u(t) \in [0; u_{max}]]$. Parameter a is the growing rate of the population, $\gamma = 1$ the surface area already converted, $\alpha \in [0, 1]$ is the production cost, $\beta \in [0; 8]$ is the converted cost and τ is the interest rate.
The viability constraint set is:

$$K = \{(s, c) \text{ such that } c \geq 0\}. \tag{32}$$

4.13 MadagascarResilience system [Martin, 2004]

We are interested in determining if the madagascar system can come back to the viability kernel. We attribute a cost for the system for each time step such that $c < C_{min}$. By default, the cost function is the time spent with a negative capital.

4.14 Biofilm2D system [Mathias et al., 2008]

A chemostat is here considered with a dilution rate equal to D. We are considering the substrate concentration s and the biomass concentration x. The chemostat is alimented with an input concentration of substrate s_0. The chemostat behavior is:

$$\frac{ds}{dt} = D(s_0 - s) - \mu(s)x \tag{33}$$

$$\frac{dx}{dt} = (\mu(s) - D)x. \tag{34}$$

The growth function $\mu(s)$ is so-called "generalized Monod equation". The Michaelis-Menten form is used for the kinetic:

$$\mu(s) = \mu_{max}\frac{s}{s + k_s} \tag{35}$$

k_s represents the Monod constant and μ_{max}, the mean maximum growth rate. This model is often used to approximate the chemostat behavior and constitutes the first model.
The control is $D \in [0; 2.10^{-5}]$ and $K = [0.2; 1] \times [1; 6]$.

4.15 Biofilm3D system [Mathias et al., 2008]

In this case, we want to have a finer model which integrates the detachment of bacterial biofilms due to the hydrodynamics stresses:

$$\frac{ds}{dt} = D(s_0 - s) - \mu(s)s \tag{36}$$

$$\frac{dx}{dt} = -f_x^{IBM}(x, s, p) + c_x \mu(s)x \tag{37}$$

$$\frac{dp}{dt} = -f_p^{IBM}(x, s, p) + c_p \mu(s)x, \tag{38}$$

$$\tag{39}$$

with p concentration of polymer, $c_x \in [0; 1]$, $c_p \in [0, 1]$ (with $c_p + c_x = 1$), and:

$$f_x^{IBM}(x, s, p) = k_x x \tag{40}$$

$$f_p^{IBM}(x, s, p) = k_p p \tag{41}$$

$$\tag{42}$$

We put $K = [0.2; 1] \times [1; 6] \times [1; 6]$. The control variable is $k_x = k_p \in [0; 1]$.

4.16 Savana2D system [Calabrese and Deffuant, 2008]

The model includes dynamics of birth, competition, death. The birth is local, because the offspring are sent around a tree occupied tree. The competition is also local: the probability to get a new tree depends on the occupation around. It is supposed that the radius of locality for birth is higher than the radius for competition. There is a natural death, and a death due to fire. Moreover, the fire depends on grazing, because grazing reduces the quantity of grass, which is the fuel for fire.
The equation ruling the evolution of the tree density is the following:

$$\frac{d\rho_1}{dt} = b(1 - \delta\rho_1)_+ \left(1 - \frac{\gamma(1-g)(1-\rho_1)}{\sigma + (1-g)(1-\rho_1)}\right)(\rho_1 - \rho_1^2) - \alpha\rho_1. \tag{43}$$

The control variable is $g \in [0; 1]$, the proportion of biomass consumed by grazers. The '+' subscript means that if the expression between parentheses is negative, then it is put to 0. Moreover, we suppose that the change g' of grazing cannot be too fast:

$$|g'| \leq g'_{max} \tag{44}$$

The density of trees must remain inside some bounds, which characterises savanna (if beyond the bounds, it becomes forest, below, it becomes prairie):

$$\rho_{1min} \leq \rho_1 \leq \rho_{1max}. \tag{45}$$

We consider a 2-dimensional state space (ρ_1, g), and we choose a control u which is the change of grazing. The transition function in discrete time is:

$$\rho_1(t + dt) = \rho_1(t) + \varphi(\rho_1(t), g(t))dt \tag{46}$$

$$g(t + dt) = g(t) + u(t)dt \tag{47}$$

With the constraints $|u(t)| \leq g'_{max}$ and $\rho_{1min} \leq \rho_1(t) \leq \rho_{1max}$.

4.17 Savana2DResilience system [Calabrese and Deffuant, 2008]

To study the resilience in this framework, we suppose that the Savanna can be submitted to some perturbations, which can change ρ_1 and g strongly. The problem is then to determine if it is possible to go back to the viability kernel, by modifying grazing.
We suppose that the perturbation can drive the system to any point $(\rho_1, g) \in [0,1] \times [0,1]$. We observe that, for $b = 8$ and $d = 0.7$, the system is not resilient for perturbations leading to $\rho_1 > 0.4$. Note here that the model is not well adapted to this problem, because in principle, the states $\rho_1 = 0$ and $\rho_1 = 1$ should be irreversible. If the dynamics were taking these irreversibilities into account, the resilient states would be very different. In particular, it would not be possible to restore the savanna states from a density $\rho_1 = 1$.

4.18 EndoNet system [Chavalarias and Chapel, 2008]

Social dilemma are situations where there is a conflict between individual interests and collective interest. The set S of strategies is composed of the three types: altruist (in proportion γ), reciprocator (in proportion δ) and selfish agents. These proportion evolve in the simplex $\Delta = \{(\gamma, \delta) \mid \gamma + \delta \leq 1\}$.

