Presentation of the Stochastic Dynamic Dual Programming Algorithm

Adrien Cassegrain, Vincent Leclère

October 10, 2017

Contents

This practical work is aimed at presenting the theory of the SDDP algorithm and apply it on a simple dam management problem.

The Stochastic Dual Dynamic Programming (SDDP) algorithm is close to the Dynamic Programming method. The main difference being that, instead of computing the Bellman function at every point, the algorithm construct a polyhedral approximation of the Bellman function. More precisely the algorithm construct a piecewise linear function that is a lower approximation of the exact Bellman function.

This document is given in three parts:

- In §1, we present a few results about piecewise linear functions. We also recall the cutting plane algorithm, known as Kelley's algorithm.
- In §2, we present the SDDP algorithm.
- In §3, we present the dam management problem we want to tackle.
- Finally, in §4, we apply the SDDP algorithm on the dam management problem.

Before going any further you should download the following code:

1. Kelley,
2. DamData,
3. sdp,
4. CutPerAlea
5. sddp.

1 Preliminary

1.1 Piecewise linear functions

Recall that the supremum of a finite number of hyperplane is a convex piecewise linear function. We consider the function

$$\left\{ \begin{array}{ccl} f & : & \mathbb{R}^n \to \mathbb{R}, \\ f(x) & := & \max\limits_{i=1\dots k} \left(a_i \cdot x + b_i \right), \end{array} \right. \tag{1}$$

where $b_i \in \mathbb{R}$ and $a_i \in \mathbb{R}^n$ for $i \in \{1, \cdots, k\}$.

Question 1 *Find a new representation of f as a linear programm. That is $f(x)$ is the value of a minimization problem with linear cost and linear equality or inequality constraints.*

1.2 The cutting plane method: Kelley algorithm

The Kelley algorithm enables someone to approximate the minimum of a convex function f on a compact C only by having an oracle able to compute at each point its value and derivative. Recall that a convex function is always above its tangents. The cutting plane method consists in constructing a *lower approximation* $\check{f}$ of the objective function f as the maximum of tangents. This lower approximation is a piecewise-linear function, and is improved throughout the algorithm.

Let describe more precisely the algorithm. We start at a given point x_0. The first lower approximation is the tangent at this point,

$$\check{f}^1(x_0) := T_1(x_0) \, .$$

Then, at each step, we computes the equation of the tangent of f at the minimum of the approximate function $\check{f}$,

$$\begin{cases} \underline{x}_k & \in \quad \arg\min_{x \in C} \check{f}^{k-1}(x) \, , \\ T_k(x) & := \quad f'(\underline{x}_k) \cdot (x - \underline{x}_k) + f(\underline{x}_k) \, . \end{cases} \tag{2}$$

The new lower approximation $\check{f}^{k+1}$ is given by the maximum of the old one and the tangent,

$$\check{f}^{k+1}(x) := \max\left(\check{f}^k(x), T_k(x)\right) \, . \tag{3}$$

Question 2 *Run the algorithm in the file* `Kelley.sce` *that picture the initial function and the cutting planes found on the same graph. Evaluate graphically the difference between the minimum found and the real one.*

Test the algorithm for different convex function and for an increasing number of iteration. Comments.

2 The Structure of SDDP

In this section we present the Stochastic Dual Dynamic Programming (SDDP) algorithm. More precisely we present the so called DOASA implementation. For clarity reason we present it in the deterministic case in a first place, and then go to the stochastic case.

2.1 The Deterministic case

The SDDP algorithm in a deterministic setting is really close to the Kelley's algorithm on a multistage problem instead of a static problem.

2.1.1 Problem statement

We consider a controled dynamic system in discrete time that follows the equation

$$\forall t \in \{0, \dots, T-1\}, \qquad x_{t+1} = f_t(x_t, u_t),$$

where $x_t \in \mathbb{X}$ is the *state* at time t of the system and $u_t \in \mathbb{U}$ the *control* applied at time t; f_t is the *dynamics* of the system. We assume that the sets $\mathbb{U}$ and $\mathbb{X}$ are compact subset of a finite dimensionnal vector space. We also assume that the functions f_t are linear. At each time t there is a convex *cost* $L_t(x_t, u_t)$ that depends on the current state and the control choosen. Moreover there is a final cost $K(x_T)$ that depends on the final state of the system. A *policy* is a sequence of functions $\pi = (\pi_1, \dots, \pi_{T-1})$ giving for any state x a control u.

