Kernel-Based Learning for Smart Inverter Control

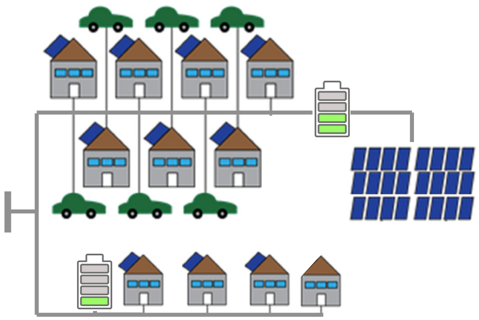
Aditie Garg, Mana Jalali, Vassilis Kekatos

November 29, 2018
Anaheim, CA

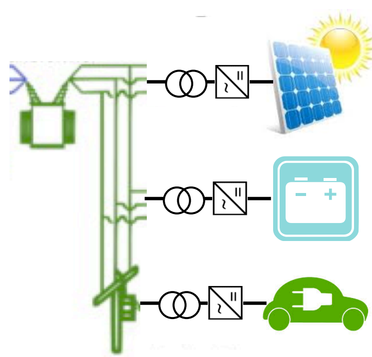
Acknowledgements NSF-CAREER-1751085

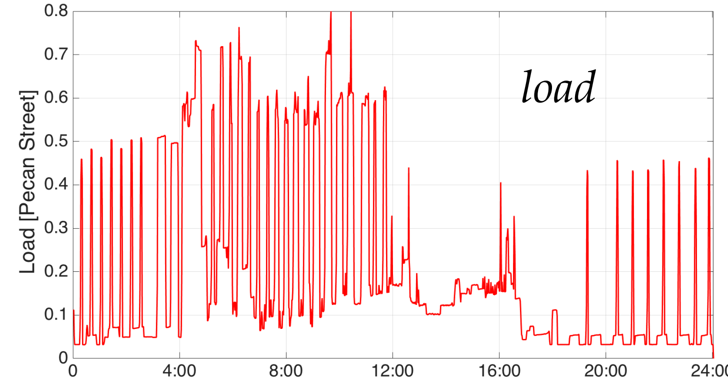
Motivation

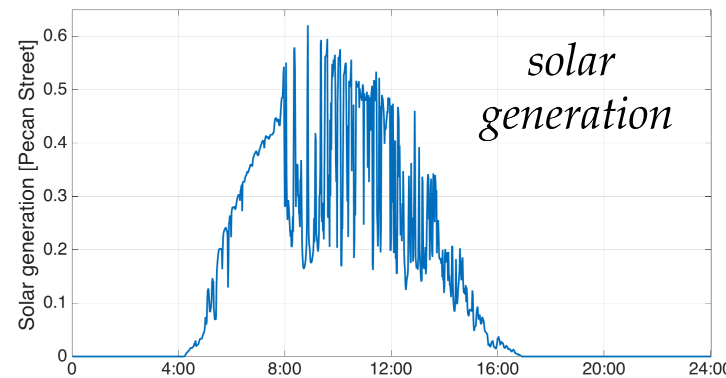
- Voltage fluctuations due to renewables
- Inefficiency of voltage control devices

