Simulation Methods for Stochastic Storage Problems: A Statistical Learning Perspective

Aditya Maheshwari

Joint work with: Dr. Michael Ludkovski
Department of Statistics and Applied Probability
University of California, Santa Barbara

9th Western Conference on Mathematical Finance

Outline

Stochastic Storage: An optimal switching problem

- A storage problem has two key components:
  - Stochastic Risk factors (autonomous dynamics).
  - An inventory variable (fully controlled).
- Dynamics of the inventory is controlled via storage regimes (typically finite, > 2).
- In comparison, optimal stopping (American option pricing) has only two regimes: $\{\text{continue}, \text{stop}\}$ and stochastic factors.

Stochastic Storage: An optimal switching problem

- A storage problem has two key components:
  - Stochastic Risk factors (autonomous dynamics).
  - An inventory variable (fully controlled).
- Dynamics of the inventory is controlled via storage regimes (typically finite, > 2).
- In comparison, optimal stopping (American option pricing) has only two regimes: $\{\text{continue}, \text{stop}\}$ and stochastic factors.

In today's talk we will discuss:

- A new modular algorithm (and its features) for optimal switching, combining Design of Experiments with Machine learning.
- Gaussian Process as a "basis-free" method for approximating continuation value.

Optimal switching problems

Gas storage valuation: A classic example

- Gas prices as stochastic factor with dynamics $P_{t_{k+1}} = P_{t_k} + b(t_k, P_{t_k})\Delta t + \sigma(t_k, P_{t_k})\,\Delta W_{t_k}$.
- Inventory variable: $I_{t_{k+1}} = I_{t_k} + a(c_{t_k})\,\Delta t$, $I_{t_k} \in [0, I_{\max}]$.
- Control c_{t_k} represents rate of injection/withdrawal or holding, fully determined by storage regime $m_{t_{k+1}} \in \{\text{inject}, \text{withdraw}, \text{hold}\}$.
- Objective: Create a policy $m_{t_{k+1}} = \mathcal{M}(t_k, P_{t_k}, I_{t_k}, m_{t_k})$ which maximizes the expected profit $v(t_0, \mathbf{P}_{t_0}, I_{t_0}, \mathbf{m}_{t_0})$ in the horizon $[0, T]$.

Optimal switching problems

Gas storage valuation: A classic example

- Gas prices as stochastic factor with dynamics $P_{t_{k+1}} = P_{t_k} + b(t_k, P_{t_k})\Delta t + \sigma(t_k, P_{t_k})\,\Delta W_{t_k}$.
- Inventory variable: $I_{t_{k+1}} = I_{t_k} + a(c_{t_k})\,\Delta t,\ I_{t_k} \in [0, I_{\max}]$.
- Control c_{t_k} represents rate of injection/withdrawal or holding, fully determined by storage regime $m_{t_{k+1}} \in \{\text{inject}, \text{withdraw}, \text{hold}\}$.
- Objective: Create a policy $m_{t_{k+1}} = \mathcal{M}(t_k, P_{t_k}, I_{t_k}, m_{t_k})$ which maximizes the expected profit $v(t_0, \mathbf{P}_{t_0}, I_{t_0}, \mathbf{m}_{t_0})$ in the horizon $[0, T]$.

Microgrid control

- Electric load and renewable output as stochastic factors.
- Battery storage system as inventory, controlled via backup diesel generator.
- Objective: Create a policy to match demand and supply of electricity and maximize the negative expected cost.

Value Function and DPE for Gas storage

- Value Function $V(t_k, P_{t_k}, I_{t_k}, m_{t_k}) =$

$$\sup_{\mathbf{m}_{t_k}} \mathbb{E}\left[\sum_{s=k}^{K-1} e^{-r(t_s - t_k)} \pi^{\Delta}(P_{t_s}, m_{t_s}, m_{t_{s+1}}) + e^{-r(T-t_k)} W(P_T, I_T) \Big| \, P_{t_k}, I_{t_k}, m_{t_k} \right],$$

where π^{Δ} incorporates the revenue and the switching cost.