We consider the following problem

$$\begin{aligned} \min_{u \in \mathbb{U}^T} \qquad & \sum_{t=0}^{T-1} L_t(x_t, u_t) + K(x_T), \\ s.t. \qquad & x_{t+1} = f_t(x_t, u_t). \end{aligned} \tag{4}$$

This problem can be solved by dynamic programming.

Question 3 *Write the Dynamic Programming equation relative to this problem. How to compute the Bellman Value function ? How to compute the optimal policy ?*

2.1.2 Bellman operator

In order to understand the SDDP algorithm it is useful to introduce the Bellman operator $\mathcal{T}_t$ of the problem.

For any time t, and any function A mapping the set of states into $\mathbb{R}$ we define:

$$\forall x \in \mathbb{X}, \qquad \mathcal{T}_t(A)(x) = \min_{u_t \in \mathbb{U}} \quad L_t(x, u_t) + A \circ f_t(x, u_t). \tag{5}$$

Question 4 *Write the Dynamic Programming equation with the Bellman operator.*

Let study a few properties of the Bellman operator.

Monotonicity For any couple of functions $(V, \overline{V})$, we have

$$\forall x \in \mathbb{X}, \quad V(x) \leq \overline{V}(x) \quad \Rightarrow \quad \forall x \in \mathbb{X}, \quad \big(\mathcal{T} V\big)(x) \leq \big(\mathcal{T} \overline{V}\big)(x). \tag{6}$$

Convexity For any function V, if L_t is jointly convex in (x, u), V is convex, and f_t is affine then

$$x \mapsto \big(\mathcal{T} V\big)(x) \qquad \text{is convex.} \tag{7}$$

Linearity For any piecewise linear function V, if L_t is also piecewise linear, and f_t affine, then

$$x \mapsto \big(\mathcal{T}V\big)(x) \qquad \text{is piecewise linear.} \tag{8}$$

Question 5 *Prove these three results.*

2.1.3 Duality property

Let J be a function mapping $\mathbb{X} \times \mathbb{U}$ into $\mathbb{R}$. Consider the function φ defined as

$$\varphi(x) = \min_{u \in \mathbb{U}} J(x,u),$$

where J is jointly convex in (x,u). Then the interpretation of the multiplier as a marginal cost implies that a subgradient $\lambda \in \partial\varphi(x_0)$ of φ at x_0 is the dual multiplier of constraint $x_0 - x = 0$ in the following problem

$$\begin{aligned} &\min_{x,u} && J(x,u), \\ s.t. & && x_0 - x = 0. \end{aligned}$$

In particular it means that, as φ is convex

$$\forall x \in \mathbb{X}, \qquad \varphi(x) \geq \varphi(x_0) + \langle \lambda, x - x_0 \rangle. \tag{9}$$

2.1.4 SDDP algorithm (deterministic case)

As in the Kelley's algorithm, the SDDP algorithm recursively constructs a lower-approximation $\check{V}_t^{(k)}$ of each Bellman function V_t as the supremum of a number of affine functions. Each of this affine functions is called a *cut*.

Stage k of the SDDP algorithm goes as follows.

Suppose that we have a collection of T lower approximation of the Bellman functions denoted $\check{V}_t^{(k)}$ that is a maximum of affine functions. In order to improve our approximation we follow an optimal trajectory $(x_t^{(k)})_t$ of the approximated problem and add a cut computed along the optimal trajectory for each Bellman function.