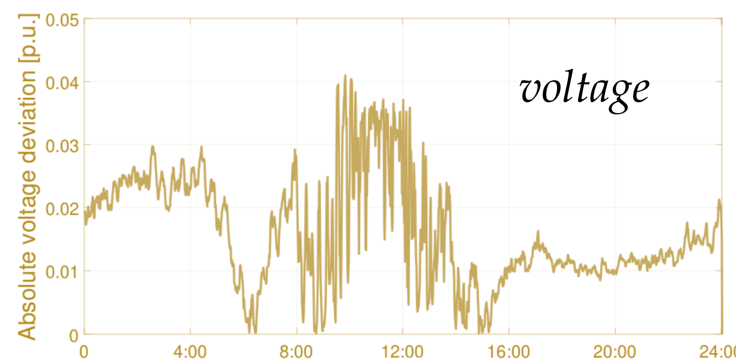


- Reactive power control with inverters



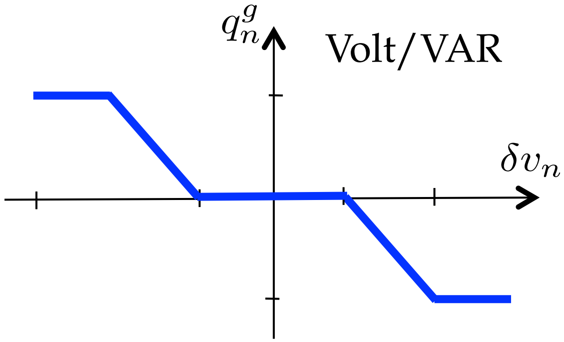


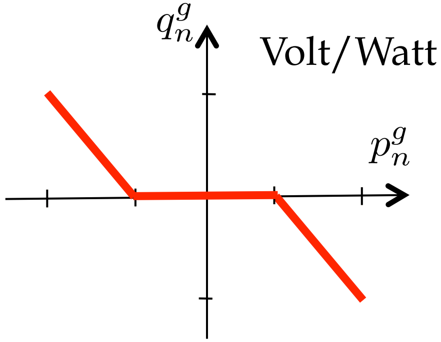




Finding reactive power setpoints

- Local control curves [Turitsyn'11; Kekatos-Baldick'15; IEEE 1547 standard]





✓ no cyber requirements
✓ suboptimal

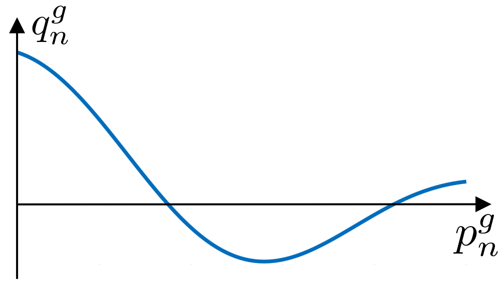
- Centralized OPF [Lavaei-Low'14, Farivar-Low'15]

✓ *high* cyber requirements
✓ optimal, possibly obsolete

- Decentralized OPF
  [Dall'Anese-Dhople-Giannakis'15, Peng-Low'16]

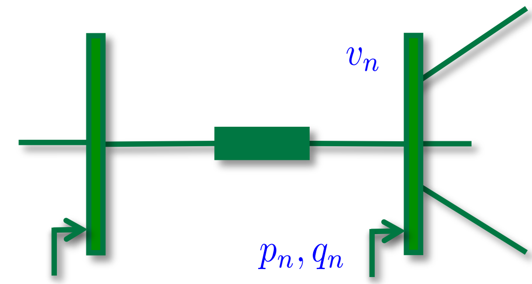
✓ *moderate* computations
✓ high communications

- Quasi-static control policies



✓ low communications
✓ close to optimal

Approximate grid model



- Single-phase *radial* grid with *N+1* nodes and *N* lines

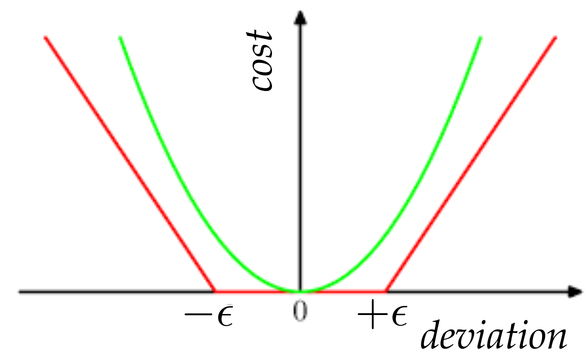
- Linearized distribution flow (LDF) model
  [Baran-Wu'89], [Deka et al'17]

$$\tilde{\mathbf{v}} \simeq \mathbf{R}(\mathbf{p}^g - \mathbf{p}^c) + \mathbf{X}(\mathbf{q}^g - \mathbf{q}^c)$$
$$= \mathbf{X}\mathbf{q}^g + \mathbf{y}$$