Value Function and DPE for Gas storage

- Value Function $V(t_k, P_{t_k}, I_{t_k}, m_{t_k}) =$

$$\sup_{\mathbf{m}_{t_k}} \mathbb{E}\left[\sum_{s=k}^{K-1} e^{-r(t_s-t_k)}\pi^{\Delta}(P_{t_s}, m_{t_s}, m_{t_{s+1}}) + e^{-r(T-t_k)}W(P_T, I_T)\Big|\ P_{t_k}, I_{t_k}, m_{t_k}\right],$$

  where π^{Δ} incorporates the revenue and the switching cost.

- Dynamic Programming Equation

$$V(t_k, P_{t_k}, I_{t_k}, m_{t_k}) = \max_{m\in\mathcal{J}} \mathbb{E}\left[\pi^{\Delta}(P_{t_k}, m_{t_k}, m) + e^{-r\Delta t}V(t_{k+1}, P_{t_{k+1}}, I_{t_{k+1}}, m)\Big|P_{t_k}\right]$$

Value Function and DPE for Gas storage

- Value Function $V(t_k, P_{t_k}, I_{t_k}, m_{t_k}) =$

$$\sup_{\mathbf{m}_{t_k}} \mathbb{E}\left[\sum_{s=k}^{K-1} e^{-r(t_s-t_k)}\pi^{\Delta}(P_{t_s}, m_{t_s}, m_{t_{s+1}}) + e^{-r(T-t_k)}W(P_T, I_T)\middle|\, P_{t_k}, I_{t_k}, m_{t_k}\right],$$

  where π^{Δ} incorporates the revenue and the switching cost.

- Dynamic Programming Equation

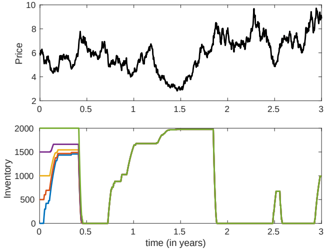
$$V(t_k, P_{t_k}, I_{t_k}, m_{t_k}) = \max_{m\in\mathcal{J}} \mathbb{E}\left[\pi^{\Delta}(P_{t_k}, m_{t_k}, m) + e^{-r\Delta t}V(t_{k+1}, P_{t_{k+1}}, I_{t_{k+1}}, m)\middle| P_{t_k}\right]$$

- Optimal Control

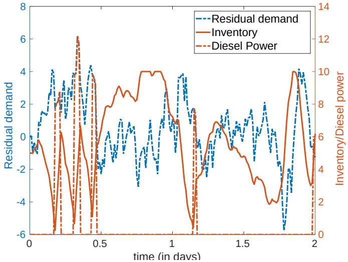
$$m^*(t_k, P, I, m) = \arg\max_{j\in\mathcal{J}} \left\{\pi^{\Delta}(P, m, j) + e^{-r\Delta t}q(t_k, P, I + a(c_{t_k}(j))\Delta t, j)\right\},$$

  where Continuation Value $q(t_k, P, I, m) := \mathbb{E}\left[V(t_{k+1}, P_{t_{k+1}}, I, m)\middle|\, P_{t_k} = P\right]$

A picture is worth a thousand words . . .



Gas Storage



Microgrid

A pathwise trajectory for two examples

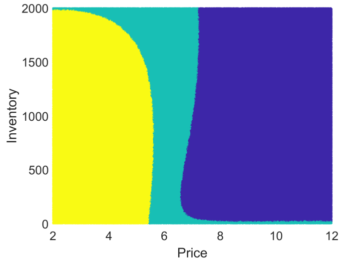
Gas Storage

- Buy low and sell high.

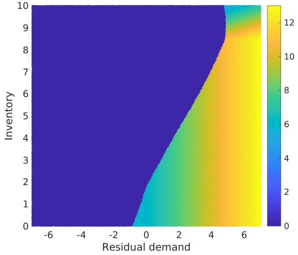
Microgrid

- Switch ON/OFF the generator to match demand and charge battery.

Control Maps



Gas Storage $m^*(P, I, +1)$



Microgrid $c^*(X, I, +1)$

Optimal Policy

How to efficiently estimate the maps $m^*(P, I, m)$?