Thus, we begin with a loop forward in time by setting $t = 0$ and $x_t^{(k)} = x_0$, and solve

$$\begin{aligned} \min_{x,u} \quad & L_t(x,u) + \check{V}_{t+1}^{(k)} \circ f_t(x,u), \\ & x = x_t^{(k)}. \end{aligned}$$

We call $\beta_t^{(k+1)}$ the value of the problem, $\lambda_t^{(k+1)}$ a multiplier of the constraint $x = x_t^{(k)}$, and $u_t^{(k)}$ an optimal solution. This can also be written as

$$\begin{cases} \beta_t^{(k+1)} & = \mathcal{T}_t\left(\check{V}_{t+1}^{(k)}\right)\left(x_t^{(k)}\right), \\ \lambda_t^{(k+1)} & \in \partial\mathcal{T}_t\left(\check{V}_{t+1}^{(k)}\right)\left(x_t^{(k)}\right). \end{cases}$$

Question 6 *Show that, by* (9), *and by monotonicity of* $\mathcal{T}_t$, *we obtain*

$$\forall x \in \mathbb{X}, \qquad \beta_t^{(k+1)} + \langle \lambda_t^{(k+1)}, x - x_t^{(k)} \rangle \leq \mathcal{T}_t\left(\check{V}_{t+1}^{(k)}\right)(x) \leq \mathcal{T}_t\left(V_{t+1}\right)(x) = V_t(x). \tag{10}$$

Thus we have by (9), and by monotonicity of $\mathcal{T}_t$,

$$\forall x \in \mathbb{X}, \qquad \beta_t^{(k+1)} + \langle \lambda_t^{(k+1)}, x - x_t^{(k)} \rangle \leq \mathcal{T}_t\left(\check{V}_{t+1}^{(k)}\right)(x) \leq \mathcal{T}_t\left(V_{t+1}\right)(x) = V_t(x). \tag{11}$$

This ensures that the function $x \mapsto \beta_t^{(k+1)} + \left\langle \lambda_t^{(k+1)}, x - x_t^{(k)} \right\rangle$ is indeed a cut, i.e an affine function below V_t. Consequently we update our approximate of V_t by defining

$$\check{V}_t^{(k+1)} = \max\left\{\check{V}_t^{(k)}, \beta_t^{(k+1)} + \left\langle \lambda_t^{(k+1)}, \cdot - x_t^{(k)} \right\rangle\right\}.$$

Furthermore we can see that $\check{V}_t^{(k+1)}$ is convex and lower than V_t. We go to the next time step by setting

$$x_{t+1}^{(k)} = f_t\left(x_t^{(k)}, u_t^{(k)}\right).$$

Upon reaching time $t = T$ we have completed iteration k of the algorithm.

2.1.5 Initialization and stopping rule

In order to initialize the algorithm it seems that we need a lower bound to all value function. According to our assumptions this bound always exist but is not necessary in order to implement the algorithm. Indeed we can choose $V_t^{(0)} = 0$ in order to compute the trajectories, and simply set $V_t^{(1)}$ equal to the first cut, which means that we "forget" $V^{(0)}$ in the maximum that determine $V_t^{(1)}$.

Note that at any step k of the algorithm we have a (non optimal) solution $(u^{(k)})_t$ of Problem (4), and we can estimate the quality of the solution. Indeed $V_0^{(k)}(x_0)$ is a lower bound to the optimal value of Problem (4). Moreover the simulated cost following the solution given by

$$\sum_{t=0}^{T-1} L_t\left(x_t^{(k)}, u_t^{(k)}\right) + K\left(x_T^{(k)}\right),$$

is an upper bound to the value of Problem (4). A reasonable stopping rule for the algorithm is given by checking that the (relative) difference of the upper and lower bound is smaller than some constant as they are both asymptotically exact bounds, i.e. both converges toward the value of Problem (4).

2.2 The Stochastic case

Now we introduce some random variables in our problem. This complexify the algorithm in different ways:

- we need some probabilistic assumptions;
- for each stage k we need to do a forward phase that yields a trajectory $(x_t^{(k)})_t$, and a backward phase that gives a new cut;
- we cannot compute an exact upper bound for the problem's value.

2.2.1 Problem statement

As in the deterministic case we consider a controled stochastic dynamic system in discrete time that follows the equation

$$\forall t \in \{0, \dots, T-1\}, \qquad x_{t+1} = f_t(x_t, u_t, W_t),$$

where $x_t \in \mathbb{X}$ is the *state* at time t of the system, u_t the *control* applied at time t, and W_t an exogeneous discrete *noise* ; f_t is the *dynamic* of the system.