- Voltage deviation options

  - least-squares $\Delta_s(\mathbf{q}^g) = \|\mathbf{X}\mathbf{q}^g + \mathbf{y}\|_2^2$

  - epsilon-insensitive $\Delta_\epsilon(\mathbf{q}^g) = \sum_{n=1}^{N} [\mathbf{e}_n^\top (\mathbf{X}\mathbf{q}^g + \mathbf{y})]_\epsilon$

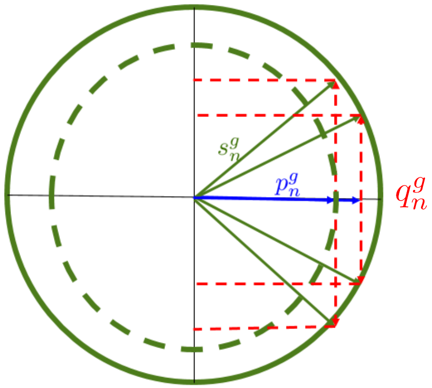


- Approximate ohmic losses [Turitsyn'11] $L(\mathbf{q}^g) = (\mathbf{q}^g - \mathbf{q}^c)^\top \mathbf{R}(\mathbf{q}^g - \mathbf{q}^c)$

Problem formulation

- Inverter setpoints to minimize voltage deviation and/or losses

$$\min_{\mathbf{q}^g} \quad C_\lambda(\mathbf{q}^g; \mathbf{y}) = \lambda \Delta(\mathbf{q}^g) + (1-\lambda) L(\mathbf{q}^g)$$
$$\text{s.to} \quad -\bar{\mathbf{q}}^g \leq \mathbf{q}^g \leq \bar{\mathbf{q}}^g$$



- Inverter setpoints as policies $q_n^g(\mathbf{z}_n) = f_n(\mathbf{z}_n)$

  - remote and local inputs $\mathbf{z}_n = [p_n^g - p_n^c \quad \bar{q}_n^g \quad q_n^c]^\top$

- Control rules as *affine* policies

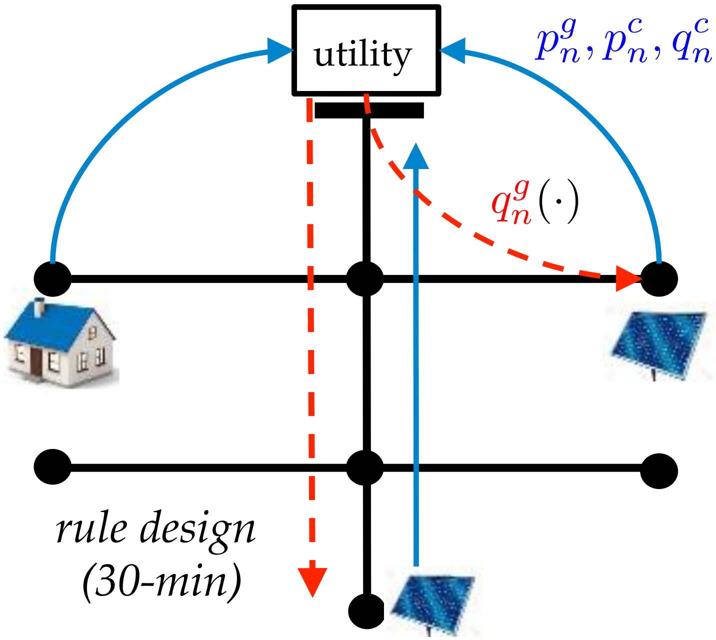
  - chance-constrained [Cherkov-Bienstock'15]; [Ayyagari-Gatsis-Taha'17]

  - robust formulations [Jabr'18]; [Lin-Bitar'18]

