Value functions in `LinearDecisionRules.jl`

Bernardo Freitas Paulo da Costa (FGV)

with Joaquim Garcia (PSR)

ICSP 2025, Paris

Value functions in LinearDecisionRules.jl

Bernardo Freitas Paulo da Costa (FGV)

with Joaquim Garcia (PSR)

ICSP 2025, Paris

2-Stage Stochastic Programming

Basic model:

$$
\begin{array}{rll}
\min & & \mathbb{E}\left[c(\xi)^\top y(\xi)\right] \\
\text{s.t.} & Ax = b, & Tx + Wy(\xi) = h(\xi) \\
 & x \geqslant 0, & y(\xi) \geqslant 0,
\end{array}
$$

- x is here-and-now, y is wait-and-see;
- c are (possibly random) costs.

2-Stage Stochastic Programming

Basic model:

$$\begin{array}{lll} \min & & \mathbb{E}\left[c(\xi)^\top y(\xi)\right] \\ \text{s.t.} & Ax = b, & Tx + Wy(\xi) = h(\xi) \\ & x \geqslant 0, & y(\xi) \geqslant 0, \end{array}$$

- x is here-and-now, y is wait-and-see;
- c are (possibly random) costs.

2-Stage Stochastic Programming

Linear Decision Rules

- Fix a parametrization where $c(\xi) = C\xi$ and $h(\xi) = H\xi$ are linear in ξ.
- Posit a linear decision rule: $y(\xi) = Y \cdot \xi$.

2-Stage Stochastic Programming

Linear Decision Rules

- Fix a parametrization where $c(\xi) = C\xi$ and $h(\xi) = H\xi$ are linear in ξ.
- Posit a linear decision rule: $y(\xi) = Y \cdot \xi$.

Reduces the flexibility of the "wait-and-see" decision

$$
\begin{array}{lll}
\min & & \mathbb{E}\left[\xi^\top C^\top Y \xi\right] \\
\text{s.t.} & Ax = b & Tx + WY\xi = H\xi \quad \forall \xi \\
& x \geqslant 0 & Y\xi \geqslant 0,
\end{array}
$$

2-Stage Stochastic Programming

Linear Decision Rules

- Fix a parametrization where $c(\xi) = C\xi$ and $h(\xi) = H\xi$ are linear in ξ.
- Posit a linear decision rule: $y(\xi) = Y \cdot \xi$.

Reduces the flexibility of the "wait-and-see" decision, but allows for a *finite* problem if the support of ξ is simple, say $\Xi = \{\, \xi \mid G\xi \geqslant f \,\}$.

$$
\begin{array}{lll}
\min & & \mathbb{E}\left[\xi^\top C^\top Y \xi\right] = \operatorname{Tr}\left(\mathbb{E}[\xi\xi^\top] C^\top Y\right) \\
\text{s.t.} & Ax = b & Tx + WY = H \\
 & x \geqslant 0 & Y = \Lambda G,\ \Lambda \geqslant 0,\ \Lambda f \geqslant 0,
\end{array}
$$

2-Stage Stochastic Programming

Linear Decision Rules

- Fix a parametrization where $c(\xi) = C\xi$ and $h(\xi) = H\xi$ are linear in ξ.
- Posit a linear decision rule: $y(\xi) = Y \cdot \xi$.

Reduces the flexibility of the "wait-and-see" decision, but allows for a *finite* problem if the support of ξ is simple, say $\Xi = \{\, \xi \mid G\xi \geqslant f \,\}$.

$$
\begin{array}{lll}
\min & & \mathbb{E}\left[\xi^\top C^\top Y \xi\right] = \operatorname{Tr}\left(\mathbb{E}[\xi\xi^\top] C^\top Y\right) \\
\text{s.t.} & Ax = b & Tx + WY = H \\
 & x \geqslant 0 & Y = \Lambda G,\ \Lambda \geqslant 0,\ \Lambda f \geqslant 0,
\end{array}
$$

The distribution of ξ appears only in the objective function, and through the 2nd-moment matrix $\mathbb{E}\left[\xi\xi^\top\right]$.