We assume that the stochastic process $(W_t)_{t=1,\dots,T-1}$ is independent in time. Moreover we assume that at any time t we know the whole past of the noise process up to, and including, time t. The independence assumption ensures that we can only consider control u_t measurable with respect to (x_t, W_t). This is the so called hazard-decision information structure where we know at time t the realization of the noise before choosing our control.

At each time t there is a jointly convex in (x_t, u_t) *cost*, $L_t(x_t, u_t, W_t)$ that depends on the current state and the control choosen ; and there is a final cost $K(x_T)$ the depends on the final state of the system. A *policy* is a sequence of measurable functions $\pi = (\pi_1, \dots, \pi_{T-1})$ giving for any state x, and realization of the noise w a control u.

We consider the following problem

$$\begin{aligned} & \min_{\pi} && \mathbb{E}\left(\sum_{t=0}^{T-1} L_t(x_t, u_t, W_t) + K(x_T)\right), \\ & s.t. && x_{t+1} = f_t(x_t, u_t, W_t), \\ & && u_t = \pi_t(x_t, W_t). \end{aligned} \tag{12}$$

2.2.2 Bellman operator

As in the deterministic case we also introduce the stochastic Bellman operator $\mathcal{T}_t$ of the problem. For any time t, and any function A mapping the set of states and noises $\mathbb{X} \times \mathbb{W}$ into $\mathbb{R}$ we define:

$$\forall x \in \mathbb{X}, \qquad \widehat{\mathcal{T}}_t(A)(x, w) = \min_{u_t \in \mathbb{U}} L_t(x, u_t, w) + A \circ f_t(x, u_t, w).$$

The stochastic Bellman operator $\mathcal{T}_t$, is given by

$$\forall x \in \mathbb{X}, \qquad \mathcal{T}_t(A)(x) = \mathbb{E}\big(\widehat{\mathcal{T}}_t(A)(x, W_t)\big) .$$

Question 7 *Write the Dynamic Programming used to solve this problem in two ways: directly, and with the help of the Bellman operator.*

Question 8 *Show that the Bellman operator have the same properties as in the deterministic case.*

2.2.3 Duality theory

Suppose that for an iteration k and a time step $t+1$ we have a lower estimation V_{t+1}^{k+1} of V_{t+1}. We consider, for any w, the problem

$$\begin{aligned} \min_{x,u} \quad & L_t(x,u,w) + \check{V}_{t+1}^{(k+1)} \circ f_t(x,u,w), \\ s.t \quad & x = x_t^{(k)}, \end{aligned} \tag{13}$$

and call $\widehat{\beta}_t^{(k+1)}(w)$ the optimal value, and $\widehat{\lambda}_t^{(k+1)}(w)$ an optimal multiplier of the constraint. As in the deterministic case we have

$$\begin{cases} \widehat{\beta}_t^{(k+1)}(w) & = \widehat{\mathcal{T}}_t\left(\check{V}_{t+1}^{(k)}\right)(x_t^{(k)}, w), \\ \widehat{\lambda}_t^{(k+1)}(w) & \in \partial_x \widehat{\mathcal{T}}_t\left(\check{V}_{t+1}^{(k)}\right)(x_t^{(k)}, w), \end{cases} \tag{14}$$

Question 9 *Show that we have*

$$\forall x \in \mathbb{X}, \quad \forall \omega, \qquad \widehat{\beta}_t^{(k+1)}(w) + \left\langle \widehat{\lambda}_t^{(k+1)}(w), x - x_t^{(k)} \right\rangle \leq \widehat{\mathcal{T}}_t\left(\check{V}_{t+1}^{(k)}\right)(x,w) \leq \widehat{V}_t(x,w)\ . \tag{15}$$

By defining

$$\begin{aligned} \beta_t^{(k+1)} &:= \mathbb{E}\left(\beta_t^{(k+1)}(W_t)\right) \\ \lambda_t^{(k+1)} &:= \mathbb{E}\left(\lambda_t^{(k+1)}(W_t)\right) \end{aligned}$$

show that we have

$$\beta_t^{(k+1)} + \left\langle \lambda_t^{(k+1)}\ , \cdot - x_t^{(k)} \right\rangle \leq V_t\ . \tag{16}$$

2.2.4 SDDP algorithm (stochastic case)

At the beginning of step k we suppose that we have, for each time step t an approximation $\check{V}_t^{(k)}$ of V_t verifying

- $\check{V}_t^{(k)} \leq V_t$,
- $V_T^k = K$,
- $\check{V}_t^{(k)}$ is convex.