Given $(e, p, \theta) \in]0,1[\times]0,0.5[\times]0,1[$, the replicators dynamics is determined by the vector field V in the simplex Δ which is given by its two components:

$$V_\delta = \frac{\delta \pi_r}{\gamma \pi_a - \delta \pi_r - (1 - \delta - \gamma)\pi_e} - \delta \mid_{e,p} \tag{48}$$

$$V_\gamma = \frac{\gamma \pi_a}{\gamma \pi_a - \delta \pi_r - (1 - \delta - \gamma)\pi_e} - \gamma \mid_{e,p} \tag{49}$$

The control parameter of this system is p. Whereas e can be thought more as a structural parameter, it is reasonable to assume that institutions that regulate the activity described by this game and can act on p by helping people to assume the risk of cooperation or by punishing defectors. However, the extent in which p can be modified depends on the institution wealth and environmental constraints. From an initial p describing the strength of the dilemma without any action of the institution, this latter will be able to bring p to a value $p + \triangle p$ with the constraint $-\triangle p_{min} < \triangle p < \triangle p_{max}$.
Moreover, we will assume that the aim of the institution is to keep the level of cooperation above a given level $p_c > p0_c$ which defines the viability domain K of the system. The level of cooperation can be computed for each state of the system (γ, δ) from the equation describing the interaction and is a fractional function of δ, γ and θ. However, for sake of clarity, we will choose simpler conditions for the definition of K : $K = (\gamma, \delta) | \delta > 0, \gamma_{min} \leq \gamma, b \leq \gamma + \delta \leq 1$ with $b > 0$

4.19 ThinTarget system

In the system in 2 dimensions (x, y), we consider the problem of reaching a thin target $C = [0, 0]$. The system must stay in the space $K = [-1; 1] \times [-1; 1]$. The dynamics is defined in discrete time:

$$\left\{ \begin{array}{l} x(t+dt) = x(t) + y(t)dt \\ y(t+dt) = y(t) + u(t)dt, \end{array} \right. \tag{50}$$

with the control $u(t) \in [-1; 1]$.
We approach the capture basin in finite time T of the system (for example, try $T = 1$).

4.20 Zermelo system [Cardaliaguet et al., 1997]

The problem is derived from the famous Zermelo problem. The aim is to drive a boat in a river, such that it can reach an island as quick as possible, without leaving the constraint set. The state (x, y) represents the position of the boat in the river, where the current decreases when it approaches the boundary of the river. The constraint set K represents the river and C the island to reach. At each time step, the captain of the boat can control its acceleration u and its direction θ. The system in discrete time can be defined by:

$$\left\{ \begin{array}{l} x(t+dt) = x(t) + (1 - 0.1y(t)^2 + u\cos\theta)dt \\ y(t+dt) = y(t) + (u\sin\theta)dt. \end{array} \right. \tag{51}$$

For example, we can put $K = [-6; 2] \times [-2; 2]$ and $C = B(0; 0, 44)$, where B is the unit ball in R^2. Controls must remain in a given interval: $u \in [0; 0, 44]$ and $\theta \in [0; 2\pi]$. The boat must reach the island before time (try $T = 7$).

4.21 CarOnTheHill system [Moore and Atkeson, 1995]

The system is in 2 dimensions: the car position x and its velocity x'. The car can be controlled thanks to a continuous variable in one dimension: the thrust a. The aim is to keep the car in a given constraint set, while reaching the target as fast as possible, with a small velocity: $C \in [0, 5; 0, 7] \times [-0, 1; 0, 1]$. The car must remain in the constraint set $K = [-1; 1] \times [-2; 2]$. The thrust is limited $a \in [-4; 4]$. The evolution of the velocity x' is function of the position x :

$$x'' = \frac{a}{\sqrt{1 + (H'(x))^2}} - \frac{gH'(x)}{1 + H'(x)^2}, \tag{52}$$

with $g = 9.81$ and :

$$H'(x) = \left\{ \begin{array}{ll} x^2 + x & \text{if } x < 0 \\ x/\sqrt{1 + 5x^2} & \text{if } x \geq 0. \end{array} \right. \tag{53}$$

We consider the dynamical system in discrete time:

$$\left\{ \begin{array}{l} x(t+dt) = x(t) + x'(t)dt \\ x'(t+dt) = x'(t) + x''(t)dt. \end{array} \right. \tag{54}$$

5 User Guide for the java executable file

If you use the **Kaviar-1.1.jar** file, you can only use the software GUI mode.

The software is formed by two main windows: the console window and the display window. The **console window** allows the modification of the parameters, start and stop approximation, save and load the results, control the system. The **display window** allows one to visualize the approximation of the viability kernel or the resilience values, and trajectories if the user chooses to control the system.

The next subsection describes the different parts of the console window, and all the parameters that can be chosen by the user.

5.1 Console window

The console window contains 7 parts. When modifying one parameter, you have to go inside a text box of the second part (that defines the parameters of the model) and press enter, in order to take into account the modifications.

5.1.1 Choosing the system

The first part, at the top left, allows to choose the dynamical system.

Figure 2: First part of the console window

5.1.2 Definition of parameters of the models

The second part, called *Population* in the figure 1, allows the definition of the viability constraint set and the parameters of the model.

