Simulation Methods for Stochastic Storage Problems: A Statistical Learning Perspective

Mike Ludkovski

with Aditya Maheshwari (UCSB)

Caesars2018 Conference, EDF, Paris, September 2018

UC Santa Barbara

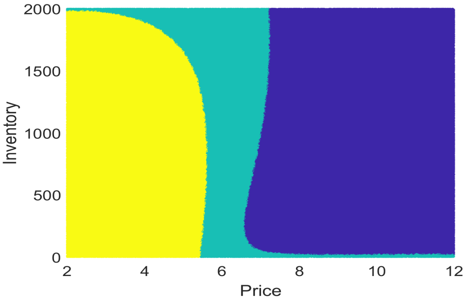
STATISTICS & APPLIED PROBABILITY CENTER FOR FINANCIAL MATHEMATICS & ACTUARIAL RESEARCH

Work Supported by NSF AMPS-1736439

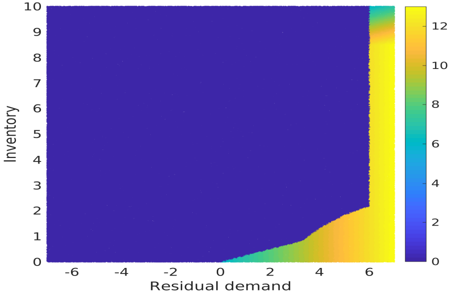
Storage Problems

- Managing inventory in the presence of risk
- Sequentially decide when to augment/draw-down inventory given a randomly evolving environment
- Stochastic Switching Control
- **Natural Gas Storage** facility (underground caverns):
  - Maximize revenue: buy low/sell high
  - Widely used by market participants through bespoke tolling/lease agreements
- **Microgrid Battery Management**:
  - Renewable generation facility (solar or wind) leading to stochastic net demand
  - Diesel generator for back-up
  - Minimize costs: optimize battery operations
  - Avoid blackouts/minimize diesel costs

Output: Feedback Control



(a) $m = +1$ (Inject)



(b) Generator OFF ($m = 0$)

Figure: *Left panel:* action to undertake when injecting into a **gas storage** facility as a function of gas price and current inventory: inject, store, withdraw. *Right panel:* action to undertake when **diesel generator** is off, as a function of current residual demand and inventory: OFF, ON (variable level).

We are interested in efficient ways of generating these maps.

Storage Problems as Control

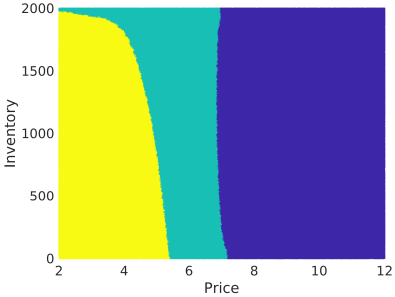
- Abstractly: stochastic control of **switching-type**
- Stochastic factor P_t (price, demand, etc)
- Inventory variable I_t
- Controlled regime $m_t \in \mathcal{M}$; switching costs $K(m, m')$
- Typically [not required]: (P_t) is exogenous (follows an SDE); (I_t) is fully endogenous, determined by (m_t) (ODE)
- Objective and cost include all of (P_t, I_t, m_t) and subject to inventory constraints $I_t \in [I_{\min}, I_{\max}]$
- View m_t as a persistent regime that is part of the system state and drives system dynamics [dependence vanishes if switching costs are zero]
- Discrete-time formulation

Natural Gas Storage

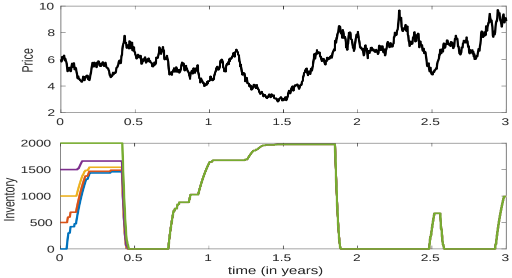
- Gas price P_t – stochastic process
- Action space: $\mathcal{M} = \{+1, 0, -1\}$: inject, store, withdraw
- Leads to control $c_t(m, I)$ (cavern pressure) and inventory impact $dI_t = a(c_t)dt$
- Capacity constraint $I_t \in [0, I_{\max}]$
- Discrete/ switching control space: pick the best action from $\mathcal{M}$
- Notation: $m_{t_{k+1}}$ is the regime on $[t_k, t_{k+1}]$, determines $I_{t_{k+1}}$ (previsible)