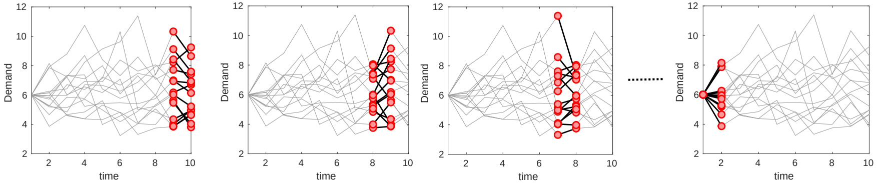
Contributions

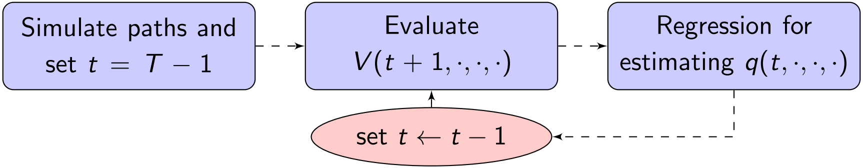
- A new algorithm Dynamic Emulation Algorithm(DEA)
  - Flexible modular template by reformulating optimal switching problem as recursive statistical learning.
  - Memory efficient and Scalable via "smart" simulation/regressions.
- Several new simulation designs which are dynamic, explores the space and exploits the distributional content.
- Gaussian Process Regression: a non-parametric tool for approximating the continuation function $q(t_k, \cdot, \cdot, m)$

Outline

1. Stochastic storage
2. Dynamic Emulation Algorithm
3. Numerical Illustrations
4. Conclusion

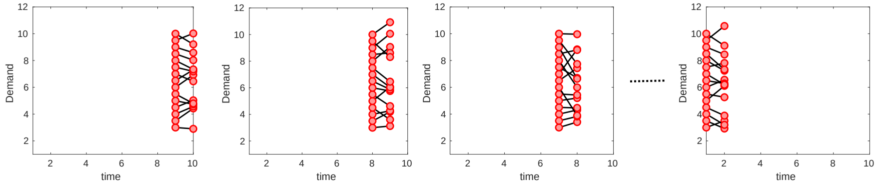
Traditional Implementation

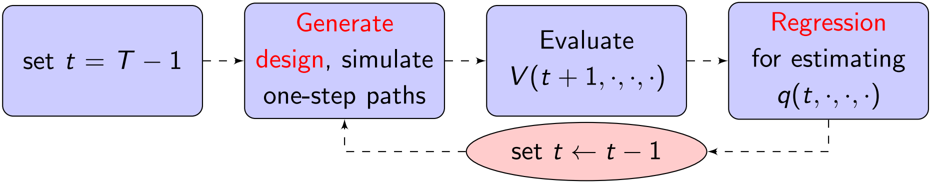




- Paths for the exogenous process is simulated and stored for [0,T].
- Controlled trajectory I_t cannot be simulated. So, replicate the exogenous process P_t/X_t for all possible values of I_t.
- Non-parametric regression methods are very slow since we need $\approx 10^4$ paths for reasonable estimation.

Dynamic Emulation Algorithm (DEA)

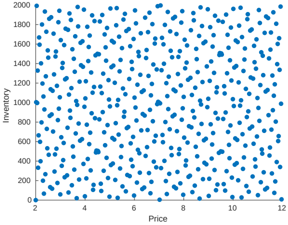




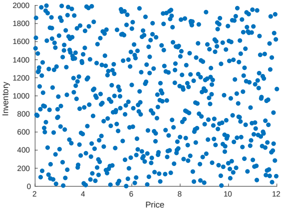
- Stochastic simulation for sequential learning of q.
- Both design and regression affects the quality of the solution.
- "Smart" design ☛ Improved learning.

Design

- Space Filling Design
  - Sobol QMC Sequences: Theoretical guarantees on “uniform” space filling.
  - Latin Hypercube Sampling (LHS): Probabilistic counterpart of QMC sequences.