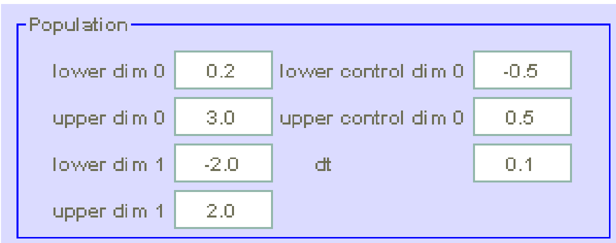


Figure 3: Second part of the console window

Table 1 describes the different parts of the window. There are as many text boxes as needed to choose the lower and upper bound of K and the control space. If the model has parameters, there are also special labels and text boxes to choose their values (not represented in the figure 3).

In case of capture basin approximation, there is an additional text box, to choose the maximal time of capture.

Element	Label	Description
text box	$lower\ dim\ i$	lower bound of K for the dimension i
text box	$upper\ dim\ i$	upper bound of K for the dimension i
text box	$lower\ control\ dim\ j$	lower bound of the j^{th} control dimension
text box	$upper\ control\ dim\ j$	upper bound of the j^{th} control dimension
label	$label\ of\ parameter\ k$	label of the k^{th} parameter
text box	$value\ of\ parameter\ k$	value of the k^{th} parameter
text box	dt	simulation time step

Table 1: Details of the second part of the console window

In case of resilience values computation, there are additional text boxes, to choose lower and upper bound of the whole space (H), the values of the cost parameters, the maximal cost, and the cost step dc (figure 4).

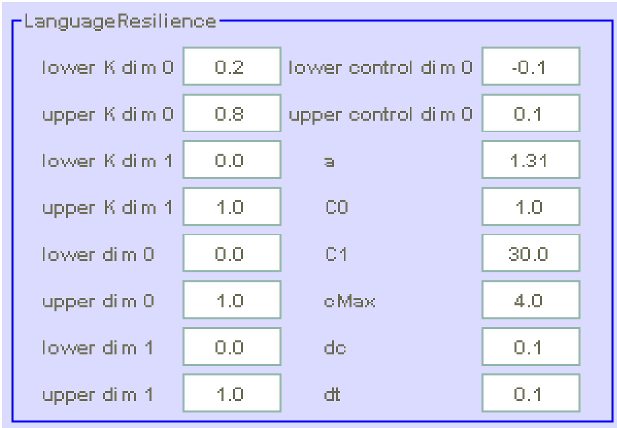


Figure 4: Second part of the console window for resilience computation

5.1.3 Viability controller

The third part, called the *Viability controller*, allows one to choose the parameters for the viability kernel approximation (and also capture basin approximation and resilience values computations). The two first line allow one to choose:

- the *number of points by dimension*: the greater the number of points, better will be the approximation. But increasing the number of points makes the execution longer;
- parameter for the gradient descent μ: a small value makes the simulation longer, a large value makes the simulation shorter but the optimization can be false;
- the *number of steps*: instead of looking if the system is viable at the next time step, we can look if it is viable at n time steps.

The 3 last lines of the left (whole grid, active learning and grid depth) allows the definition of the algorithm to use: *whole grid* means that you use the algorithm that works on the whole grid, *active learning* means that you use the active learning algorithm and *grid depth* allows one to choose the grid depth in case of active learning algorithm (the text box is not enable for whole grid).

On the right, you can choose if you want an outer or inner approximation (only for capture basin approximation and resilience values computation - not visible in case of viability kernel approximation).

This part also allows the visualization of the individual trajectories starting from each point of the grid, by checking *Trajectories?* (whole grid algorithm) or the points tested at each iteration (active learning algorithm).

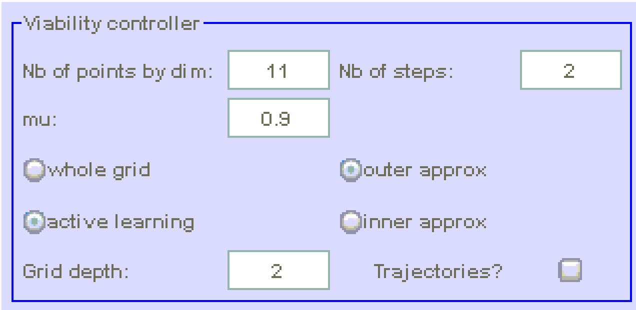


Figure 5: Third part of the console window

5.1.4 SVM configuration

The fourth part concerns the configuration of the SVM. To compute the SVM function, three algorithms are available:

- Simple, that implements SimpleSvm algorithm [Loosli et al., 2006]
- Balk or SoftBalk [Loosli et al., 2008]. These two algorithms allow the automatic bandwith autosetting, starting from an initial gamma value. Balk algorithm does not tolerate any misclassified points, whereas softBalk tolerates few errors (better to use softBalk).
- C SMO, from LibSvm Librairy (for more details about the library and its parameter settings, go to `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`)

You can choose some default parameters, detailed in figure 6. Parameter *epsilon* sets the tolerance of the termination criterion for the SVM computation algorithm. Its value must be small. In many cases, the best *SVM type* to choose is $Soft - Balk$. The value of C must be large, in order to fulfill the condition of the theorem (for more information about the theorem, see [Deffuant et al., 2007]). With the *gaussian kernel*, the SVM function can approximate all the classification functions . With the gaussian kernel, you have to choose the *gamma* (γ, see glossary) parameter, that allows the control of the shape of the SVM function: a small value of *gamma* gives smooth shapes while a large value allows the approximation of more irregular shapes.