Solution Structure

- Controlled inventory: $\hat{I}_{t_{k+1}} = \hat{I}_{t_k} + a\big(c_{t_k}(\hat{m}_{t_{k+1}}(t, P, I, \hat{m}_{t_k}))\big)\Delta t$
- If P_t is high – inject; if low : withdraw.
- Key output is the control **map** from which recursively read-off $m^*(t, P_t, I_t, m_t)$.



(a) Control map



(b) Pathwise control/inventory

Figure: *Left panel:* snapshot of the control map $\hat{m}(t, P, I)$ at $t = 2.7$ years. *Top right:* a given trajectory of commodity price (P_t) following logarithmic mean reverting dynamics. *Lower right:* Corresponding trajectories of controlled inventory $\hat{I}_t$ starting at $\hat{I}_0 \in \{0, 500, 1000, 2000\}$. Obtained from the PR-1D solution scheme.

Stochastic Control Formulation

- Value function

$$V(t_k, P, I, m) = \sup_{\mathbf{m}_{t_k}} \mathbb{E}\left[v(t_k, \mathbf{P}_{t_k}, I_{t_k}, \mathbf{m}_{t_k}) \Big| \, P_{t_k} = P, I_{t_k} = I, m_{t_k} = m \right]$$

- Payoff is $v(t_k, \mathbf{P}_{t_k}, I_{t_k}, \mathbf{m}_{t_k}) := \sum_{s=k}^{K-1} e^{-r(t_s - t_k)} [\pi(P_{t_s}, c_{t_k}(m_{t_{s+1}}))\Delta t - K(m_{t_s}, m_{t_{s+1}})] + e^{-r(T-t_k)} W(P_T, I_T)$
- Continuation value: $q(t_k, P, I, m) := \mathbb{E}\left[e^{-r\Delta t} V(t_{k+1}, P_{t_{k+1}}, I, m) \Big| \, P_{t_k} = P \right]$.
- Dynamic Programming equation

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

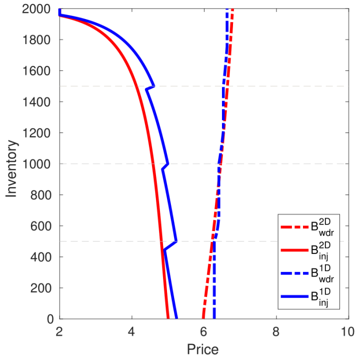
- Optimal control $m^*_{t_{k+1}}(t_k, P_{t_k}, I_{t_k}, m_{t_k})$ is the argmax above.

Existing Solution Methods

- PDE-based (Chen and Forsyth '08, Thompson '16) – degenerate in I-coordinate; limited to 1D factor models.
- **Regression Monte Carlo:** Carmona-L ('10), Denault et al ('13), Boogert and de Jong ('08, '11), Warin ('10), Bauerle and Riess ('16), Malyscheff and Trafalis ('17), Balata and Palczewski ('17),....
- Classic RMC: generate forward paths, use the resulting stochastic mesh to solve the DPE, employ cross-sectional regression for the conditional expectation. Construct pathwise rewards.
- Key challenge for applying RMC is how to handle the endogenous I_t – cannot do global path simulation
  - Inventory path back-propagation and quasi-simulation
  - Treat I_t as a parameter, solve a collection of 1-D problems in P_t
  - Control randomization

Regression: The -1D Discretization Trick

- Reduce to a finite number of 1D switching problems by discretizing the inventory I, solved in parallel
- $M_I + 1$ levels $I_0, I_1, \ldots, I_{M_I}$, fit in P for each level, i.e. optimize for $\hat{h}_j(P) := \vec{\beta}_j^{\,T} \vec{\phi}(P)$ for $j = 0, \ldots, M_I$
- Interpolate:
  $\widehat{h}_{t_k}(P, I) := \delta(I)\ \hat{h}_j(P) + (1 - \delta(I))\ \hat{h}_{j+1}(P)$,
  where $\delta(I) = \frac{I - I_j}{I_{j+1} - I_j}$.
- Makes the policy smooth in P but not in I; the discretization grid (I_m) plays a big role
- Works very well but limited scope