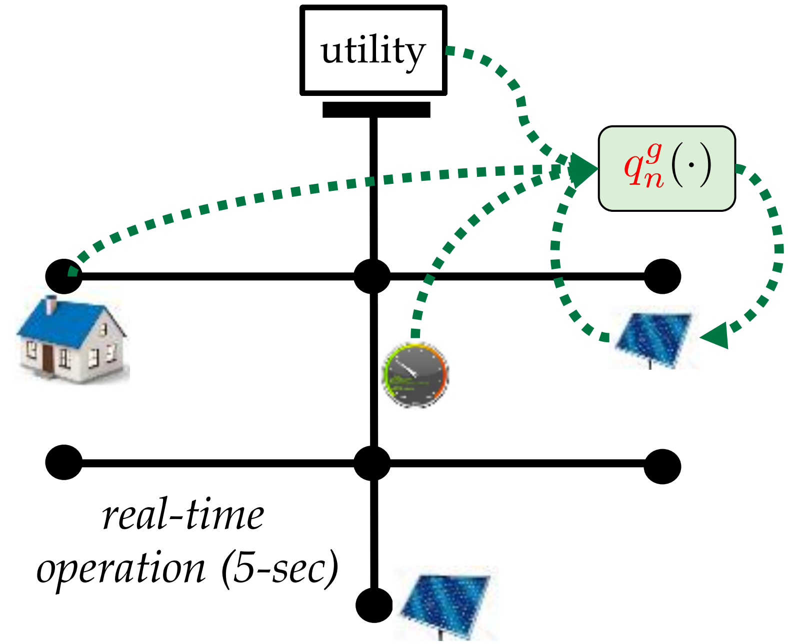
  - closed-loop [Baker, Bernstein, Dall'Annese, Zhao'18]

- Policy outputs heuristically projected within feasible range

Training and operation



- ✓ utility collects injection data
- ✓ designs control rules
- ✓ rules downloaded to inverters



- ✓ inverters operate in real-time
- ✓ remote data transferred to inverters

- Design control rules as *non-linear* policies

Kernel-based learning

- Given data $\{(x_t \in \mathcal{X}, z_t \in \mathbb{R})\}_{t=1}^{T}$, and kernel function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$

$$f^* = \arg\min_{f \in \mathcal{H}_{\mathcal{K}}} \sum_{t=1}^{T} (z_t - f(x_t))^2 + \mu \|f\|_{\mathcal{K}}$$

where $\mathcal{H}_{\mathcal{K}} := \left\{ f(x) = \sum_t K(x, x_t) a_t \right\}$

- *Representer's Theorem:* Minimizing function depends only on training data

$$f^*(x) = \sum_{t=1}^{T} K(x, x_t) a_t^*$$

- Functional minimization as vector optimization

$$\arg\min_{\mathbf{a}} \|\mathbf{z} - \mathbf{K}\mathbf{a}\|_2^2 + \mu \sqrt{\mathbf{a}^\top \mathbf{K} \mathbf{a}}$$

Hastie, Tibshirani, Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer, 2009.

Least-squares regression

- Control rule design as function fitting using T scenario data

$$\begin{aligned} \min \quad & \sum_{t=1}^{T} C_\lambda(\mathbf{q}_t^g; \mathbf{y}_t) + \mu \sum_{n=1}^{N} \|f_n\|_{\mathcal{K}_n} \\ \text{over} \quad & q_{n,t}^g = f_n(\mathbf{z}_{n,t}) + b_n, \; \{f_n \in \mathcal{H}_{\mathcal{K}_n}\} \\ \text{s.to} \quad & |q_{n,t}^g| \leq \bar{q}_{n,t}^g \end{aligned}$$

- Jointly learning inverter functions can be solved as a quadratic program

$$q_{n,t}^g(\mathbf{z}_{n,s}) = \sum_{t=1}^{T} K(\mathbf{z}_{n,s}, \mathbf{z}_{n,t}) a_{n,t}^* + b_n^*$$

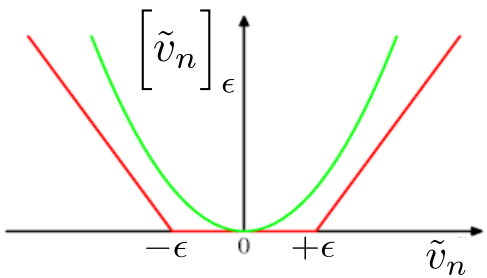
- Rule described by pre-specified kernel and $\{\mathbf{z}_{n,t}, a_{n,t}^*\}_{t=1}^{T}, b_n^*$