2-Stage Stochastic Programming

Sample Average Approximation: SAA

Kelley's cutting plane / L-shaped method builds approximations to the value function

$$
\begin{aligned}
V(x) = \min_{y(\xi)} \quad & \mathbb{E}\left[c(\xi)^\top y(\xi)\right] \\
\text{s.t.} \quad & W y(\xi) = h(\xi) - Tx, \\
& y(\xi) \geqslant 0.
\end{aligned}
$$

2-Stage Stochastic Programming

Sample Average Approximation: SAA

Kelley's cutting plane / L-shaped method builds approximations to the value function

$$V(x) = \min_{y(\xi)} \quad \mathbb{E}\left[c(\xi)^\top y(\xi)\right]$$
$$\text{s.t.} \quad Wy(\xi) = h(\xi) - Tx,$$
$$y(\xi) \geqslant 0.$$

Key points:

- $V(x)$ is a *convex function* of x, so $V(x) \geqslant V(x_0) + \nabla V(x_0)^\top (x - x_0)$ for any x_0;
- The gradient $\nabla V(x_0)$ is given by the *optimal dual solution* of the SAA problem at x_0.

2-Stage Stochastic Programming

Sample Average Approximation: SAA

Kelley's cutting plane / L-shaped method builds approximations to the value function

$$V(x) = \min_{y(\xi)} \quad \mathbb{E}\left[c(\xi)^\top y(\xi)\right]$$
$$\text{s.t.} \quad Wy(\xi) = h(\xi) - Tx,$$
$$y(\xi) \geqslant 0.$$

Key points:

- $V(x)$ is a *convex function* of x, so $V(x) \geqslant V(x_0) + \nabla V(x_0)^\top (x - x_0)$ for any x_0;
- The gradient $\nabla V(x_0)$ is given by the *optimal dual solution* of the SAA problem at x_0.

Exchanging $\min$ and $\mathbb{E}$:

$$V(x) = \mathbb{E}\left[\min_{y \in Y(x,\xi)} c(\xi)^\top y(x,\xi)\right]$$

shows that *y is a feedback control*: depends on the uncertainty ξ and the 1st-stage decision x.

Multistage problems

- The 2-stage problem can be generalized to *multistage* problems, where the uncertainty ξ is observed at different stages: $\xi_1, \xi_2, \ldots, \xi_T$;
- The decision $y_t(\xi)$ at stage t can depend on the uncertainty observed at previous stages.

Multistage problems

- The 2-stage problem can be generalized to *multistage* problems, where the uncertainty ξ is observed at different stages: $\xi_1, \xi_2, \ldots, \xi_T$;
- The decision $y_t(\xi)$ at stage t can depend on the uncertainty observed at previous stages.
- Typically, there is a *state* $x_t(\xi)$ at each stage, depending on the realization of the uncertainties and the decisions y_t.

Multistage problems

- The 2-stage problem can be generalized to *multistage* problems, where the uncertainty ξ is observed at different stages: $\xi_1, \xi_2, \ldots, \xi_T$;
- The decision $y_t(\xi)$ at stage t can depend on the uncertainty observed at previous stages.
- Typically, there is a *state* $x_t(\xi)$ at each stage, depending on the realization of the uncertainties and the decisions y_t.

A multistage problem can be written as:

$$\begin{aligned} \min \quad & \mathbb{E}\left[\textstyle\sum_{t=1}^{T} c_t(\xi)^\top y_t(\xi)\right] \\ \text{s.t.} \quad & Ax_t(\xi) + Bx_{t-1}(\xi) + Wy_t(\xi) = h_t(\xi) \quad \forall t \\ & x_t \geqslant 0,\ \ y_t(\xi) \geqslant 0. \end{aligned}$$

Multistage problems

Linear Decision Rules

- The decision $x_t(\xi)$ can depend on the *history* of the uncertainties $\xi_1, \ldots, \xi_t$;
- The LDR becomes $x_t(\xi) = X_{t,1} \cdot \xi_1 + X_{t,2} \cdot \xi_2 + \ldots + X_{t,t} \cdot \xi_t$.