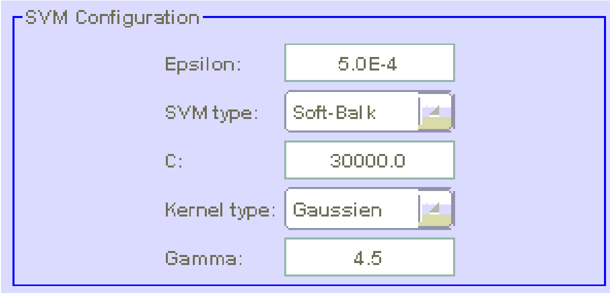


Figure 6: Fourth part of the console window

5.1.5 Execution and control

The fifth part concerns the execution.

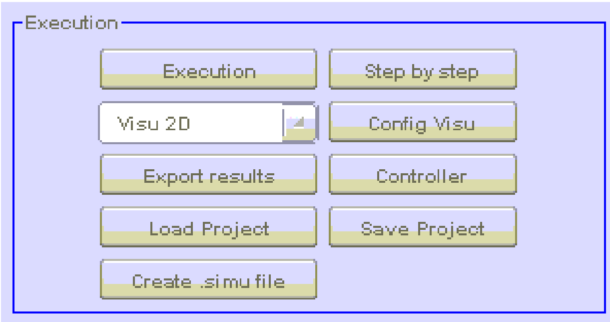


Figure 7: Fifth part of the console window

Table 2 describes the different buttons of the fifth part.

Button	**Description**
Execution	runs the program until the final approximation of the viability kernel is reached or the resilience values are computed
Step by step	shows all the iterations of the algorithm
2D view	choice of the visualization: no view or 2d
Config Visu	configures the display, by opening a configuration display window (see next paragraph (display configuration window))
Export results	export the results in a $.txt$ file (see next paragraph (Export file structure))
Controller	configures the controller, by opening a controller configuration window (see next paragraph (controller configuration window))
Load project	loads an approximation you have already made and saved
Save project	saves the results in 2 files: *.log* and *.svm* files.
Create .simu file	saves the settings in a .simu file (to use in the batch mode).

Table 2: Details of the second part of the console window

Display configuration window

Figure 8 presents the window of the configuration display. In this window, you can choose the dimension to display, and the value of the other dimensions for the display of the slices in 2D. If you modify the values or the dimensions to display, you have to press the *Update 2D view* to see the results on the display window. You can also activate or deactivate the view of the *theoretical curves* of the viability kernel (if they are available). You can choose whether or not to display the points of the grid. You can also only visualize the points that are support vectors by pressing the *Only SV* button. For the resilience values computation, you'll have another window (on the right of figure 8). Here, you can choose the number of level lines of the cost you want to display.

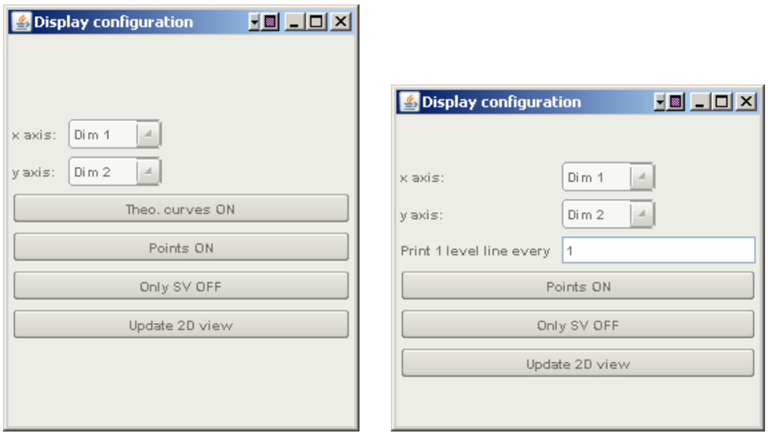


Figure 8: Configuration display window

Export file structure

The $.txt$ file contains the list of grid points and the SVM function parameters. The first part of the file is devoted to the list of the grid points. Each line corresponds to a point: each coordinate for all the dimensions are separated by a tab, and at the end of each line, separated by a tab from the coordinate of the last dimension, the label of the point ($+1$ if the point is viable, -1 if it is not). The list of points and the SVM function parameters are separated by an empty line. The SVM function is defined by a list of support vectors points: one point by line, coordinates separated by tab, at the end of each line, separated by a tab from the coordinates, the value of α_i (see glossary). The last line of the line corresponds to the value of b (see glossary).

Controller configuration window

Figure 9 presents the window of the configuration display.

This window allows the definition of the parameters of the controller. To use this button, you must have already run the algorithm until the final approximation was reached. In this window, you choose the starting point of the trajectory. You also choose the number of time steps of the trajectory. You have to choose the *number of time steps of anticipation* and the value of the SVM to define the *security distance*. This value must be greater than -1. A value of -1 value indicates that you allow no security distance (it correspond to the SVM value that define the boundary of the approximation). The value to choose depends on the security distance you want, and also on the number of points in each dimension. The finer the grid, the larger the SVM value must be. The greater the security distance, the bigger the value of the SVM must be. You can check the distance obtained in the display window, the security distance is colored light blue.