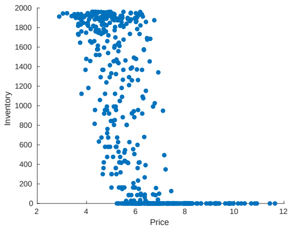
2D Sobol QMC sequence



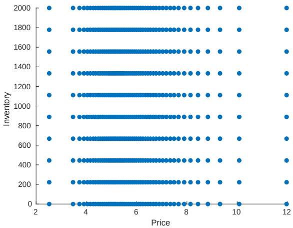
2D LHS design

Design

- Probabilistic Design: Mimics the distribution of (P, I). Too targeted, fails to explore the space.
- Gridded Design: Probabilistic in P and equally spaced in I



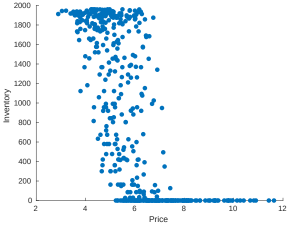
Joint Probabilistic design



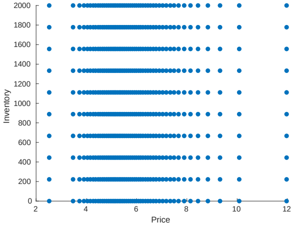
Conventional product design

Design

- Probabilistic Design: Mimics the distribution of (P, I). Too targeted, fails to explore the space.
- Gridded Design: Probabilistic in P and equally spaced in I



Joint Probabilistic design



Conventional product design

- Mixture Design: Combination of space filling and probabilistic designs (exploration-exploitation tradeoff).

Stochastic Simulator (at time t_k)

- Choose the design $\mathcal{D}_k := (P^n_{t_k}, I^n_{t_{k+1}}, m^n_{t_{k+1}}, n = 1, \ldots, N_k)$.
- Generate one-step paths $P^n_{t_k} \mapsto P^n_{t_{k+1}}$.
- Calculate one-step-ahead pathwise profits:

$$v^n_{k+1} := \max_{j \in \mathcal{J}} \left\{ \pi^{\Delta}(P^n_{t_{k+1}}, m^n_{t_{k+1}}, j) + e^{-r\Delta t} \hat{q}(t_{k+1}, P^n_{t_{k+1}}, I^n_{t_{k+1}} + a(c(j))\Delta t, j) \right\}$$

Stochastic Simulator (at time t_k)

- Choose the design $\mathcal{D}_k := (P^n_{t_k}, I^n_{t_{k+1}}, m^n_{t_{k+1}}, n = 1, \dots, N_k)$.
- Generate one-step paths $P^n_{t_k} \mapsto P^n_{t_{k+1}}$.
- Calculate one-step-ahead pathwise profits:

$$v^n_{k+1} := \max_{j \in \mathcal{J}} \left\{ \pi^{\Delta}(P^n_{t_{k+1}}, m^n_{t_{k+1}}, j) + e^{-r\Delta t} \hat{q}(t_{k+1}, P^n_{t_{k+1}}, I^n_{t_{k+1}} + a(c(j))\Delta t, j) \right\}$$

- Learn the input-output relationship between $(P_{t_k}, I_{t_{k+1}}, m_{t_{k+1}})_{n=1}^{N_k}$ and $(v_{k+1})_{n=1}^{N_k}$ using L^2 projection on an approximation space $\mathcal{H}_k$.

$$\hat{q}(t_k, \cdot, \cdot, \cdot) = \arg\min_{h_{t_k} \in \mathcal{H}_k} \sum_{n=1}^{N} \left| h_{t_k}(P^n_{t_k}, I^n_{t_{k+1}}, m^n_{t_{k+1}}) - v^n_{k+1} \right|^2.$$

Example: $\mathcal{H}_k = \mathrm{span}(\phi_1, \dots, \phi_R)$ and $h_{t_k}(\cdot) = \sum_{i=1}^{R} \beta_i \phi_i(\cdot)$

- We use Gaussian Process as a new alternate for $\mathcal{H}$.