RMC for Storage: Machine Learning Perspective

- Abstract away the conventional RMC particulars
- The storage model is viewed as stochastic simulator(s) that produces noisy pathwise observations conditional on the control
- Learn the continuation or q-value $q(t, P, I, m)$: cost-to-go conditional on next-step regime m; done recursively over t
- Sub-modules [each has multiple viable schemes]:

;

- Put another way:

RMC for Storage: Machine Learning Perspective

- Abstract away the conventional RMC particulars
- The storage model is viewed as stochastic simulator(s) that produces noisy pathwise observations conditional on the control
- Learn the continuation or q-value $q(t, P, I, m)$: cost-to-go conditional on next-step regime m; done recursively over t
- Sub-modules [each has multiple viable schemes]:
  - Approximation of the conditional expectation defining q;
- Put another way:

RMC for Storage: Machine Learning Perspective

- Abstract away the conventional RMC particulars
- The storage model is viewed as stochastic simulator(s) that produces noisy pathwise observations conditional on the control
- Learn the continuation or q-value $q(t, P, I, m)$: cost-to-go conditional on next-step regime m; done recursively over t
- Sub-modules [each has multiple viable schemes]:
  - Approximation of the conditional expectation defining q;
  - Evaluation of the optimal control [trivial if m is discrete]
- Put another way:

RMC for Storage: Machine Learning Perspective

- Abstract away the conventional RMC particulars
- The storage model is viewed as stochastic simulator(s) that produces noisy pathwise observations conditional on the control
- Learn the continuation or q-value $q(t, P, I, m)$: cost-to-go conditional on next-step regime m; done recursively over t
- Sub-modules [each has multiple viable schemes]:
  - Approximation of the conditional expectation defining q;
  - Evaluation of the optimal control [trivial if m is discrete]
  - Evaluation of the pathwise continuation value
- Put another way:

RMC for Storage: Machine Learning Perspective

- Abstract away the conventional RMC particulars
- The storage model is viewed as stochastic simulator(s) that produces noisy pathwise observations conditional on the control
- Learn the continuation or q-value $q(t, P, I, m)$: cost-to-go conditional on next-step regime m; done recursively over t
- Sub-modules [each has multiple viable schemes]:
  - Approximation of the conditional expectation defining q;
  - Evaluation of the optimal control [trivial if m is discrete]
  - Evaluation of the pathwise continuation value
- Solve the storage control problem $\Leftrightarrow$ Recursively identify the control map + value function given a pathwise reward simulator
- Put another way:

RMC for Storage: Machine Learning Perspective

- Abstract away the conventional RMC particulars
- The storage model is viewed as stochastic simulator(s) that produces noisy pathwise observations conditional on the control
- Learn the continuation or q-value $q(t, P, I, m)$: cost-to-go conditional on next-step regime m; done recursively over t
- Sub-modules [each has multiple viable schemes]:
  - Approximation of the conditional expectation defining q;
  - Evaluation of the optimal control [trivial if m is discrete]
  - Evaluation of the pathwise continuation value
- Solve the storage control problem $\Leftrightarrow$ Recursively identify the control map + value function given a pathwise reward simulator
- Put another way: build an $\underbrace{\textit{empirical}}_{simulation-based}$ approximation to the value function $V(t, \cdot)$ — machine learning

L. -Maheshwari (2018)

- Develop a template for a simulation-based approach to stochastic storage
- Extend latest Regression Monte Carlo techniques to switching control
- A plug-and-play modular algorithm interpreted as a (machine/statistical) learning problem
- Nests existing literature + offers MANY more choices
- Illustrate with some examples (not meant to benchmark yet): Gas Storage/Microgrid Control
- Contributes to the `StOpt` library developed by Xavier Warin...

