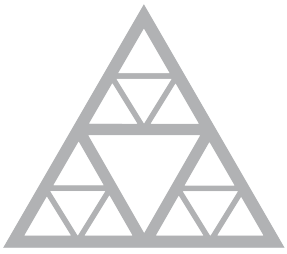
A stochastic viability approach to flooding prevention and adaptation

Clément Renault

June 19, 2013



École des Ponts
ParisTech

Contents

1 Problem statement

During this practical class we will consider a city that presents flooding risk. You work with the city council and you have two possibilities to protect the city: you can either build a seawall (or improve an existing one) or invest in rescue measures. You want to plan the decisions for the period $\mathbb{T} = [t_0, T]$ in order to minimize the risk of facing a dramatic flooding. We will consider time steps of one year (each year you decide where you want to invest and it takes you one year to improve your system). A flooding generates a cost that depends on its height. The city cannot afford more than a certain amount of money each year for flooding and this includes both the cost of the measures against flooding and the cost of flooding damages.

1.1 Adaptation and prevention dynamics

Building a seawall improves the city's *capacity* $C(t)$ to avoid a flood whereas investing in rescue processes improves the city's *adaptation* ability $K(t)$. During year t your city's state is defined by $x(t) = (C(t), K(t))$. The set of possible states for the city is $\mathbb{X}$. Each year you will have to choose between investing in one of those variables, investing in both or in none of them. Your control variable is defined by $u(t) = (c(t), k(t))$ with $\mathbb{U}$ the set of possible controls. Since the system can evolve we define a function Dyn from $\mathbb{T} \times \mathbb{X} \times \mathbb{U}$ to $\mathbb{X}$ that represents the influence of the control variable on the state variable :

$$x(t+1) = \mathsf{Dyn}(t, x(t), u(t)) \ . \tag{1}$$

Here the dynamic equations for your city are :

$$C(t+1) = C(t) + c(t) \ , \tag{2}$$

and

$$K(t+1) = K(t) + k(t) \ , \tag{3}$$

Of course you do not know the water level you will have to face on year $t+1$. This water level is represented by the variable $w(t+1) \in \mathbb{W}$.
On year t $H(C(t), w(t)) = w(t) - C(t)$ is the height of the flooding (if negative you do not have a flooding).

1.2 Viability thresholds on budget and seawall

There are two viability thresholds that represent the fact that you cannot spend too much money on flooding risk and that if the flooding is too important it will cause too much damage in the city:

- your budget is limited

$$\forall t, L(t, x(t), u(t), w(t)) \leq L_{max} \ , \tag{4}$$

- you do not want to have more a certain height of water in your city

$$\forall t, H(C(t), w(t)) \leq H_{max} \ . \tag{5}$$

At a given time t the set of states $x(t)$ that verify the viability constraints is noted $\mathbb{A}(t)$ and the set of controls that verify the viability constraints are noted $\mathbb{B}(t, x(t))$.

1.3 Viability kernels

Let's introduce the notion of policy. A policy Pol is a function from $\mathbb{T} \times \mathbb{X}$ to $\mathbb{U}$ so that $u(t) = \mathsf{Pol}(t, x(t))$ that gives the decision to make at every time and state of the city. The criterion chosen to evaluate a policy's quality is the probability of not respecting one of those viability thresholds during the considered period. To be even more precise we will determine the viability kernels of our system at a certain level of probability β :

$$\mathbb{V}iab_\beta(t_0) = \left\{ \begin{array}{c|l} \text{initial states} & \text{there exists a policy} \quad \mathsf{Pol} \\ x \in \mathbb{X} & \text{and controls} \quad u(\cdot) = (u(t_0), .., u(T-1)) \\ & \text{and states} \quad x(\cdot) = (x(t_0), .., x(T)) \quad \text{with} \quad x(t_0) = x \\ & \text{that verify} \quad t \in \{t_0, ..., T-1\} \\ & \text{the policy :} \quad u(t) = \mathsf{Pol}(t, x(t)) \\ & \text{the dynamic :} \quad x(t+1) = \mathsf{Dyn}(t, x(t), u(t), w(t)) \\ & \text{and so that :} \\ & \mathbb{P}\left(w(\cdot) \in \mathbb{W}^{T-t_0} \left| \begin{array}{l} \forall t \in [t_0, T-1], \\ L(t, x(t), u(t), w(t)) \leq L_{max}, \\ w(t) - C(t) \leq H_{max}, \\ w(T) - C(T) \leq H_{max} \end{array} \right. \right) \geq \beta \end{array} \right\} \tag{6}$$