Multistage problems

Linear Decision Rules

- The decision $x_t(\xi)$ can depend on the *history* of the uncertainties $\xi_1, \ldots, \xi_t$;
- The LDR becomes $x_t(\xi) = X_{t,1} \cdot \xi_1 + X_{t,2} \cdot \xi_2 + \ldots + X_{t,t} \cdot \xi_t$.
- Increases the number of decision variables for the LDR matrices.

Multistage problems

Linear Decision Rules

- The decision $x_t(\xi)$ can depend on the *history* of the uncertainties $\xi_1, \ldots, \xi_t$;
- The LDR becomes $x_t(\xi) = X_{t,1} \cdot \xi_1 + X_{t,2} \cdot \xi_2 + \ldots + X_{t,t} \cdot \xi_t$.
- Increases the number of decision variables for the LDR matrices.

SAA

- We discretize the uncertainty process into a finite number of scenarios, organized into a scenario tree.
- There is a value function V_n for each *node* of the tree.

Multistage problems

Stagewise independence

If the realization ξ_t at stage t is independent of the previous realizations $\xi_1, \ldots, \xi_{t-1}$, then the value functions depend *only on the stage t*.

- Efficient *dynamic programming* recursion, calculating $V_t(x_{t-1})$ from $V_{t+1}(x_t)$:

$$V_t(x_{t-1}) = \min_{\text{feasible } x_t, y_t} \mathbb{E}_{\xi_t} \left[c_t(\xi_t)^\top y_t(\xi_t) + V_{t+1}(x_t(\xi_t)) \right].$$

- One standard algorithm is *SDDP*, exploring the state space using Monte Carlo sampling of scenarios (and building cuts).

Our goals

- Build *time decompositions* suitable for linear decision rules;
- Recover *value functions* from such decompositions;
- Obtain a *reasonable policy* with moderate computational effort.

Back to the 2-stage problem

Notation: x is the incoming state, y the decision, and z the outgoing state:

$$\begin{aligned} V(x) = \quad \min \quad & \mathbb{E}\left[c(\xi)^\top y(x,\xi) + f(z(x,\xi))\right] \\ \text{s.t.} \quad & Tx + W_y y(x,\xi) + W_z z(x,\xi) = h(\xi) \\ & y(x,\xi), z(x,\xi) \geqslant 0. \end{aligned}$$

Back to the 2-stage problem

Notation: x is the incoming state, y the decision, and z the outgoing state:

$$\begin{aligned} V(x) = \quad \min \quad & \mathbb{E}\left[c(\xi)^\top y(x,\xi) + f(z(x,\xi))\right] \\ \text{s.t.} \quad & Tx + W_y y(x,\xi) + W_z z(x,\xi) = h(\xi) \\ & y(x,\xi), z(x,\xi) \geqslant 0. \end{aligned}$$

- Decisions y (and z) depend on the previous state x and the uncertainty ξ;

Back to the 2-stage problem

Notation: x is the incoming state, y the decision, and z the outgoing state:

$$\begin{aligned} V(x) = \quad \min \quad & \mathbb{E}\left[c(\xi)^\top y(x,\xi) + f(z(x,\xi))\right] \\ \text{s.t.} \quad & Tx + W_y y(x,\xi) + W_z z(x,\xi) = h(\xi) \\ & y(x,\xi), z(x,\xi) \geqslant 0. \end{aligned}$$

- Decisions y (and z) depend on the previous state x and the uncertainty ξ;
- The LDR for this problem sets $y(x,\xi) = Y_\xi(x) \cdot \xi$ and $z(x,\xi) = Z_\xi(x) \cdot \xi$;