RMC Background: Optimal Stopping

- State process $X_\cdot$, payoff $\mathbf{h}(X_t)$, discrete-time: $t = 1, 2, \ldots, T$
- Objective: maximize reward $V(t,x) = \sup_\tau \mathbb{E}_{t,x}\left[h(X_\tau)\right]$
- Solution: $\tau^* = \inf\{t : X_t \in \mathfrak{S}_t\} \wedge T$. Stopping region: $\mathfrak{S}_t = \{x : V(t,x) = h(x)\}$
- Timing value (aka q-value):

$$T(t,x) := \mathbb{E}_{t,x}\left[V(t+1, X_{t+1})\right] - h(x) = \mathbb{E}_{t,x}\left[h(X_{\tau_{t+1}})\right] - h(x).$$

- Then $\mathfrak{S}_t = \{x : \mathbf{T}(\mathbf{t},\mathbf{x}) < \mathbf{0}\}$ and $V(t,x) = h(x) + \max(T(t,x), 0)$
- Simplest control problem: compare 2 alternatives and choose the best action
- (Later come back to multiple alternatives and multiple state variables)

Regression Monte Carlo

- The key step is to compute the **conditional expectation**
- Recast it as a learning task: put in $X_t = x$, get back $V(t+1, X_{t+1}^{t,x}) = y$
- Want to learn the input/output relationship between x's and y's
- Build an emulator (statistical surrogate) using some training data
- Use the emulator to predict for new test data
- Ingredients:
  - The emulator architecture
  - Training data
  - How to obtain y's in the context of DP
  - Performance metrics

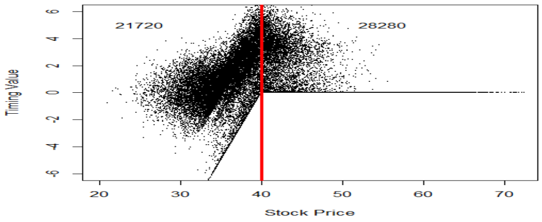
Regression Monte Carlo

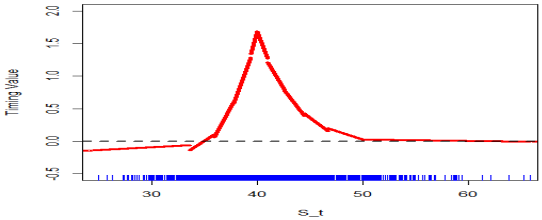
- The key step is to compute the **conditional expectation**
- Recast it as a learning task: put in $X_t = x$, get back $V(t+1, X_{t+1}^{t,x}) = y$
- Want to learn the input/output relationship between x's and y's
- Build an emulator (statistical surrogate) using some training data
- Use the emulator to predict for new test data
- Ingredients:
  - The emulator architecture
  - Training data
  - How to obtain y's in the context of DP
  - Performance metrics
- These questions are well-addressed within the Statistical Learning paradigm

First-Generation RMC

- Late '90s: Carriere '96, Tsitsiklis-van Roy '00, Longstaff-Schwartz '01
- Emulator architecture: classical OLS regression/linear model
  - Parametrize the emulator via basis functions
  - Empirically estimate the coefficients $\vec{\alpha}$
- Training data:
  - $x^n_t := X^{0,x_0,(n)}_t$ — database of N global X_t-paths
  - Probabilistic simulation design reflects the distribution of X^{0,x_0}_t
- Simulator responses:
  - TvR: $y^n_t = \hat{V}(t+1, X^{(n)}_{t+1})$, one-step look-ahead
  - LS: $\tau' = \inf\{s > t : X^{(n)}_s \in \hat{\mathfrak{S}}_s\}$ and $y^n_t := h(X^{(n)}_{\tau'})$ – pathwise reward
- Performance metric: $\sum_t \|\hat{V}(t,\cdot) - V(t,\cdot)\|_2$

Longstaff-Schwartz 1.0: $\{x_t, y_t\}^{1:N}$ pairs





- 1-D Bermudan Put in GBM model, $K = 40$ strike
- Piecewise linear approximation of $T(t, x)$ (10 knots in x) at $t = 0.6$
- 50,000 $x_t^n \sim \mathit{LogNormal}$
- wild response distribution + low signal-to-noise
- inefficient design: out-of-the-money $x_t > 40$ samples are useless
- Non-adaptive regression w/22 degrees of freedom