These viability kernels can be numerically computed using dynamic programming. This matter is discussed in *Sustainable Management of Natural Resources. Mathematical Models and Methods* by Michel De Lara and Luc Doyen. We will now look at an implementation of this method to compute viability kernels. Then we will study the impact of thresholds variations on the maximal probability of respecting the viability constraints for a given state of the system.

2 Numerical evaluations

We consider that the random variables $(w(0), ..., w(T-1))$ are continuous, independent and identically distributed with a known distribution.

2.1 Basics

Let's start with the basic functions of our program.

```
//CONTROLS, we define here the controls we can have
c_state=[0:1:1];
k_state=[0:1:1];
//1 will be an upgrade of the corresponding equipment
//0 means we don't upgrade this equipment

//VIABILITY CONSTRAINTS
h_max= 150;//cm, this is the maximum height of water we
    want in our city
cost_max=1.5*10^7; //This is the maximum you can spend
   per year for the flooding

function [test]=test_viab(h, cost)
    test=bool2s(h<h_max)*bool2s(cost<cost_max+1);
endfunction
//This function verifies if your system respects the
   two viablity constraints

//SIMULATION DURATION
horizon=10; //in years
```

```
//POSSIBLE STATES
C_state=[0:20:200];
K_state=[0:0.1:1];

//FLOODING GENERATOR AND DAMAGE EVALUATION
//This first function evaluates the type of flooding we
    will have.
//It is based on historical data of water rise in
   rivers. We distinguish five types of rise.
function k =flooding_aux()
s=rand();
k=0;
if s>0.99 then k=4;
        elseif s>0.97 then k=3;
                elseif s>0.9366 then k=2;
                        elseif s>0.8366 then k=1;
end;
k;
endfunction;

//To each type of rise is assigned a numerical height
function [h]=flooding()
    k=flooding_aux();
    s=rand();
    h=250+s*110;
    if k==0 then h=s*80;
    elseif k==1 then h=80+s*35;
    elseif k==2 then h=115+s*55;
    elseif k==3 then h=170+s*80;
    end;
    h;
endfunction

//According to the city's capacity C we determine the
   height of water in the city.
function [d]=damage_level(C, h)
    h_city=max(0,h-C);
    d=0;
```

```
    if h_city>250 then d=4;
    elseif h_city>170 then d=3;
    elseif h_city>115 then d=2;
    elseif h_city>80 then d=1;
    end;
    d;
endfunction

//Then we evaluate the value of the damages caused to
   the city.
//This depends on the city's adaptation capacity K.
function [cost]=damage_value(d, K)
    cost=109.5*10^6;
    if d==0 then cost=0;
    elseif d==1 then cost=9.5*10^6;
    elseif d==2 then cost=15.8*10^6;
    elseif d==3 then cost=40.5*10^6;
    end;
    cost=(1-K)*cost;
endfunction

//CONSTRUCTION COSTS
w_k_max=10^7;
w_c_unit=10^7;
//We consider the cost of building 20cm of seawall is
   constant :
function [w_c]=cost_c(C)
    w_c=w_c_unit;
endfunction

//But going from K-1 to K gets more and more expensive
   as K tends to 1
//The first adaptation measures are easy to implement
   but then you need more complex systems.
function [w_k]=cost_k(K)
    w_k=w_k_max*K^2;
endfunction
```