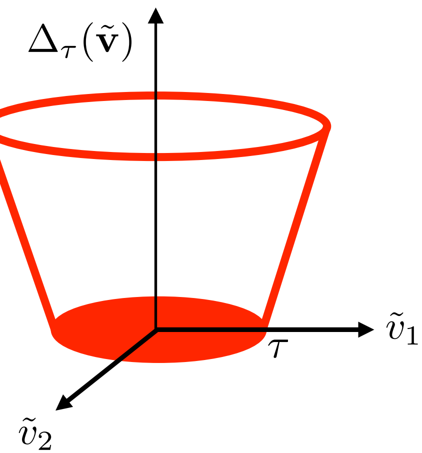
Support vector inverter control

Lemma: Voltage costs $(\lambda = 1)$ inducing *sparse* representations for inverter rules

$$\Delta_\epsilon(\mathbf{q}^g) = \sum_{n=1}^{N} \left[\tilde{v}_n\right]_\epsilon \quad \Longrightarrow \quad a_{n,t} = 0 \quad \text{if} \quad |\tilde{v}_{m,t}| \leq \epsilon \ \forall m$$

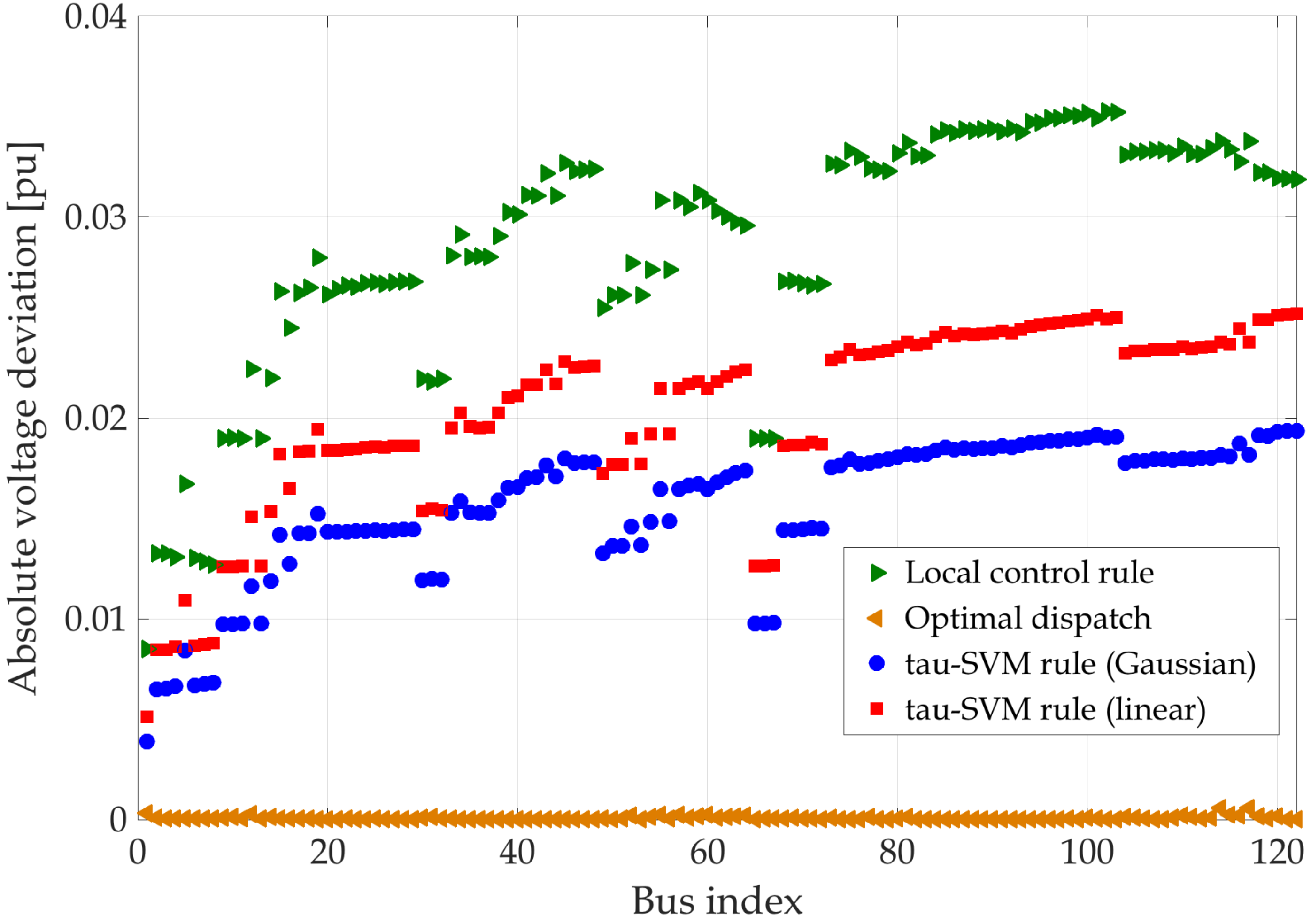
$$\Delta_\tau(\mathbf{q}^g) = \left[\|\tilde{\mathbf{v}}\|_2\right]_\tau \quad \Longrightarrow \quad a_{n,t} = 0 \quad \text{if} \quad \|\tilde{\mathbf{v}}_t\|_2 \leq \tau$$