Forward path We randomly select a scenario $(w_0, \dots, w_{T-1}) \in \mathbb{W}^T$. We recursively, by a forward loop in time, define a trajectory $(x_t^{(k)})_{t=0,\dots,T}$ by $x_{t+1}^{(k)} = f_t(x_t^{(k)}, u_t^{(k)}, w_t)$, where $u_t^{(k)}$ is an optimal solution of

$$\min_{u \in \mathbb{U}} L_t\left(x_t^{(k)}, u, w_t\right) + \check{V}_{t+1}^{(k)} \circ f_t\left(x_t^{(k)}, u, w_t\right). \tag{17}$$

This trajectory is a trajectory given by the optimal policy of Problem (12) where V_t is replaced by its current approximation $\check{V}_t^{(k)}$.

Backward path We reader improve the approximation of V_t by adding a cut at $V_t(x_t^{(k)})$. We procede backward in time by setting $t = T-1$, and solve Problem (13) for every w in the support of W_t. We obtain $\widehat{\lambda}_t^{(k+1)}(w)$ and $\widehat{\beta}_t^{(k+1)}(w)$. As W_t is discrete we can compute exactely the expectation $\lambda_t^{(k+1)}$ and $\beta_t^{(k+1)}$, and define the new lower approximation

$$\forall x \in \mathbb{X}, \qquad \check{V}_t^{(k+1)}(x) = \max\left\{\check{V}_t^{(k)}(x), \beta_t^{(k+1)} + \left\langle \lambda_t^{(k+1)}, x - x_t^{(k)} \right\rangle\right\}$$

Question 10 *Check that $\check{V}_t^{(k+1)}$ is a lower convex approximation of V_t.*

Go one step back in time: $t \leftarrow t-1$. Upon reaching $t = 0$ we have completed step k of the algorithm.

Question 11 *For one iteration of the SDDP, algorithm, how many one-step one-alea problem (Problem* (17)*) do we need to solve ?*

2.2.5 Initialization and stopping rule

We can intialize as suggested in the deterministic case. However in order to accelerate the convergence it can be useful to bypass a few forward paths by abritrarily choosing some trajectories $(x_t^{(k)})_t$.

As in the deterministic case we have an exact lower bound of Problem (12) given by $V_0^{(k)}(x_0)$. However, determining an upper bound is more difficult. Indeed, the upper bound is given as the expectation of the cost induced by the optimal strategy of the approximated problem. This expectation is to be taken on the whole process $(W_0, \dots, W_{T-1})$ and cannot be computed exactly. Consequently this expectation is estimated by Monte-Carlo methods, and we have no control over the error of our solution.

A heuristic stopping rule consist in stopping the algorithm if the lower bound is in the confidence interval of the upper bound for a determined number of Monte-Carlo simulation.

2.2.6 A few other possibilities

In the algorithm presented here, also called DOASA, we select one scenario (one realization of $(W_1, \dots, W_{T-1})$) to do a forward and backward path. It is also possible to select a number N of scenarios to do the forward path (computation can be parallelized). Then during the backward path we add N cuts to V_t before computing the cuts on V_{t-1}. This is the original implementation of the algorithm.

Otherwise the CUPPS algorithm suggest to use $\check{V}_{t+1}^{(k)}$ instead of $V_{t+1}^{(k+1)}$ in the computation of the cuts (14). In practice it means that we do only a forward path at step k:

- select randomly a scenario $(w_t)_{t=0,\dots,T-1}$;
- at time t we have a state $x_t^{(k)}$, we solve problem (13) for every possible realization of W_t in order to compute the new cut for V_t;

- choose the optimal control corresponding to the realization $W_t = w_t$ in order to compute the state $x_{t+1}^{(k)}$ where the cut for V_{t+1} will be computed, and goes to the next step.

Moreover it is possible to select more than one scenario, and everything can be parralelized in this implementation.