2.2 The use of dynamic programming in order to compute viability kernels

Now we can focus on the dynamic programming equation. To do this we will first evaluate the probability to respect the viability constraints on year t given a city state and controls. This is what the next function does.

```
//Here we generate a vector of 10 000 river heights to
   compute the probabilities
vect_h=[];
for i=1:1:10000
    vect_h=[vect_h, flooding()];
end;


//Now we are able to determine the probability to be
   viable
//for a given state and controls
function [prob]=viab_proba_dyn()
    h=length(K_state);
    w=length(C_state);
    cc=length(c_state);
    kk=length(k_state);
    tot=length(vect_h);
//prob is our core matrix. It is tridimensional.
//The first two dimensions represent the state.
//The third dimension represents all the possible
   controls.
//We will fill this matrix with the probability of
   being viable for a state and controls on a year t
    prob=zeros(h,w,cc*kk); //initiation of prob : zeros
        everywhere
    for j=1:1:w //loop on the possible K states
        for i=1:1:h //loop on the possible C states
            for ccc=1:1:cc //loop on the possible C
               controls
                for kkk=1:1:kk //loop on the possible K
                    controls
```

```
                    try
//First we compute the cost of the controls
                        cost_controle=(ccc-1)*cost_c(
                            C_state(j+ccc-1))+(kkk-1)*
                            cost_k(K_state(i+kkk-1));
//We initiate the sum to evaluate the probability of
   being viable for each state and controls
                        sum_prob=0;
//We try all the values in vect_h and add 1 to sum when
    it is viable
//we don't forget the cost of the damages of a
   potential flooding
                        for k=1:1:tot
                            sum_prob=sum_prob+test_viab
                                (vect_h(k)-C_state(j),
                                damage_value(
                                damage_level(C_state(j),
                                vect_h(k)),K_state(i))+
                                cost_controle);
                        end;
//We represented the possible controls on only one
   dimension in prob.
//The third argument of prob is :
//1 : no investment in C or K
//2 : investment in C only
//3 : investment in K only
//4 : investment in C and K
                        prob(i,j,ccc+cc*kkk-cc)=
                            sum_prob/tot; //This gives
                            an experimental probability
                    catch
                        prob(i,j,ccc+cc*kkk-cc)=0;
                    end; //end of try
                end; //end of the loop on K controls
            end; //end of the loop on C controls
        end; //end of the loop on C states
    end; //end of the loop on K states
    prob
```

```
endfunction
```

This last function might take a while so you might wanna launch it before reading the rest of the document. The next step is to study all the possible trajectories from any state and to determine the one that maximizes the probability of being viable on the considered period of time.

```
function [prob_hor, optimal_path]=proba_viab_horizon(
    prob, horizon)
    h=length(K_state);
    w=length(C_state);
    cc=length(c_state);
    kk=length(k_state);
    tot=length(vect_h);
//To get the probability of being viable without
    investment
//(which is what we consider we do on the last year of
    the horizon)
//we extract from prob the matrix for each state and no
    investment.
//This corresponds to 1 for the third argument of prob.
    prob_hor=prob(:,:,1);
//We need two matrixes for the rest of the function :
//prob_hor is year t and prob_var year t-1
//but when we change of year they must be the same
//because we don't know yet the values for year t-1
    prob_var=prob_hor;
//We also want the optimal path from any state to
//maximize the probability of being viable
    optimal_path=zeros(h,w,horizon);
//Here we go backwards in time
    for tt=(horizon-1):(-1):0
//We begin a loop on the possible C states
        for j=1:1:w
//And on the possible K states
            for i=1:1:h
//We want the best probability knowing what happens
    next :
//we go backwards !
```

```
                    maxi=prob_hor(i,j)*prob(i,j,1);
//And when we find the best controls we write the
   optimal_path
                    optimal_path(j,i,tt+1)=1;
//We try all the controls (same system than before)
                    for l=2:1:4
                        try
                            var=prob_hor(i+bool2s(l==3|l
                               ==4),j+bool2s(l==2|l==4))*
                               prob(i,j,l);
//we catch if we are out of range in prob_hor
                        catch
                            var=0;
                        end//end of try
//If we found something better we write it
                        if var>maxi then
                            maxi=var;
                            optimal_path(j,i,tt+1)=l;
                        end //end of if
                    end //end of controls loop
                    prob_var(i,j)=maxi; //we found the
                       best probability for this state
                end //end of K states loop
            end //end of C states loop
            prob_hor=prob_var; //we copy prob_hor and
               prob_var
        end //end of time loop
endfunction
```

Now for each state we have the highest probability of being viable and the path to achieve that probability.