Gaussian Process Regression (GPR)

- Assuming that the input-output relationship as

$$y^n = h(x^n) + \sigma^2 \xi,$$

where $\xi \sim \mathcal{N}(0,1)$, $x^n = (P^n_{t_k}, I^n_{t_{k+1}})$ and $y^n = v^n_{k+1}$.
- We assume $h(\cdot)$ is a realization of Gaussian random field.
- $\{h(x_i)\}_{i=1}^N$ is a sample from the multivariate normal distribution (MVND) with mean $\{m(x_i)\}_{i=1}^N$ and covariance $\{\kappa(x_i, x_j)\}_{i,j=1}^N$.
- Conditional distribution at new sites $\mathbf{x}_*$ is also multivariate normal i.e. $h(\mathbf{x}_*)|(x_i, y_i)_{i=1}^N \sim MVND(m_*(\mathbf{x}_*), \kappa_*(\mathbf{x}_*, \mathbf{x}_*))$

Gaussian Process Regression (GPR)

- Assuming that the input-output relationship as

$$y^n = h(x^n) + \sigma^2 \xi,$$

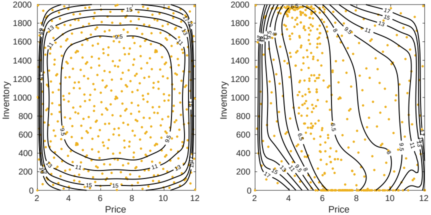
where $\xi \sim \mathcal{N}(0,1)$, $x^n = (P^n_{t_k}, I^n_{t_{k+1}})$ and $y^n = v^n_{k+1}$.
- We assume $h(\cdot)$ is a realization of Gaussian random field.
- $\{h(x_i)\}_{i=1}^N$ is a sample from the multivariate normal distribution (MVND) with mean $\{m(x_i)\}_{i=1}^N$ and covariance $\{\kappa(x_i, x_j)\}_{i,j=1}^N$.
- Conditional distribution at new sites $\mathbf{x}_*$ is also multivariate normal i.e. $h(\mathbf{x}_*)|(x_i, y_i)_{i=1}^N \sim MVND(m_*(\mathbf{x}_*), \kappa_*(\mathbf{x}_*, \mathbf{x}_*))$

Advantages of using GPR:

- Analytic tractability for a non-parametric method.
- Smooth approximation for the function and even its derivative.
- Standard error on estimates at any new location x.

Gaussian Process Regression

- Rectangular level sets for the standard error due to space filling design (Sobol QMC).
- Mixture design leads to lower standard error in the switching region.



Impact of design on posterior standard error. Left: Sobol QMC, Right: Mixture design

Allows for sequential design (Gramacy and Ludkovski (2015), Ludkovski (2018) and Hu and Ludkovski (2016)).

Outline

Example1: Gas Storage

Design	Regression Scheme	Simulation Budget Low	Medium	Large
Conventional	PR-1D	4,965	5,097	5,231
	GP-1D	4,968	5,107	5,247
Adaptive 1D	PR-1D	5,061	5,187	5,246
	GP-1D	5,079	5,195	5,245
Dynamic	GP-1D	5,132	5,225	5,266
	Mixed	5,137	5,205	5,228
Mixture 2D	PR-2D	4,820	4,835	4,834
	GP-2D	5,137	5,210	5,233

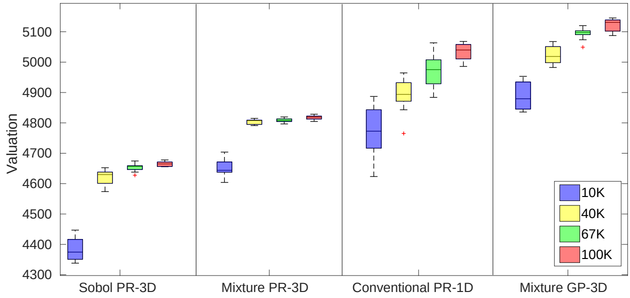
Valuation $\hat{V}(0, 6, 1000)$ (in thousands) using different design-regression pairs and three simulation budgets: Low $N \simeq 10K$, Medium $N \simeq 40K$, Large $N \simeq 100K$. PR: Polynomial regression, GP: Gaussian process

- Dynamic design: Varying budget, particularly higher simulation budget at the boundary due to penalty.
- Dynamic design + Mixed Regression: Varying budget and regression scheme as we move through the backward induction.
- Mixture design: 60% probabilistic and 40% space filling design.

☛ Higher valuation with mixture and dynamic design compared to conventional methods.