Enhancements: RMC v1.5

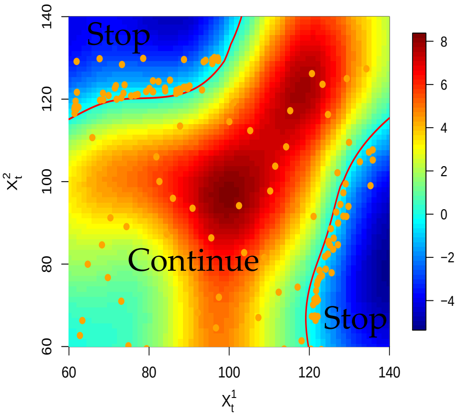
- Improved regression: adaptive bases; regularized; nonparametric;
- Improved performance metrics/ convergence proofs to handle dependent regressions
- Belomestny: $|V(0,x) - \hat{V}(0,x)| \le \mathbb{E}_{0,x}\left[\sum_{s=0}^{T} |T(s,X_s)| 1_{\{X_s \in Err_s\}}\right]$ where $Err_t := \mathfrak{S}_t \triangle \hat{\mathfrak{S}}_t = \{x : \text{sign } T(t,x) \neq \text{sign } \hat{T}(t,x)\}$.
- Hybrid methods for high-dimensions: regression + interpolation, regression + PDE
- Take expectations then project vs. Project then take expectations ("Regress Later")
- Warin, Gobet, Oosterlee, Belomestny, Stentoft, Kohler, Bender, Egloff, Tompaidis, ... (4000+ citations to LS)

Our team in the past 5 years: RMC 2.0

- Templated algorithm with a unified stochastic simulation view. Multiple strategies to be combined in a plug-and-play fashion
+ Adaptive Experimental designs
+ Modern emulator frameworks $\to$ Gaussian Process regression
+ Objective is to learn the sign of $T(t, x)$ (rank reward vs continuation) – contour-finding
- Multiple links to machine learning (Stats/CS/OR which all have their own terminology)

Illustrating RMC 2.0: 2D Max Call

- Color-coded according to $T(t,x)$
- Red contour indicates the stopping boundary
- **preferentially target regions** where $T(t,\cdot)$ changes signs
- Use active learning heuristics – add x sites where $\hat{T}(t,x) \simeq 0$ or where $Sd(\hat{T}(t,x))$ is large
- Bespoke regression + bespoke mesh (100 adaptive sites x 40 replications)



Dynamic Emulation for Stochastic Storage Problems

- Recall the building blocks of RMC:
- Regression – how to approximate the conditional expectation (parametric, non-parametric, bivariate)
- Design – which data to use for the regression/experiment design: judiciously selecting simulations to run
- Simulation – how to simulate pathwise continuation values
- Combine above with a modular framework that brings mix-and-match capabilities
- Advantages of DAE
  - New choices: nonparametric regression
  - New adaptive designs (rather than one ad hoc proposal)
  - New batched designs
  - Straightforward to modify for different contexts/higher dimensions
  - Easy to incorporate additional bells-and-whistles
- Unifies Optimal Stopping RMC and Storage RMC (single software library)

Dynamic Emulation Algorithm

Data: K (time steps), (N_k) (simulation budgets per step) // N_k and D_k can change line to line

Generate design $\mathcal{D}_{K-1,m} := (\mathbf{P}_{K-1}^{\mathcal{D}_{K-1,m}}, \mathbf{I}_K^{\mathcal{D}_{K-1,m}})$ of size N_{K-1} for each $m \in \mathcal{J}$.

Generate one-step paths $P_{K-1}^{n,\mathcal{D}_{K-1,m}} \mapsto P_K^{n,\mathcal{D}_{K-1,m}}$ for $n = 1, \dots, N_{K-1}$ and $m \in \mathcal{J}$

Terminal condition: $v_{K,m}^n \leftarrow W(P_K^{n,\mathcal{D}_{K-1,m}}, I_K^{n,\mathcal{D}_{K-1,m}})$ for $n = 1, \dots, N_{K-1}, m \in \mathcal{J}$

for $k = K-1, \dots, 1$ **do** // Loop in time-steps

  for $m \in \mathcal{J}$ **do**

    $\hat{q}(k,\cdot,\cdot,m) \leftarrow \arg\min_{h_k \in \mathcal{H}_k} \sum_{n=1}^{N_k} |h_k(P_k^{n,\mathcal{D}_{k,m}}, I_{k+1}^{n,\mathcal{D}_{k,m}}) - v_{k+1,m}^n|^2$ // Regression