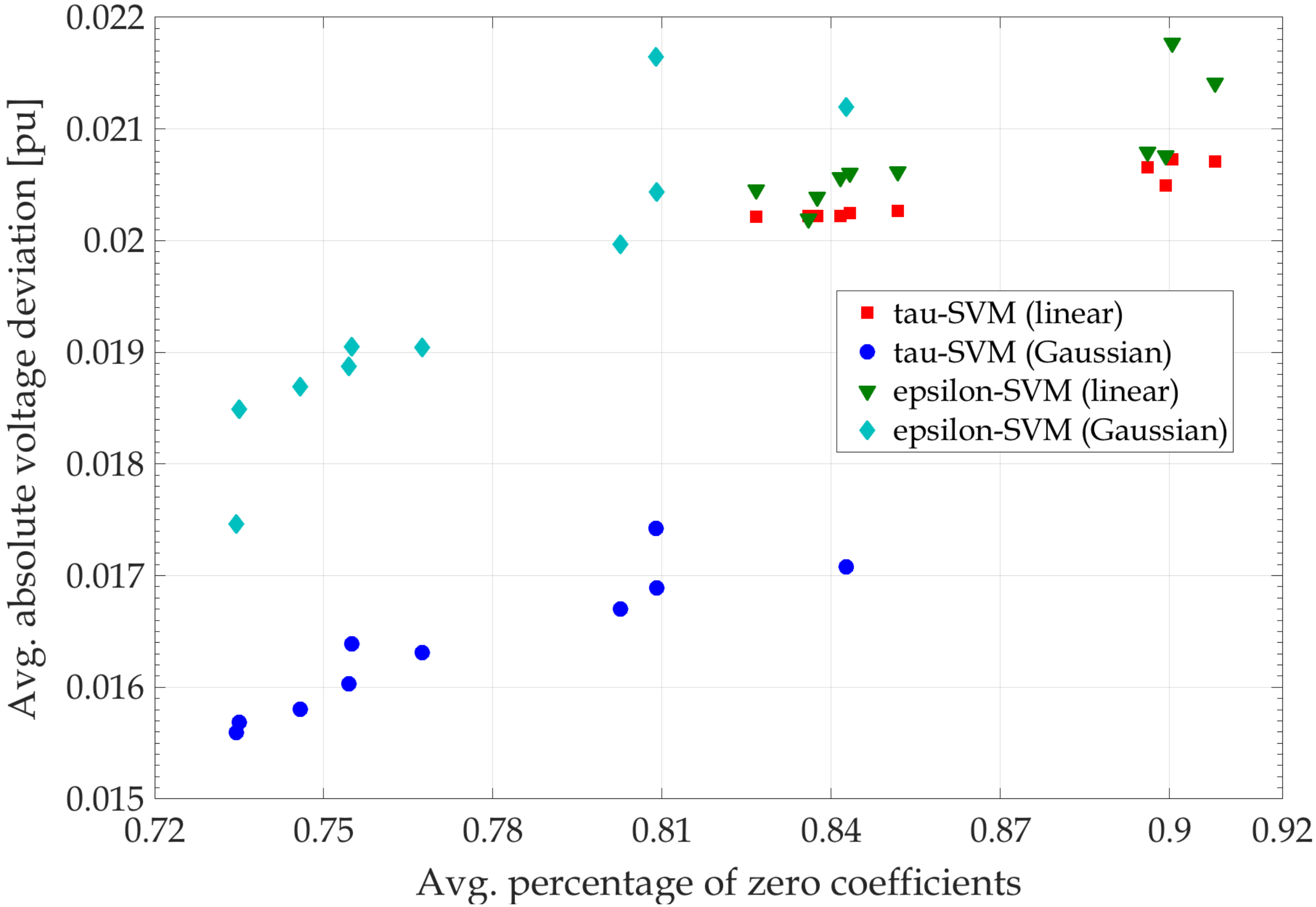
- Different from SVMs, *block* voltage penalties yield *support feeder scenarios*