Finally it is always possible, and most of the time fruitful to compute some cuts before starting the algorithm. One way of doing this is to bypass the forward phase by choosing the trajectory $(x_t^{(k)})_{t=0,\dots,T}$ instead of computing it as some optimal trajectory.

3 Problem data

We consider a dam manager intenting to minimize the intertemporal cost obtained by turbining the water in a dam, when the water inflow (rain, outflows from upper dams) is supposed to be stochastic.

3.1 Dam dynamics

$$\underbrace{S_{t+1}}_{\text{future volume}} = \underbrace{S_t}_{\text{volume}} - \underbrace{q_t}_{\text{turbined}} - \underbrace{\delta_t}_{\text{spilled}} + \underbrace{a_t}_{\text{inflow}} , \quad t \in \mathbb{T} := \{t_0, \dots, T-1\}$$

with

- time $t \in \overline{\mathbb{T}} := \{t_0, \dots, T\}$ is discrete (days), and t denotes the beginning of the period $[t, t+1[$,
- S_t volume (stock) of water at the beginning of period $[t, t+1[$, belonging to the set $\mathbb{X} = [0, S^\sharp]$, made of water volumes (h m^3), where $S^\sharp$ is the maximum dam volume,
- a_t inflow water volume (rain, etc.) during $[t, t+1[$, belonging to $\mathbb{W} = \{0, 1, 2, \dots, a^\sharp\}$, and supposed to be known at the beginning of the period (Hazard-Decision),
- q_t turbined outflow volume during $[t, t+1[$, decided at the beginning of period $[t, t+1[$, belonging to the set $\mathbb{U} = [0, q^\sharp]$, where $q^\sharp$ is the maximum which can be turbined by time unit (and produce electricity),
- δ_t spilled outflow volume during $[t, t+1[$, decided at the beginning of period $[t, t+1[$, belonging to the set $\mathbb{D} = \mathbb{R}_+$, and such that

$$0 \leq q_t + \delta_t \leq S_t + a_t \ . \tag{18}$$

The dam manager is supposed to make a decision, here turbining q_t and spilling δ_t, at the beginning of the period $[t, t+1]$, *after* knowing the water inflow a_t.

A water inflows scenario is a sequence as

$$a := \big(a_{t_0}, \dots, a_{T-1}\big). \tag{19}$$

Compared to other methods, we have added the spilled outflow volume δ in order to get a linear dynamics and we are placed in the Hazard-Decision case. These two points are mandatory to apply the SDDP algorithm.

3.2 Criterion: intertemporal cost

The manager original problem is one of *cost minimization* where turbining one unit of water has unitary cost $-p_t$. On the period from t_0 to T, the costs sum up to

$$\sum_{t=t_0}^{T-1} -p_t q_t + K\big(S_T\big) \ , \tag{20}$$

where

- a *scenario* of costs
$$-p(\cdot) = \big(-p_{t_0}, \dots, -p_{T-1}\big) \tag{21}$$
is given (data of the problem) and supposed to be known in advance (deterministic setting),
- the final term $K\big(S_T\big)$ gives value to the water volume in the dam at the horizon T.

3.3 Probabilistic model for water inflows

We suppose that sequences of uncertainties $(a_{t_0}, \dots, a_{T-1})$ are random variables with a known probability distribution $\mathbb{P}$ on the set $\{0, \dots, a^\sharp\}$, with $\mathbb{P}\{a_t = a\}$ given.

Notice that the random variables $(a_{t_0}, \dots, a_{T-1})$ are independent, but that they are not necessarily identically distributed. This allows us to account for seasonal effects (more rain in autumn and winter).

3.4 Numerical data

We now perform numerical simulations, and try different policies.

Time. We consider a daily management over one year

$$t_0 = 1 \quad \text{and} \quad T = 365 \ , \tag{22}$$

Bounds. Concerning the dam and water inflows, we consider the following bounds:

$$S_0 = 0 \text{ hm}^3 \ , \quad S^\sharp = 100 \text{ hm}^3 \ , \quad q^\sharp = \frac{0.4}{7} \times S^\sharp \quad \text{and} \quad a^\sharp = \frac{0.5}{7} \times S^\sharp. \tag{23}$$

These figures reflect the assumptions that, during one week, one can release at maximum 40% of the dam volume, and that during one week of full water inflows, an empty dam can be half-filled.