Back to the 2-stage problem

Notation: x is the incoming state, y the decision, and z the outgoing state:

$$\begin{aligned} V(x) = \quad \min \quad & \mathbb{E}\left[c(\xi)^\top y(x,\xi) + f(z(x,\xi))\right] \\ \text{s.t.} \quad & Tx + W_y y(x,\xi) + W_z z(x,\xi) = h(\xi) \\ & y(x,\xi), z(x,\xi) \geqslant 0. \end{aligned}$$

- Decisions y (and z) depend on the previous state x and the uncertainty ξ;
- The LDR for this problem sets $y(x,\xi) = Y_\xi(x) \cdot \xi$ and $z(x,\xi) = Z_\xi(x) \cdot \xi$;
- The Linear Feedback Decision Rule (LFDR) sets $\begin{cases} y(x,\xi) = Y_\xi \cdot \xi + Y_x \cdot x & \text{and} \\ z(x,\xi) = Z_\xi \cdot \xi + Z_x \cdot x. \end{cases}$

Some observations

1. Several "value function recursions":

$$
\begin{aligned}
V(x) &= \mathbb{E}_\xi \left[c(\xi)^\top y^*(\xi) + f(z^*(\xi)) \right] \\
V_{LDR}(x) &= \mathbb{E}_\xi \left[c(\xi)^\top Y^*(x) \cdot \xi + f(Z^*(x) \cdot \xi) \right] \\
V_{LFDR}(x) &= \mathbb{E}_\xi \left[c(\xi)^\top (Y_\xi \cdot \xi + Y_x \cdot x) + f(Z_\xi \cdot \xi + Z_x \cdot x) \right].
\end{aligned}
$$

Some observations

1. Several “value function recursions”:

$$V(x) = \mathbb{E}_\xi \left[c(\xi)^\top y^*(\xi) + f(z^*(\xi))\right]$$
$$V_{LDR}(x) = \mathbb{E}_\xi \left[c(\xi)^\top Y^*(x) \cdot \xi + f(Z^*(x) \cdot \xi)\right]$$
$$V_{LFDR}(x) = \mathbb{E}_\xi \left[c(\xi)^\top (Y_\xi \cdot \xi + Y_x \cdot x) + f(Z_\xi \cdot \xi + Z_x \cdot x)\right].$$

2. For fixed x, the LFDR is a feasible LDR for both y and z, so $V_{LFDR}(x) \geqslant V_{LDR}(x) \geqslant V(x)$.

Some observations

1. Several "value function recursions":

$$V(x) = \mathbb{E}_\xi \left[c(\xi)^\top y^*(\xi) + f(z^*(\xi)) \right]$$
$$V_{LDR}(x) = \mathbb{E}_\xi \left[c(\xi)^\top Y^*(x) \cdot \xi + F^*(x) \cdot \xi \right]$$
$$V_{LFDR}(x) = \mathbb{E}_\xi \left[c(\xi)^\top (Y_\xi \cdot \xi + Y_x \cdot x) + F_\xi \cdot \xi + F_x \cdot x \right].$$

2. For fixed x, the LFDR is a feasible LDR for both y and z, so $V_{LFDR}(x) \geqslant V_{LDR}(x) \geqslant V(x)$.

3. We also need to represent $f(z)$ as linear functions...

Some observations

1. Several "value function recursions":

$$V(x) = \mathbb{E}_\xi \left[c(\xi)^\top y^*(\xi) + f(z^*(\xi)) \right]$$
$$V_{LDR}(x) = \mathbb{E}_\xi \left[c(\xi)^\top Y^*(x) \cdot \xi + F^*(x) \cdot \xi \right]$$
$$V_{LFDR}(x) = \mathbb{E}_\xi \left[c(\xi)^\top (Y_\xi \cdot \xi + Y_x \cdot x) + F_\xi \cdot \xi + F_x \cdot x \right].$$

2. For fixed x, the LFDR is a feasible LDR for both y and z, so $V_{LFDR}(x) \geqslant V_{LDR}(x) \geqslant V(x)$.
3. We also need to represent $f(z)$ as linear functions. . .
4. So $\frac{\partial V_{LFDR}(x)}{\partial x}$ is constant!