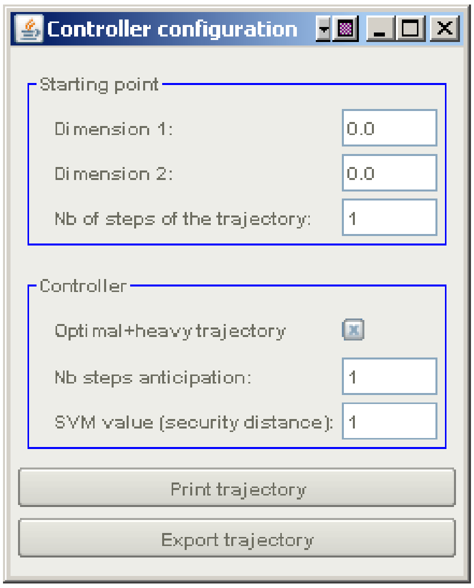


Figure 9: Viability controller window

Press the *Display trajectory* button to visualize the result. Press the *Export trajectory* button to export the list of points of the trajectory in a *.txt* file. This file contains the list of points, one line by point, and the coordinates are separated by tab.

In case of **viability kernel** approximation, in the *controller part*, the check box will indicate that you draw an heavy trajectory.

In case of **capture basin** approximation, in the *controller part*, the check box will indicate that you draw an optimal trajectory (that drives the system into the target in minimal time). Like this is an optimal controller, you don't have to choose the *number of steps of the trajectory*, the *number of steps anticipation* and the *security distance* (the text boxes are not enabled in this case).

In case of **resilience values** computation, in the *controller part*, the check box will indicate that you draw first an optimal trajectory (that drives the system into the viability kernel with a minimal cost) and then an heavy trajectory (once the system has came back) (see figure 9. Thus, the values you choose for the *number of steps of the trajectory*, the *number of steps anticipation* and the *security distance* are used only for the heavy controller part.

5.1.6 Indicators

The sixth part gives some indicators about the execution (figure 10. The indicatores are: the current *number of iterations* (for a step by step execution) or the final number of iterations (for execution mode), the *number of learns* (correspond to the number of computation of the SVM function, it is the same that the number of iterations in the version 1.0 of the algorithm, but will be different when the later version, including active learning, would be available), the final *time* of the approximation, the *number of points* of the whole grid, the *number of support vectors* of the current SVM (step by step mode) or the number of SVs of the final SVM (execution mode). The variable *M1* gives

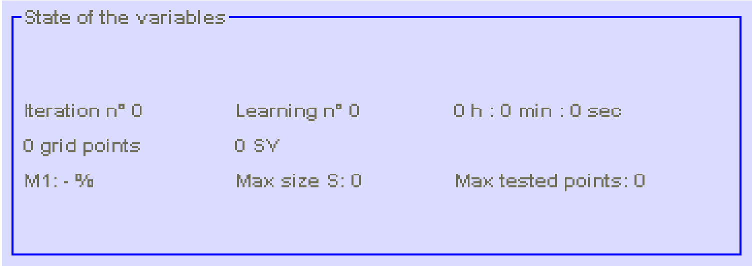


Figure 10: Sixth part of the console window

an indication of the progress of the algorithm. It indicates the number of grid points that have had their labels changed, relative to the size of the whole grid. A small value can mean that this is one of the last iterations of the approximation.

For active learning algorithm, you'll have also 2 more indicators: *max size S* that gives the maximal size of the learning set during the whole execution and *max tested points* that gives the maximal number of points that have been tested during the algorithm.

5.1.7 Log of the execution

The seventh part is the log of the execution. It summarizes the parameters chosen for the approximation, and during the run, indicates the progression of the algorithm. When the system is being controlled, it also indicates the current point of the trajectory and the control used. During the controller procedure, it also indicates the coordinates of the points of the trajectory and the control chosen at each time step.

5.2 Display window

2D view

The display window represents, in 2D, the current or the final approximation of the viability kernel. The viability kernel is represented in dark blue. The points of the grid are also displayed: a yellow point represents a viable point and a red point, a non-viable point. The black lines represent the boundary of the real viability kernel (if the theoretical curves are available). If resilience is computed and if chosen on the configuration display window, the level lines of the resilience values are drawn. The colors of the level lines are a function of the resilience values: blue (high resilience value) to red (low resilience value). The given resilience values does not correspond to the value of the point x, but to the value of an initial point that will jump to that point x.

To change the size of the window, drag the window corner to enlarge or diminish it and click 2 times on the drawing. To zoom without changing the size of the window, click 3 times. To come back to the initial size, click 2 times.

The window is composed by 2 parts: the configuration part and the visualization part.

In the visualization part, the approximation is represented. The $?$ button on the top right explains how to move the drawing.