    Generate design $\mathcal{D}_{k-1,m} := (\mathbf{P}_{k-1}^{\mathcal{D}_{k-1,m}}, \mathbf{I}_k^{\mathcal{D}_{k-1,m}})$ of size N_{k-1} for each $m \in \mathcal{J}$

    Generate one-step paths $P_{k-1}^{n,\mathcal{D}_{k-1,m}} \mapsto P_k^{n,\mathcal{D}_{k-1,m}}$ for $n = 1, \dots, N_{k-1}$ // TvR style

  end

  for $n = 1, \dots, N_{k-1}$ *and* $m \in \mathcal{J}$ **do** // Predict

    $m' \leftarrow \mathrm{argmax}_{j \in \mathcal{J}} \{\pi^\Delta(P_k^{n,\mathcal{D}_{k-1,m}}, m, j) + \hat{q}(k, P_k^{n,\mathcal{D}_{k-1,m}}, I_k^{n,\mathcal{D}_{k-1,m}} + a(c_k(j))\Delta t, j)\}$

    $v_{k,m}^n \leftarrow \pi^\Delta(P_k^{n,\mathcal{D}_{k-1,m}}, m, m') + e^{-r\Delta t}\hat{q}(k, P_k^{n,\mathcal{D}_{k-1,m}}, I_k^{n,\mathcal{D}_{k-1,m}} + a(c_k(m'))\Delta t, m')$

  end

end

return $\{\hat{q}(k,\cdot,\cdot,m)\}_{k=1,m\in\mathcal{J}}^{K-1}$ // Output is a collection of fitted emulators

New Regression Proposal: Gaussian Processes

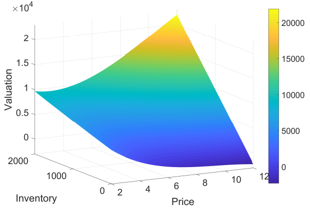
- Random field representation of the unknown q-value – non-parametric, similar to kernel regression. $x \equiv (P, I)$
- Squared-Exp. Kernel: $H_{ij} := \kappa(x^i, x^j) = \sigma_f^2 \exp\Big(-\frac{1}{2}(x^i - x^j)^T \mathbf{\Sigma}^{-1}(x^i - x^j)\Big)$
- Hyperparameters: lengthscales Σ_{ii} – smoothness in the i-th dimension; process variance σ_f controls amplitude, prior mean $m(\cdot)$.
- The estimator is in terms of posterior mean/posterior variance, obtained from MVN conditional formulas
- Prediction at x_* after training at $\mathbf{x}$:

$$\hat{q}(k, x_*, m) = m(x_*) + H_* \mathbf{H}^{-1}(\mathbf{v_{k+1,m}} - m(\mathbf{x}))$$

- Fitting a GPR means estimating the hyperparameters

Using GP emulators in DEA

- GPR is good for non-uniform designs since intrinsically a local interpolator; this is crucial for smooth bivariate fits
- GPR has N' degrees of freedom for N' unique x_t^n's
- Utilize a batched design (like a MC forest) – multiple y's at each x
- Batching improves signal-to-noise ratio and reduces GP regression overhead.
- Posterior variance can be used to create fully adaptive designs ("Active Learning")



Other Regression Choices

- Global Polynomial regression
- Regress only in P; discretize + interpolate in I
- LOESS regression
- Any approximation architecture can be used – just need `train` and `predict` methods
- The performance is necessarily linked to the choice of the training data

How to train your DEA: $\mathcal{D}_k := (x, y)^{1:N_k}$

- In what dimension: Joint -2D (P, I) or -1D by discretize+interpolate in I
- Space-filling vs Targeted – explore the entire state space or focus on most relevant regions
- How to generate $\mathcal{D}_k$: is $\mathbf{x}$ randomized or deterministic across runs?
- Replication/Batching – re-use same x for multiple simulations, then pre-average pathwise value a la nested MC
- Design Size – N_k can be time-dependent
- Mix-and-match across time-steps

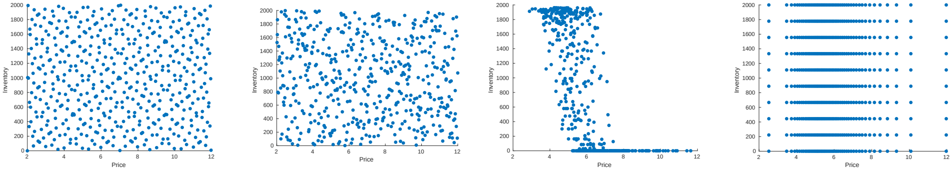
Comparing Implementations of DEA: Gas Storage

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

Table: 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 $N \simeq 100K$.