More general decision rules

- We *lift* x to a higher-dimensional space and only require that the decision rule is linear in the *lifted* variables;
- If $x = \sum_{i=1}^{J} x_i$, this leads to a piecewise-linear decision rule.

More general decision rules

- We *lift* x to a higher-dimensional space and only require that the decision rule is linear in the *lifted* variables;
- If $x = \sum_{i=1}^{J} x_i$, this leads to a piecewise-linear decision rule.
- We have $V_{PWLF}(x) = \mathbb{E}\left[c(\xi)^\top(\hat{Y}_\xi \cdot \xi + \hat{Y}_x \cdot L(x)) + \ldots\right]$.

More general decision rules

- We *lift* x to a higher-dimensional space and only require that the decision rule is linear in the *lifted* variables;
- If $x = \sum_{i=1}^{J} x_i$, this leads to a piecewise-linear decision rule.
- We have $V_{PWLF}(x) = \mathbb{E}\left[c(\xi)^\top(\hat{Y}_\xi \cdot \xi + \hat{Y}_x \cdot L(x)) + \ldots\right]$.
- $L(x)$ is a *piecewise-linear* function, so we might need to *convexify* V_{PWLF}.

More general decision rules

- We *lift* x to a higher-dimensional space and only require that the decision rule is linear in the *lifted* variables;
- If $x = \sum_{i=1}^{J} x_i$, this leads to a piecewise-linear decision rule.
- We have $V_{PWLF}(x) = \mathbb{E}\left[c(\xi)^\top(\hat{Y}_\xi \cdot \xi + \hat{Y}_x \cdot L(x)) + \dots\right]$.
- $L(x)$ is a *piecewise-linear* function, so we might need to *convexify* V_{PWLF}.

Theorem

The natural polyhedral representation, including the simplicial constraints

$$0 \leqslant x_J \leqslant \cdots \leqslant x_2 \leqslant x_1 \leqslant (\max x)/J$$

More general decision rules

- We *lift* x to a higher-dimensional space and only require that the decision rule is linear in the *lifted* variables;
- If $x = \sum_{i=1}^{J} x_i$, this leads to a piecewise-linear decision rule.
- We have $V_{PWLF}(x) = \mathbb{E}\left[c(\xi)^\top(\hat{Y}_\xi \cdot \xi + \hat{Y}_x \cdot L(x)) + \ldots\right]$.
- $L(x)$ is a *piecewise-linear* function, so we might need to *convexify* V_{PWLF}.

Theorem

The natural polyhedral representation, including the simplicial constraints

$$0 \leqslant x_J \leqslant \cdots \leqslant x_2 \leqslant x_1 \leqslant (\max x)/J$$

yields $\begin{cases} \text{a convexification of } V_{PWLF}; \\ \end{cases}$

More general decision rules

- We *lift* x to a higher-dimensional space and only require that the decision rule is linear in the *lifted* variables;
- If $x = \sum_{i=1}^{J} x_i$, this leads to a piecewise-linear decision rule.
- We have $V_{PWLF}(x) = \mathbb{E}\left[c(\xi)^\top(\hat{Y}_\xi \cdot \xi + \hat{Y}_x \cdot L(x)) + \dots\right]$.
- $L(x)$ is a *piecewise-linear* function, so we might need to *convexify* V_{PWLF}.

Theorem

The natural polyhedral representation, including the simplicial constraints

$$0 \leqslant x_J \leqslant \cdots \leqslant x_2 \leqslant x_1 \leqslant (\max x)/J$$

yields $\begin{cases} \text{a convexification of } V_{PWLF}; \\ \text{which remains a valid upper bound for } V_{LDR}(x), \text{ so for } V(x). \end{cases}$