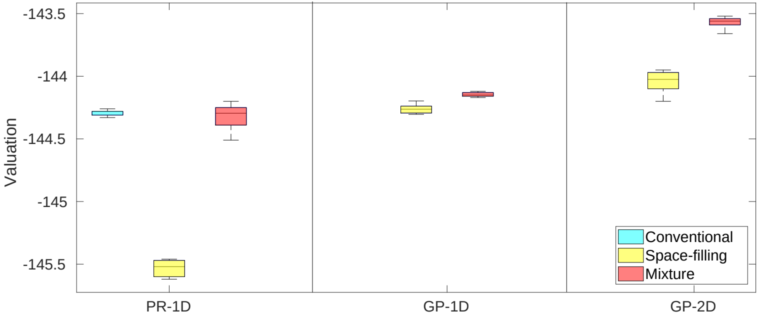
Example2: Double Gas Storage

Jointly estimate the valuation of two independent gas storage facilities. State space (P_t, I_t^1, I_t^2), where I_t^j is the inventory for j^{th} facility.



☛ Space Filling design < Conventional design < Mixture design.
☛ Gaussian process better than the state-of-the-art 1D regressions.
☛ Parametric 1D regressions significantly outperform parametric -3D irrespective of the design.

Example3: Microgrid



☛ Poor performance of space filling designs across regression methods, GP more robust than the state-of-art.

☛ GP-2D with mixture design substantially better than Parametric -1D.

Conclusion

So far . . .

- We developed a new Dynamic Emulation Algorithm:
  - Scalable (very low memory requirement).
  - Flexible (vary budget/regression/design distribution across time-steps).
  - Fully controlled processes can be handled with ease (work in progress)
- Non-parametric regression methods for learning the value function.
- Emphasis on "smart" design through several examples exhibiting dynamic, exploratory and exploitative aspects of the algorithm.
- See `https://arxiv.org/abs/1803.11309` for Microgrid example and other details.

Conclusion

So far . . .

- We developed a new Dynamic Emulation Algorithm:
  - Scalable (very low memory requirement).
  - Flexible (vary budget/regression/design distribution across time-steps).
  - Fully controlled processes can be handled with ease (work in progress)
- Non-parametric regression methods for learning the value function.
- Emphasis on "smart" design through several examples exhibiting dynamic, exploratory and exploitative aspects of the algorithm.
- See `https://arxiv.org/abs/1803.11309` for Microgrid example and other details.

DEA =

Conclusion

So far ...

- We developed a new Dynamic Emulation Algorithm:
  - Scalable (very low memory requirement).
  - Flexible (vary budget/regression/design distribution across time-steps).
  - Fully controlled processes can be handled with ease (work in progress)
- Non-parametric regression methods for learning the value function.
- Emphasis on "smart" design through several examples exhibiting dynamic, exploratory and exploitative aspects of the algorithm.
- See `https://arxiv.org/abs/1803.11309` for Microgrid example and other details.

DEA = Modular template + "Smart" Design + "Smart" Regression

Conclusion

So far . . .

- We developed a new Dynamic Emulation Algorithm:
  - Scalable (very low memory requirement).
  - Flexible (vary budget/regression/design distribution across time-steps).
  - Fully controlled processes can be handled with ease (work in progress)
- Non-parametric regression methods for learning the value function.
- Emphasis on "smart" design through several examples exhibiting dynamic, exploratory and exploitative aspects of the algorithm.
- See `https://arxiv.org/abs/1803.11309` for Microgrid example and other details.

DEA = Modular template + "Smart" Design + "Smart" Regression

Work in progress

- We extend Dynamic Emulation Algorithm for stochastic optimal control with probabilistic constraints.

References

Balata, A., Palczewski, J.: Regress-Later Monte Carlo for Optimal Inventory Control with applications in energy (2017)

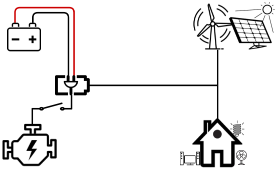
Heymann, B, Bonnans, J.F, Martison, P, Silva, F, Lanas, F, Jimenez, G: Continuous optimal control approaches to microgrid energy management (2015)

Heymann, B, Bonnans, J.F, Silva, F, Jimenez, G: A Stochastic Continuous Time Model for Microgrid Energy Management (2016)

Ludkovski, M. Kriging Metamodels and Experimental Design for Bermudan Option Pricing, Journal of Computational Finance (2018)

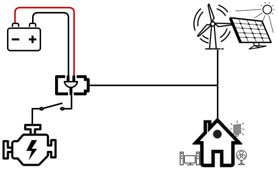
Gevret, H., Langrené, N., Lelong, J., and Warin, X., and Maheshwari, A., Stochastic optimization library in c++ (2018).