Explaining Design Choices

- Probabilistic – reflects the distribution of $(P_t, \hat{I}_t)$
- Space Filling: explore continuation values throughout the input domain. Sub-choices:
  - Quasi Monte Carlo sequences (Sobol)
  - Latin Hypercube Sampling
  - Gridded
- Adaptive – target efficient learning of the action boundaries



(a) 2D Sobol QMC sequence $\mathcal{S}_2$

(b) 2D randomized space-filling $\mathcal{L}_2$

(c) Joint Probabilistic design $\mathcal{P}_2$

(d) Conventional product design $\mathcal{P} \times \mathcal{G}$

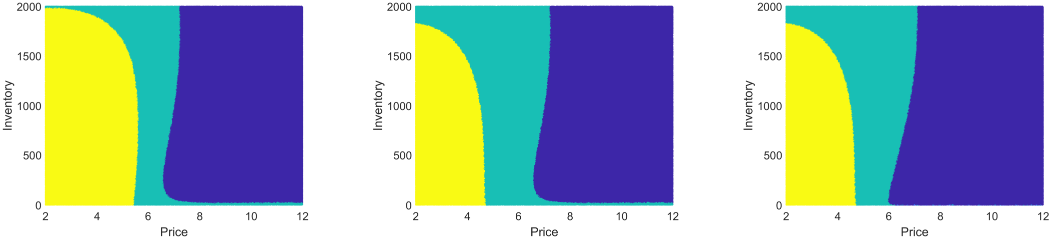
Figure: Different simulation designs $\mathcal{D}$; in all cases $N = 500$.

Some Conclusions

- Space-filling designs do not target enough (unless really fine-tune the input domain)
- Probabilistic designs do not explore enough
- Mixtures of the two work very well
- In 1+1 dimensions, the -1D methods work extremely well, but GP-based -2D regression is competitive
- The latter requires batched designs for efficient computation
- GPR straightforwardly generalizes to higher dim (cf. 3D example with 2 storage facilities in the paper)

Add Switching Costs

With switching costs $K(m, m')$ the current regime m_t becomes part of the state due to preference for inertia. Now have to compute 3 different continuation functions $\hat{q}(\cdot, m)$.

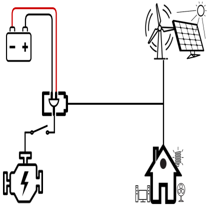


(a) $m = +1$ (Inject) (b) $m = 0$ (Store) (c) $m = -1$ (Withdraw)

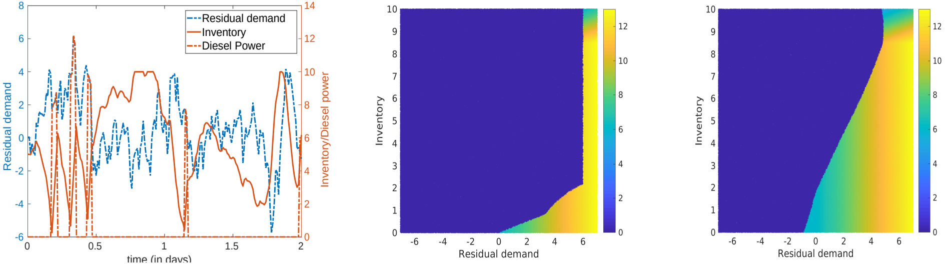
Figure: The control maps $\hat{m}(t, P, I, m)$ at $t = 2.7$ years for the model with switching costs $K(-1, 1) = K(0, 1) = 15000$; $K(1, -1) = K(0, -1) = 5000$; $K(1, 0) = K(-1, 0) = 0$. The colors are $\hat{m}_{t+\Delta t} = +1$ (inject, light yellow), $\hat{m}_{t+\Delta t} = 0$ (store, medium cyan), $\hat{m}_{t+\Delta t} = -1$ (withdraw, dark blue). The solution used GP-2D regression with Mixture design.