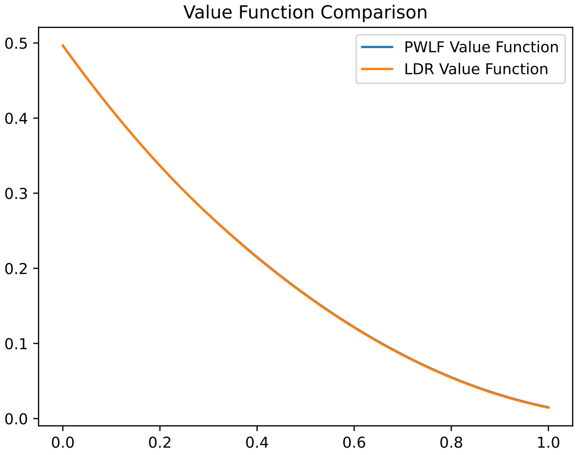
A toy example

```
using JuMP, LinearDecisionRules
using Ipopt, Distributions

demand = 0.3

m = LDRModel(Ipopt.Optimizer)
@variable(m, vi in Uncertainty(distribution=Uniform(0,1)))
@variable(m, inflow in Uncertainty(distribution=Uniform(0,0.2)))
@variable(m, 0 <= vf <= 1)
@variable(m, gh >= 0.0)
@variable(m, gt >= 0.0)

@constraint(m, balance, vf == vi - gh + inflow)
@constraint(m, gt + gh == demand)

@objective(m, Min, gt^2 + vf^2/2 - vf + 0.5)
```

A toy example (cont.)

```
# Solve the primal LDR
set_attribute(m, SolvePrimal(), true)
set_attribute(m, SolveDual(), false)
set_attribute(vi, BreakPoints(), 5)
optimize!(m)

# Get the value function
VF = JuMP.Model()
@variable(VF, x)
@objective(VF, Min, 0.0)
set_parametric_objective!(VF, m, Dict(vi => x))
```

A toy example

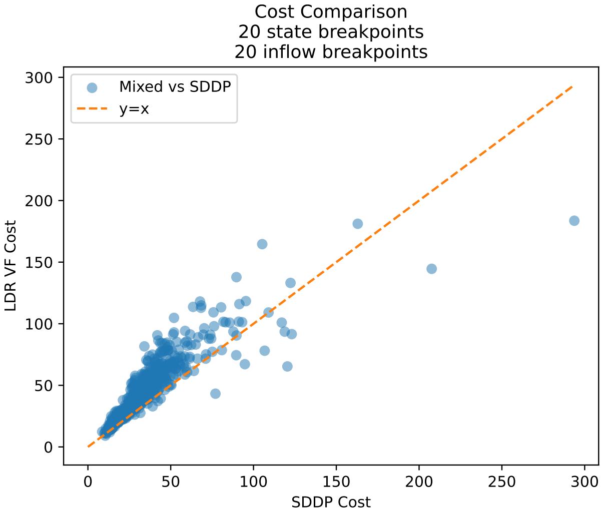


Slightly more complex example

- Still one-dimensional;
- 24 stages, several thermal plants;
- Triangular uncertainty distribution;
- 20 breakpoints for the decision rules.

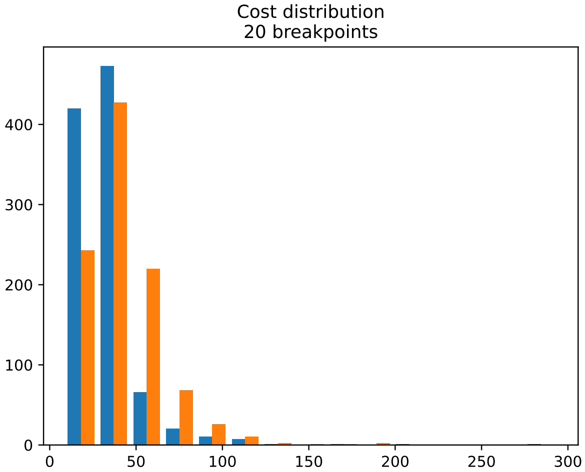
Slightly more complex example

- Still one-dimensional;
- 24 stages, several thermal plants;
- Triangular uncertainty distribution;
- 20 breakpoints for the decision rules.