Costs scenario. The scenario $-p(\cdot) = \big(-p_{t_0}, \dots, -p_{T-1}\big)$ of costs is known in advance. We produce it by one sample from the expression

$$-p_t = (1+\varepsilon(t))\overline{-p} \quad \text{with} \quad \overline{-p} = -66\text{MWh/hm}^3 \times 2.7 \text{ euros/MWh}^3 \tag{24}$$

where $\varepsilon(t)$ is drawn from a sequence of i.i.d. uniform random variables in $[-1/2, 1/2]$.

Water inflow scenarios. We produce scenarios $a(\cdot) := \big(a_{t_0}, \dots, a_{T-1}\big)$ of water inflows by a sequence of independent random draws from the set $\{0, 1, \cdots, a^\sharp\}$.

3.5 Water turbined policy

An admissible policy $\pi : \mathbb{T} \times \mathbb{X} \times \mathbb{W} \to \mathbb{U} \times \mathbb{D}$ assigns an amount of water turbined and spilled $(q, \delta) = \pi_t(S, a)$ to any state S of dam stock volume, to any decision period $t \in \mathbb{T}$ and to any water inflow $a \in \mathbb{W}$, while respecting the constraints

$$\begin{cases} 0 & \leq q_t \leq q^\sharp \ , \\ 0 & \leq \delta_t \ , \\ 0 & \leq q_t + \delta_t \leq S_t + a_t \ , \\ 0 & \leq S_t + a_t - q_t - \delta_t \leq S^\sharp \end{cases} \tag{25}$$

that can also be written

$$\begin{cases} -q_t & & & \leq & 0 \ , \\ q_t & & & \leq & q^\sharp \ , \\ & -\delta_t & & \leq & 0 \ , \\ q_t & +\delta_t & -S_t & \leq & a_t \ , \\ -q_t & -\delta_t & +S_t & \leq & S^\sharp - a_t . \end{cases} \tag{26}$$

The system (26) is equivalent to

$$\underbrace{\begin{pmatrix} -1 & 0 \\ 1 & 0 \\ 0 & -1 \\ 1 & 1 \\ -1 & -1 \end{pmatrix}}_{A} \cdot \begin{pmatrix} q_t \\ \delta_t \end{pmatrix} + S_t . \underbrace{\begin{pmatrix} 0 \\ 0 \\ 0 \\ -1 \\ 1 \end{pmatrix}}_{E} \leq \underbrace{\begin{pmatrix} 0 \\ q^\sharp \\ 0 \\ a_t \\ S^\sharp - a_t \end{pmatrix}}_{b(t)} . \tag{27}$$

In the following, we will simply write as defined $A \cdot \begin{pmatrix} q \\ \delta \end{pmatrix} + S_t . E \leq b(t)$.

Once the policy given, we obtain a volume trajectory $S(\cdot) = (S_{t_0}, \cdots, S_{T-1}, S_T)$, a turbined trajectory $q(\cdot) = (q_{t_0}, \cdots, q_{T-1}, q_T)$ and a spilled trajectory $\delta(\cdot) = (\delta_{t_0}, \cdots, \delta_{T-1}, \delta_T)$ produced by the "closed-loop" dynamics

$$\left\{\begin{aligned} S_{t_0} &= S_0 \ , \\ S_{t+1} &= S_t + a_t - q_t - \delta_t \ , \\ \big(q_t, a_t\big) &= \pi_t\big(S_t, a_t\big) \ . \end{aligned}\right. \tag{28}$$

and function of the scenario $a(\cdot)$. Thus, in the end, we obtain the cost

$$\mathsf{Crit}^{\pi_{t_0}}\big(S_0, a_{t_0}, \cdots, a(T)\big) := \sum_{t=t_0}^{T-1} -p_t q_t - K\big(S_T\big) \ . \tag{29}$$

3.6 Minimization under constraints

The aim of this part is to compute the minimum of a given convex function under linear constraints and to find a cutting plane. We will use this to find an optimal strategy.