Microgrids: Avoiding Blackouts

- Renewable generation facility (solar or wind)
- Microgrid demand – Net demand X_t (e.g. follows an OU)
- Battery to smooth out local fluctuations; +diesel generator for back-up
- Battery $I_{t_{k+1}} = I_{t_k} + a(c_{t_k})\Delta t = I_{t_k} + B_{t_k}\Delta t$
- Diesel is OFF or ON ($m = 1$) with output
  $c_{t_k}(1) = X_{t_k}\mathbf{1}_{\{X_{t_k}>0\}} + B_{\max} \wedge \frac{I_{\max}-I_{t_k}}{\Delta t}$
- Battery output for balancing purposes:
  $B_{t_k} := a(c_{t_k}) = -\frac{I_{t_k}}{\Delta t} \vee \left(B_{\min} \vee (c_{t_k} - X_{t_k}) \wedge B_{\max}\right) \wedge \frac{I_{\max}-I_{t_k}}{\Delta t}.$
- Imbalance $S_{t_k} = c_{t_k} - X_{t_k} - B_{t_k}$ – should be zero
- $S_{t_k} > 0$ – **curtailment**; $S_{t_k} < 0$ – **blackout**
- Cost: $\pi(c, X) := -c^{\gamma} - |S|\left[C_2\mathbf{1}_{\{S<0\}} + C_1\mathbf{1}_{\{S>0\}}\right]$



Controlling the Diesel

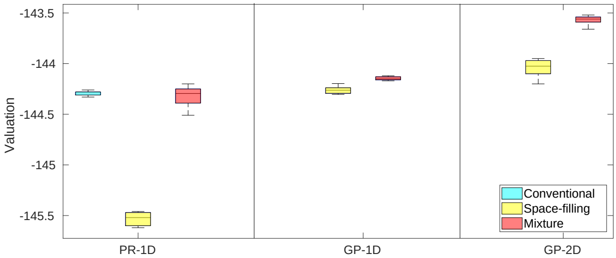


(a) Pathwise trajectory (b) Generator OFF ($m = 0$) (c) Generator ON ($m = 1$)

Figure: *Left panel:* trajectory of the residual-demand (X_t), corresponding to policy (c_t) and the resultant inventory trajectory $(\hat{I}_t)$. *Middle and right panels:* the control policy $\hat{c}(t, X, I, m)$ at $t = 24$ hours. Recall that $c(0) = 0$ whenever the diesel is OFF. All panels are based on GP-2D regression and Mixture design $\mathcal{D} = \mathcal{P}_2(0.5N) \cup \mathcal{L}_2(0.5N)$.

DEA for Microgrid Control

Straightforward adaptation; Discretize potential diesel output, $m \in \{0, 1, 2, \ldots, 10\}$



- Joint regression together with a mixture design wins again
- Discretization in I works less well because the solution is more nonlinear
- Quite different setup, but results are consistent with the storage example

Looking Ahead: RMC v3.0

- RMC is poised to be a centerpiece of Applied Stochastics
- Export the outlined ideas to:
  - Quantitative Finance: XVA, parametric pricing/hedging
  - Risk Management: (T)VaR estimation, capital requirements
  - Probability: BSDEs
  - Control: approximate dynamic programming
  - OR/UQ: stochastic simulation
- Ultimate goals: speed; scalability; efficiency; adaptivity
- Handle more complicated (i) optimization in m; (ii) risk attitudes; (iii) constraints

References

Gramacy R., and M. Ludkovski, 2015. Sequential Design for Optimal Stopping Problems, SIAM Journal on Financial Mathematics, 6(1), 748–775.

Ludkovski, M., 2018. Kriging Metamodels and Experimental Design for Bermudan Option Pricing. Journal of Computational Finance, 22(1), 37–77, arxiv.org/abs/1509.02179

Ludkovski, M and A. Maheshwari, 2018. Simulation Methods for Stochastic Storage Problems: A Statistical Learning Perspective, Submitted. arxiv.org/abs/1803.11309

Binois, M., Gramacy, R. B., Ludkovski, M., 2018. Practical heteroskedastic Gaussian process modeling for large simulation experiments. Journal of Computational and Graphical Statistics, to Appear. arXiv:1611.05902.

Thank You!