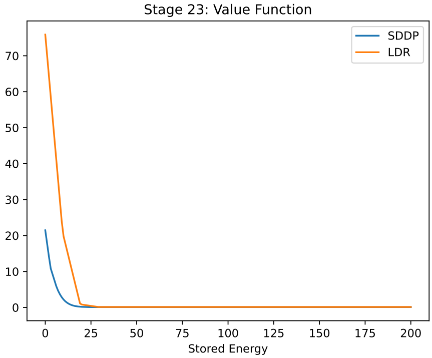


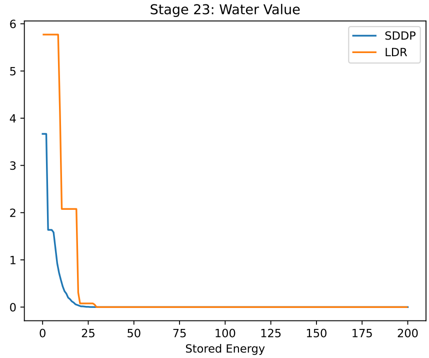
Slightly more complex example

- Still one-dimensional;
- 24 stages, several thermal plants;
- Triangular uncertainty distribution;
- 20 breakpoints for the decision rules.

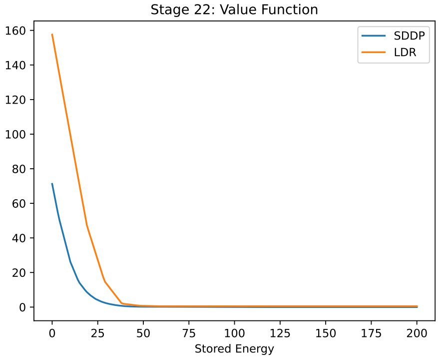


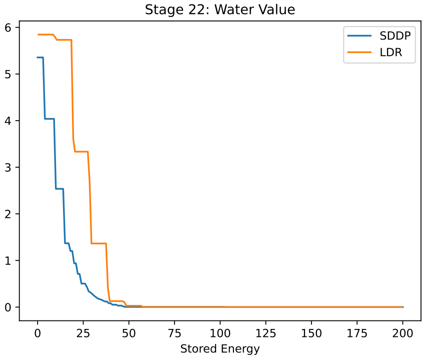
Slightly more complex example



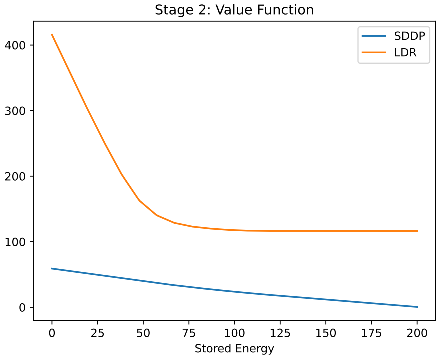


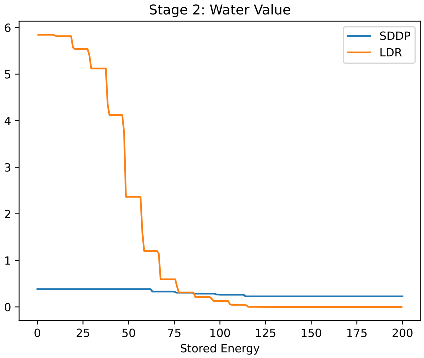
Slightly more complex example





Slightly more complex example





Next steps

Multistage decision rules:

- Generalize `FirstStage` to accommodate decisions x_t which can only depend on *observed* uncertainties $\xi_1, \ldots, \xi_t$;
- Will benefit from correlated uncertainties to model more complex processes.

Next steps

Multistage decision rules:

- Generalize `FirstStage` to accommodate decisions x_t which can only depend on *observed* uncertainties $\xi_1, \ldots, \xi_t$;
- Will benefit from correlated uncertainties to model more complex processes.

Performance:

- Speed-up *model building* for larger problems (ongoing);
- *Auto-tune breakpoints* (number and position);
- Adaptive *state* distribution.

GitHub Package

Documentation

Questions?