Numerical tests



- Pecan Str data (8am-8pm) on IEEE 123-bus feeder (1-phase)
- 70% solar penetration with 1.1 inverter oversizing
- Train for T=30 one-minute data; validate on next 30 one-min data

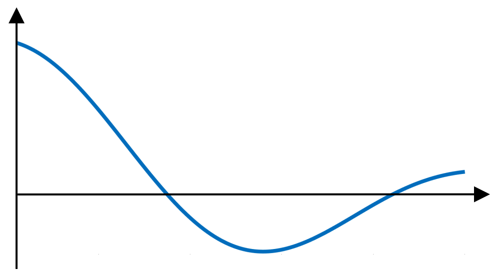
Performance vs. sparsity

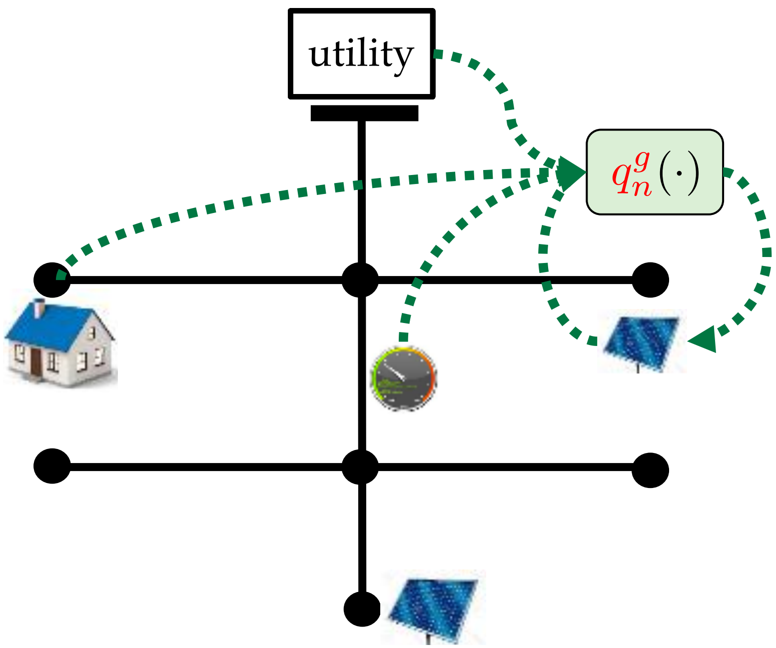


- Sparsity in coefficients is controlled by ϵ or τ through cross-validation

Conclusions

- ☑ learning non-linear control policies
- ☑ data-based feeder-wide designs
- ☑ SVM costs for communication savings





- ☐ closed-loop control
- ☐ remote input and kernel selection
- ☐ multi-stage formulations

Thank you!