2.3 Graphical results

Let's take a look at what we produced.

```
f=scf();
x=0:1:10;y=x;plot3d(x,y,tab);
```

```
h=get("hdl");
h.color_flag=1;
f.color_map=hotcolormap(1000);
xtitle('Maximum viability probabilities for each
    beginning state of the city','K','C','P');
```

You should get something that looks like Figure 1.

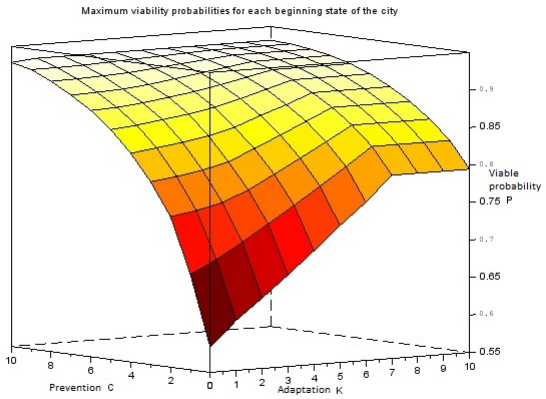


Figure 1: Maximum viability probabilities for each initial state

Let's now take a look at some viability kernels for different levels of probability of being viable. They are shown in Figure 2

```
contour2d([0:0.1:1],[0:20:200],tab
    ,[0.8,0.9,0.91,0.92,0.93]);
xtitle('Viability kernels','K','C');
```

2.4 Thresholds variations

We have studied how to evaluate the probability to respect the viability constraints for any given state of the city. Now we will focus on the impact of thresholds variations on this probability for a given state of the city.

```
function [tab]=probaviab_seuils(c,k,horizon)
    for h=1:11
        h_max=(h-1)*15;
```

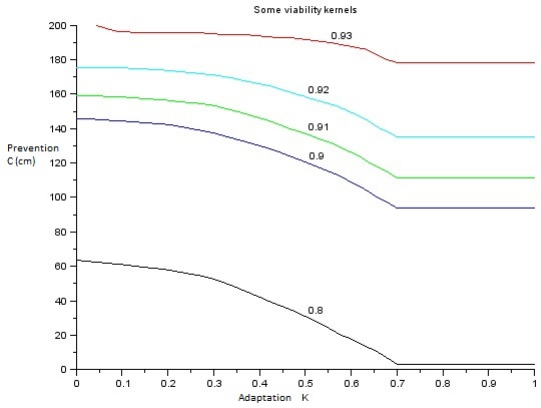


Figure 2: Viability kernels for different levels of probability of being viable

```
        for cost=1:11
            cost_max=(cost-1)*.15*10^7;
            prob=viab_proba_dyn();
            prob_hor=proba_viab_horizon(prob,horizon);
            tab(cost,h)=prob_hor(k,c);
        end
    end
endfunction
//It is important to notice that k and c are not the
   real values but the index numbers of the state in
   K_state and C_state
```

We can then plot the result.

```
f=scf();
x=0:1:10;y=x;plot3d(x,y,tab);
h=get("hdl");
h.color_flag=1;
f.color_map=hotcolormap(1000);
xtitle('Thresolds variations','Cost','Height','P')
```

This can take a really long time to compute. This code can be modified in order to save the result for all of the states at once (which is useful given the time it takes to compute). The result looks like this.

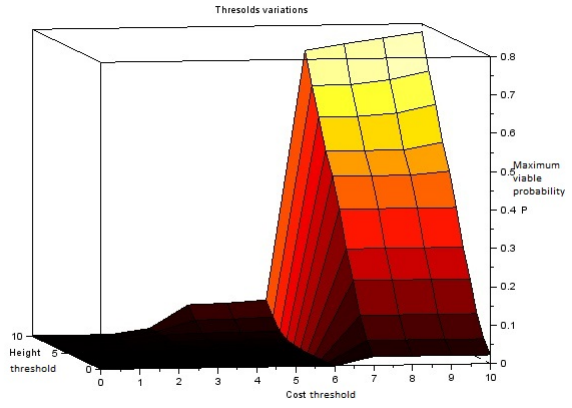


Figure 3: Impact of thresholds variation for a system in the state C=40cm, K=0.3. The corresponding indexes in C_state and K_state are respectively 3 and 4.