The configuration part is composed by:

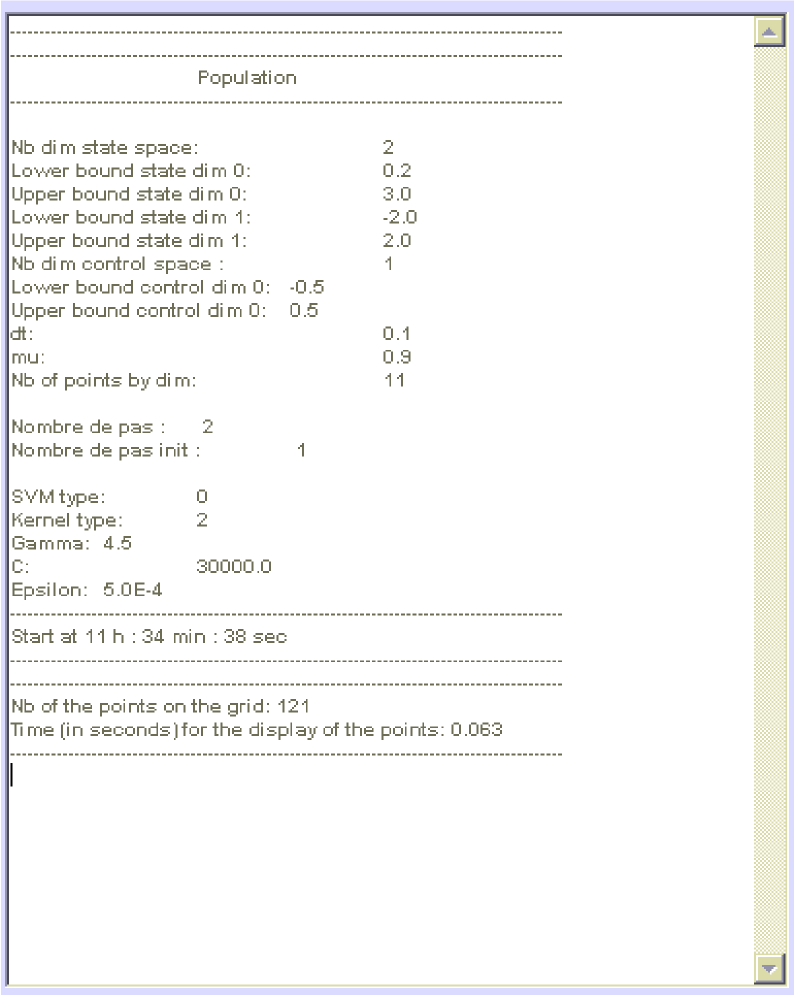


Figure 11: Seventh part of the console window

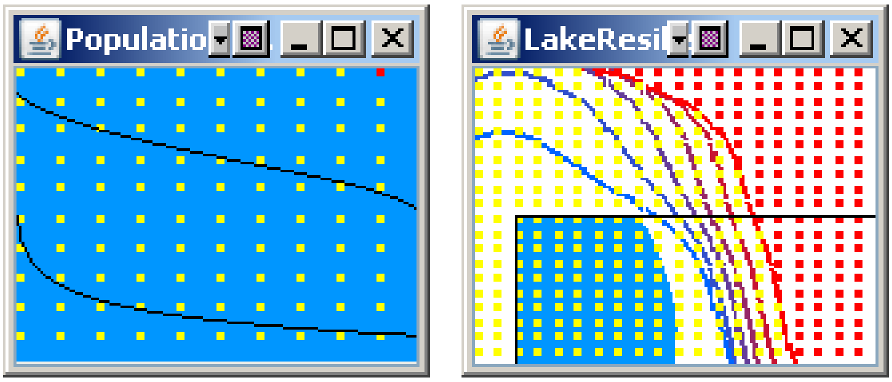


Figure 12: Display window

- *Version*:either color or black and white;
- *K espace ON*: click on the button to not display the boundaries of K, and click another time to display it;

- *Axis ON*: to display or not the axis;
- *Graduation*: to put some thicks on the axis (the greater the value, the more the number of graduations is);
- *Points ON*: to display or not the points of the grid;
- *Kernel ON*: click one time to change the colors, and another time to not display the kernel;
- *Theoric ON*: to display or not the theoretical curves, if available;
- *Precision*: a high value makes the precision of the drawing of the kernel better but is longer to print;
- *Axis*: to choose the axis;
- *Save*: to save the graphics in a .png file;
- *Update*: to update the visualization if modifications of the configuration.

6 User guide for adding a dynamical system

You have to create a new class file, with the name of your model (for example, *MyClass.java*). Depending on the results you want, you have to extend your class from:

- *Dynamic_System* if you want to approximate viability kernels;
- *Dynamic_System_Target* if you want to approximate capture basins;
- *Dynamic_System_Resilience* if you want to compute resilience values.

In this class, you have to create a main method to add your model and launch the software:

```
public static void main(String[] args) {
    // Init
    Kaviar kaviar = new Kaviar();
    // Optional: to add default models
    //kaviar.addModels(Kaviar.DEFAULT_MODEL);
    // Optional: to one of the default models
    //kaviar.addModels(Population.class);
    // replace MyModel by the name of your model
    kaviar.addModels(MyModel.class);
    // Launche the GUI
    kaviar.startGUI();
}
```

6.1 Approximating viability kernels

You have to define the constructor of *MyClass*, with the following fields in the same order:

N°	Fields	Type	Comments
1	$nbDim\_$	int	dimension of the state space
	$nbDimControl\_$	int	dimension of the control space
	$nbDimParam\_$	int	number of parameters
2	$initSystem()$	method	allow to initialize arrays of right size
3	$lowerBoundState\_[i]$	float	lower bound of the i^{th} dimension of K
	$upperBoundState\_[i]$	float	upper bound of the i^{th} dimension of K
	$labelDimStateSpace\_[i]$	float	label of the i^{th} dimension of K (optional)
	$lowerBoundControl\_[j]$	float	lower bound of the j^{th} control
	$upperBoundControl\_[j]$	float	upper bound of the j^{th} control
	$labelDimControlSpace\_[i]$	float	i^{th} control dimension label (optional)
	$labelParam\_[k]$	String	label of the k^{th} parameter
	$valueParam\_[k]$	float	value of the k^{th} parameter
4	$dt\_$	float	dt value
	$theoricCurves\_$	boolean	true if theoretical curves available
	$nbLineTheoric$	int	number of parts of the theoretical curves