The problem is

$$\begin{aligned} V_t^k(z) = \min_{q,\delta} \quad & L_t(z,q,\delta,a_t) + V_{t+1}^k \circ f(z,q,\delta,a_t), \\ s.t. \quad & A \cdot \begin{pmatrix} q \\ \delta \end{pmatrix} + E \cdot z \leq b(t) \end{aligned} \tag{30}$$

In order to use the linear solver, we have to transform this formula into one of the form

$$\begin{aligned} \min_{u'} \quad & C \cdot u' \ , \\ s.t. \quad & A' \cdot u + E' \cdot z \leq b'(t) \ . \end{aligned} \tag{31}$$

We remind the reader that V_{t+1}^k is a piecewise linear function written

$$V_{t+1}^k(z) = \max_{0 \leq i \leq k} \quad \{\lambda_i \cdot z + \beta_i\} \ ,$$

which is equivalent to (as seen in § 1.1)

$$\begin{aligned} V_{t+1}^k(z) = \min_{\gamma} \quad & \gamma \ , \\ s.t. \quad & \gamma \geq \lambda_i \cdot z + \beta_i, \qquad \forall i \in \{1, \dots, k\} \end{aligned} \tag{32}$$

By writting

$$u = \begin{pmatrix} q \\ \delta \\ \gamma \end{pmatrix}, \quad A'_k = \begin{pmatrix} A & & 0 \\ -\lambda_1 & -\lambda_1 & -1 \\ \vdots & \vdots & \vdots \\ -\lambda_k & -\lambda_k & -1 \end{pmatrix}, \quad E'_k = \begin{pmatrix} E \\ \lambda_1 \\ \vdots \\ \lambda_k \end{pmatrix} \text{ and } \quad b'_k(t) = \begin{pmatrix} b(t) \\ -\beta_1 - \lambda_1 \cdot a_t \\ \vdots \\ -\beta_k - \lambda_k \cdot a_t \end{pmatrix},$$

we obtain the wanted form for the minimization problem (see (31)).

The last step is to find the parameters of the cutting plane of the Bellman function V_t^k at a given point z. To do so, we transform the state z in the previous equation in a variable

x and add the constraint $x = z$. The Lagrange multiplier of this equality constraint is the slope of a new cutting plane and the minimum cost is the ordinate at the origin.

The following macro returns the parameters of the plane (λ, β) and the point at which the minimum is reached u_{opt} given the matricial form of the cost $(-p, V)$, the constraints $(A, b(t), E)$, the value of the uncertainty a, the dynamics given by (Q, R) and the state z.

Question 12 *Explain the objective of the function **`CutPerAlea`**. Where, in the SDDP algorithm, is this function required ? How many times is the function called for a step k of the SDDP algorithm ?*

4 Dam optimal management

We now solve the problem presented earlier in this practical work, by Dynamic Programming and by Stochastic Dual Dynamic Programming.

Question 13 *First of all open the file **`damdata.sce`**, and understand the different macro given in this file, in particular the macro **`trajectories`**.*

*Now define a policy (a function taking a time t, a current stock s and an inflow a and giving an admissible control), and test it. Plot (using **`plot2d2`**) the trajectories of the stock with this policy.*

4.1 DP resolution

We solve exactly the problem using Dynamic Programming.

Question 14 *In the file `sdp.sce` a Dynamic Programming approach is presented. Understand how the function works, and plot the optimal stock trajectory using function from `damdata.sce`.*

From the function of `sdp.sce` what is the exact value of the optimization problem ? Simulate the optimal strategy and give an estimation of this value and asymptotic error bounds.

What do you observe for the final stock? Explain why.

4.2 SDDP resolution

We now apply an SDDP approach to the same problem.

In the files that were downloaded at the beginning, the function `SDDP` creates an approximate of the Bellman function given the constraints and random trajectories, then the function `sddp_policy` defines the optimal policy given the former approximate.

Question 15 *Using the function of **`SDDP`** picture the trajectory of the stocks corresponding to the optimal policy. Evaluate the optimal payoff, and the error bounds.*

Evaluate the lower bound given by SDDP. Deduce the estimated gap (in %).

Plot the improvement of the gap with iterations of SDDP.

4.3 Comparison of SDDP and DP

Question 16 *Compare the numerical results given by the DP approach, and the SDDP approach.*

Sum up the theoretical and practical differences between DP and SDDP: what are the pros and cons of each method.