Microgrid Management



- Net demand (Load - Renewables) as stochastic factor with dynamics $X_{t_{k+1}} - X_{t_k} = \alpha(\underline{X} - X_{t_k})\Delta t + \sigma \Delta W_{t_k}$.
- Inventory variable: $I_{t_{k+1}} = I_{t_k} + a(c_{t_k})\Delta t = I_{t_k} + B_{t_k}\Delta t,\ I_{t_k} \in [0, I_{\max}]$.
- Control of diesel generator $c_{t_k}(1) = X_{t_k}\mathbf{1}_{\{X_{t_k}>0\}} + B_{\max} \wedge \frac{I_{\max} - I_{t_k}}{\Delta t}$.
- Regime $m_{t_{k+1}} \in \{\text{Generator ON}, \text{Generator OFF}\}$.

Microgrid Management



- Net demand (Load - Renewables) as stochastic factor with dynamics $X_{t_{k+1}} - X_{t_k} = \alpha(\underline{X} - X_{t_k})\Delta t + \sigma \Delta W_{t_k}$.
- Inventory variable: $I_{t_{k+1}} = I_{t_k} + a(c_{t_k})\Delta t = I_{t_k} + B_{t_k}\Delta t$, $I_{t_k} \in [0, I_{\max}]$.
- Control of diesel generator $c_{t_k}(1) = X_{t_k}\mathbf{1}_{\{X_{t_k}>0\}} + B_{\max} \wedge \frac{I_{\max} - I_{t_k}}{\Delta t}$.
- Regime $m_{t_{k+1}} \in \{\text{Generator ON}, \text{Generator OFF}\}$.
- Battery $B_{t_k} := a(c_{t_k}) = -\frac{I_{t_k}}{\Delta t} \vee \left(B_{\min} \vee (c_{t_k} - X_{t_k}) \wedge B_{\max}\right) \wedge \frac{I_{\max} - I_{t_k}}{\Delta t}$.
- One step profit $\pi(c, X) := -c^{\gamma} - |S|\Big[C_2\mathbf{1}_{\{S<0\}} + C_1\mathbf{1}_{\{S>0\}}\Big]$, where $S_{t_k} = c_{t_k} - X_{t_k} - B_{t_k}$.
  - $S_{t_k} < 0$ leads to blackout; $S_{t_k} > 0$ waste of energy.

Example1: Gas Storage

Design	Regression Scheme	Simulation Budget Low	Medium	Large
Conventional	PR-1D	4,965	5,097	5,231
	GP-1D	4,968	5,107	5,247
	PR-2D	4,869	4,888	4,891
	LOESS-2D	4,910	4,969	5,011
	GP-2D	4,652	5,161	5,243
Space-filling	PR-1D	4,768	4,889	5,028
	GP-1D	4,854	5,064	5,224
	PR-2D	4,762	4,789	4,792
	LOESS-2D	4,747	4,912	4,934
	GP-2D	4,976	5,080	5,133
Adaptive 1D	PR-1D	5,061	5,187	5,246
	GP-1D	5,079	5,195	5,245
Dynamic	GP-1D	5,132	5,225	5,266
	Mixed	5,137	5,205	5,228
Mixture 2D	PR-2D	4,820	4,835	4,834
	LOESS-2D	4,960	4,987	5,003
	GP-2D	5,137	5,210	5,233

Valuation $\hat{V}(0, 6, 1000)$ (in thousands) using different design-regression pairs and three simulation budgets: Low $N \simeq 10K$, Medium $N \simeq 40K$, Large $N \simeq 100K$. The valuations are averages across 10 runs of each scheme except for LOESS-2D with large budget: due to the excessive overhead of LOESS only a single run was carried out.