Table 3: Fields for *Dynamic_System*

The type *MyClass.java* must implement the following inherited abstract method:

Method	**Params**	**Returns**
computePhi(Point inP, Point inU) *compute* $\varphi(x(t), u(t))$	inP: $x(t)$ inU: $u(t)$	Point $\varphi(x(t), u(t))$

6.2 Approximating capture basins

You have to define the constructor of *MyClass*, with the fields defined in table 3 plus (in the right order):

N°	**Fields**	**Type**	**Comments**
4	$tMax\_$	float	maximal time of capture

Table 4: tab:Fields for *Dynamic_System_Target*

The type *MyClass.java* must implement the following inherited abstract method:

Method	**Params**	**Returns**
computePhi(Point inP, Point inU) *compute* $\varphi(x(t), u(t))$	inP: $x(t)$ inU: $u(t)$	Point $\varphi(x(t), u(t))$
centerOfTheTarget(int i) *gives the* i^{th} *coordinate of the center of the target*	i: dimension	float i^{th} coordinate
isInTheTarget(Point inP) *tests if* $x(t) \in C$	inP: $x(t)$	boolean true if $x(t) \in C$

6.3 Computing resilience values

You have to define the constructor of *MyClass*, with the fields defined in table 3 plus (in the right order):

N°	**Fields**	**Type**	**Comments**
2	$nbCost\_$	int	number of parameters for the cost
4	$lowerK\_[l]$ $upperK\_[l]$ $valueCost\_[m]$ $cMax\_$ $dc\_$	float float float float float	lower bound of the l^{th} dimension of K upper bound of the l^{th} dimension of K value of the m^{th} cost maximal cost cost step

Table 5: tab:Fields for *Dynamic_System_Resilience*

The type *MyClass.java* must implement the following inherited abstract method:

Method	**Params**	**Returns**
computePhi(Point inP, Point inU) *compute* $\varphi(x(t), u(t))$	inP: $x(t)$ inU: $u(t)$	Point $\varphi(x(t), u(t))$
computeCost(Point inP) *gives the cost of a point* $x(t)$	inP: $x(t)$	float cost for $x(t)$

6.4 Changing the shape of K

By default, the viability constraint set is an hyper-rectangle. It is possible to change the shape of K by redefining the method $viab(Point\ inP)$ inside *MyClass*. In this case, the values of $lowerBoundState$ and $upperBoundState$ must define the coordinates of an hyper-rectangle in which is included your viability constraint set. Then, in the method $viab(Point\ inP)$, you have to specified if the point inP in parameter is located inside K.
An exemple can be found in the class $Bilingual.java$.

7 User guide for running the program in the batch mode

7.1 Running the program

To run the software in a batch mode, files with extension $.simu$ are needed. They contain all the information needed. The next subsection details the construction of a $.simu$ file.

To run the program in a batch mode, write the following instruction in a command window:

```
java -cp Kaviar-1.1.jar Appli/Batch Conso.simu
```

Replace $Conso.simu$ by the name of your file.
At the end of the execution, you'll have the results written in the command window:

```
Execution --> Conso Consumption December 11, 2007 5-31-44 PM CET/
Results
9 iterations
9 learns
18 SV
961 points
Computed in 0 h 0 m 9 s.
```

The execution creates 2 files: one with a $.svm$ extension, one with a $.log$ extension. The text in the console indicates which directory the files are saved in (here, in a directory called *Conso Consumption December 11, 2007 5-31-44 PM CET/*), the number of iterations and learns needed to obtain the final approximation, the number of support vectors of the last approximation and the size of the whole grid. It also indicates the time needed to compute the final approximation. To visualize the approximation, you have to run the program in the GUI mode, either by using the $.jar$ file or by executing the following instruction in a command window:

```
java -cp Kaviar-1.1.jar Appli/GUI
```

In the GUI console, use the *load project* button and select the $.svm$ file to visualize the results.

The batch mode also allows one to get the details of all the iterations, by using the parameter $-v$ in the instructions (equivalent to the step by step execution in GUI mode):

```
java -cp Kaviar-1.1.jar Appli/Batch Conso.simu -v
```

At the end of the execution, you'll have the results written:

```
Execution --> Conso Consumption December 11, 2007 5-36-28 PM CET/
Iter. 1 - M1 = 23.51717%
Iter. 2 - M1 = 9.781478%
Iter. 3 - M1 = 3.850156%
Iter. 4 - M1 = 1.2486992%
Iter. 5 - M1 = 0.7284079%
Iter. 6 - M1 = 0.41623312%
Iter. 7 - M1 = 0.20811656%
Iter. 8 - M1 = 0.10405828%
Iter. 9 - M1 = 0.0%
Results
9 iterations
9 learns
```

```
18 SV
961 points
Computed in 0 h 0 m 10 s.
```

Here, there are as many $.simu$ and $.svm$ files saved as the number of iterations.

7.2 Constructing a .simu file

The $.simu$ files must be carefully constructed. These files are composed of lines (one line per parameter to define). For each line, indicate the name of the parameter you want to define, and their values must be located at the right of ":". The names of the parameters are not important (they are only for recall), but they have to be defined in the right order. Table 6 gives the different components of the file.

The following code gives an example of a $.simu$ file for the lakeResilience system:

```
Name             : LanguageResilience
lower K dim 0    : 0.2
upper K dim 0    : 0.8
lower K dim 1    : 0
upper K dim 1    : 1
cost 0           : 1
cost 1           : 30
max cost         : 60
dc               : 0.1
lower dim 0      : 0
upper dim 0      : 1
lower dim 1      : 0
upper dim 1      : 1
lower control 0 : -0.1
upper control 0 : 0.1
a                : 1.31
dt               : 0.05
epsilon          : 0.0005
svm type         : 7
c                : 30000
kernel type      : 2
gamma            : 4
nbPtsByDim       : 81
nb steps         : 8
mu               : 0.5
outer approx     : true
passive          : true
grid depth       : 1
```

Lines	Description
$name$	name of the system, such that written in the attribute $names\_$ of the $Model$ class
$lower\ K\ i$ $upper\ K\ i$	value of the lower bound of K for dimension i (only for resilience computation) value of the upper bound of K for dimension i (only for resilience computation)
Comments	first lower and upper for dimension 1, lower and upper for dimension 2 etc.
$cost\ j$ $maxcost$ dc	value of the j^{th} cost (only for resilience computation) value of maximal cost (only for resilience computation) value of the cost step (only for resilience computation)
$lower\ dim\ k$ $upper\ dim\ k$	lower bound of K for the dimension k (lower bound of H if resilience computation) upper bound of K for the dimension k (upper bound of H if resilience computation)
Comments	first lower and upper for dimension 1, lower and upper for dimension 2 etc.
$lower\ control\ dim\ l$ $upper\ control\ dim\ l$	lower bound of the l^{th} control dimension upper bound of the l^{th} control dimension
$value\ of\ parameter\ m$	value of the m^{th} parameter (if there are parameters)
dt $tmax$	simulation time step maximal time of capture (only for capture basin approximation)
$epsilon$ $svm\ type$ C $kernel\ type$ $gamma$	tolerance of termination criterion for SVM computation must be 0 for *C SMO* 5 for Balk, 6 for SimpleSVM and 7 for softBalk value of C parameter must be 2 $gamma$ parameter for the SVM
$nb\ of\ pts\ by\ dim$ $number\ of\ steps$ mu $outer\ approx$ $passive$ $grid\ depth$	number of points by dimension number of steps mu value for gradient descent true if outer approximation, false for inner false if active learning algorithm grid depth for active learning

Table 6: Details of the parameters of a $.simu$ file

References

[Abrams and Strogatz, 2003] Abrams, D. and Strogatz, S. (2003). Modelling the dynamics of language death. *Nature*, 424(6951):900–900.

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

[Aubin, 2002] Aubin, J.-P. (2002). An introduction to viability theory and management of renewable resources. In *Coupling Climate and Economic Dynamics, Geneva*.

[Bernard et al., 2008] Bernard, C., Chapel, L., Deffuant, G., Martin, S., and San-Miguel, M. (2008). Maintaining viability and resilience of endangered languages. Working paper - european project PATRES.

[Calabrese and Deffuant, 2008] Calabrese, J. and Deffuant, G. (2008). First study of patres savana model viability and resilience. Working paper - european project PATRES.

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

[Carpenter et al., 1999] Carpenter, S.-R., Brock, W., and Hanson, P. (1999). Ecological and social dynamics in simple models of ecosystem management. *Conservation Ecology*, 3.

[Chapel and Deffuant, 2007] Chapel, L. and Deffuant, G. (2007). SVM viability controller active learning: Application to bike control. In *Proceedings of the IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning (ADPRL'07), Honolulu, USA, April 1-5*, pages 193–200.

[Chapel et al., 2007] Chapel, L., Martin, S., and Deffuant, G. (2007). Lake eutrophication: using resilience evaluation to compute sustainable policies. In *Proceedings of the 10th international conference on environmental science and technology (CEST 2007), Kos Island, Greece, September 5-7*, pages A204–A211.

[Chavalarias and Chapel, 2008] Chavalarias, D. and Chapel, L. (2008). Viability kernel in social dilemma. case study on emergence of cooperation on endogenous networks. Working paper - european project PATRES.

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

[Janssen and Carpenter, 1999] Janssen, M.-A. and Carpenter, S.-R. (1999). Managing the resilience of lakes: A multi-agent modeling approach. *Conservation Ecology*, 3(2).

[Loosli et al., 2008] Loosli, G., Deffuant, G., and Canu., S. (2008). Balk: Bandwidth autosetting for svm with local kernels. application to data on incomplete grids,. In *CAP08: Confŕence francophone sur l'apprentissage automatique*.

[Loosli et al., 2006] Loosli, G., Lee, S.-G., Guigue, V., Rakotomamonjy, A., and Canu, S. (2006). Perception d'états affectifs et apprentissage. *RIA - Revue d'intelligence artificielle, Edition spéciale Interactions Emotionnelles*.

[Martin, 2004] 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).

[Mathias et al., 2008] Mathias, J., Deffuant, G., and Mabrouk, N. (2008). Aggregation of individual based models of bacterial biofilms. Working paper - european project PATRES.

[Minett and Wang, IP] Minett, J. W. and Wang, W. S.-Y. (IP). Modelling endangered languages: The effects of bilingualism and social structure. *Lingua*.

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