Voltage Regulation of Smart Grids using Machine Learning Tools

Mana Jalali

September 23, 2019

Blacksburg, Virginia

Keywords: Support vector machines, multi-kernel learning, voltage regulation, linearized distribution flow model

Voltage Regulation of Smart Grids using Machine Learning Tools

Mana Jalali

(ABSTRACT)

Smart inverters have been considered the primary fast solution for voltage regulation in power distribution systems. Optimizing the coordination between inverters can be computationally challenging. Reactive power control using fixed local rules have been shown to be subpar. Here, nonlinear inverter control rules are proposed by leveraging machine learning tools. The designed control rules can be expressed by a set of coefficients. These control rules can be nonlinear functions of both remote and local inputs. The proposed control rules are designed to jointly minimize the voltage deviation across buses. By using the support vector machines, control rules with sparse representations are obtained which decrease the communication between the operator and the inverters. The designed control rules are tested under different grid conditions and compared with other reactive power control schemes. The results show promising performance.

Voltage Regulation of Smart Grids using Machine Learning Tools

Mana Jalali

(GENERAL AUDIENCE ABSTRACT)

With advent of renewable energies into the power systems, innovative and automatic monitoring and control techniques are required. More specifically, voltage regulation for distribution grids with solar generation is a can be a challenging task. Moreover, due to frequency and intensity of the voltage changes, traditional utility-owned voltage regulation equipment are not useful in long term. On the other hand, smart inverters installed with solar panels can be used for regulating the voltage. Smart inverters can be programmed to inject or absorb reactive power which directly influences the voltage. Utility can monitor, control and sync the inverters across the grid to maintain the voltage within the desired limits. Machine learning and optimization techniques can be applied for automation of voltage regulation in smart grids using the smart inverters installed with solar panels. In this work, voltage regulation is addressed by reactive power control.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Integration of residential and commercial solar generation into the power system is an upcoming challenging task. Solar farms which are usually connected at the end of long feeders can incur voltage excursions across the grid, while frequent power flow may exceed the apparent power capabilities of substation transformers [3]. Solar generation from residential photovoltaics (PVs) can fluctuate by up to 15% of their rating within one-minute [4]. Utility-owned voltage control equipment, such as load-tap-changing transformers, capacitor banks, and step-voltage regulators, perform discrete control actions, and their lifespan is related to the number of switching operations [5]. Voltage regulation under increasing renewable generation may require more frequent switching and further installations of the voltage control equipment, thus critically challenging reactive power control in distribution grids.

On the other hand, PVs are interfaced by inverters featuring advanced communication, metering, and control functionalities. Using inverters for reactive power control has been advocated as a fast-responding solution [3]. The amended IEEE 1547 standard allows inverters to be operating at non-unit power factors [6]. Nonetheless, coordinating in real-time hundreds of inverters distributed over a feeder is a formidable task. In a typical setup, the values of instantaneous loads and solar generation are communicated to a utility controller; the controller minimizes ohmic losses subject to voltage regulation constraints; and the computed setpoints are sent back to the inverters. The problem of finding the optimal reactive injection setpoints for inverters is an instance of the optimal power flow (OPF) task, which is not convex in general. To tackle the problem of non-convexity, convex relaxations have been alternatively proposed; see [7] for a survey. The uncertainty in loads and solar generation over the next control period is usually accounted for through stochastic and robust formulations [8], [9]. To reduce complexity, approximate grid models have also been employed [10], [11], [12]; though heavy two-way utility-inverter communication is still needed.

Alternatively, decentralized solvers where inverters decide their setpoints upon communicating with neighboring inverters have been devised [13], [14], [15]. On the other extreme, localized schemes suggest having inverters implementing Volt-VAR and/or Watt-VAR curves given only local measurements [3]. Although such rules have been analytically shown to be stable and fast-converging, their equilibria unfortunately do not coincide with the sought OPF minimizers [16], [10], [17], [18]. In fact, there are some cases where no-reactive support option outperform the local rules [19].

The previous literature review indicates that centralized schemes incur high computational complexity; decentralized solvers require multiple communication exchanges among inverters; and local schemes have no performance guarantees. As a middle-ground solution, inverter setpoints can be designed in a quasi-static fashion via control rules. A rule expresses each setpoint as an affine function of given inputs, such as generation, load, or voltage. Albeit the related control rules are optimized periodically in a centralized way, the rules operate in real time. Inverter control using affine rules has been accomplished using chance-constrained [20]; robust [19], [21]; and closed-loop formulations [22]. However, optimal rules are not necessarily linear: If an apparent power constraint becomes active, reactive injections can become nonlinear functions of solar generation. To capture this nonlinearity, recent approaches engage learning models which are *trained to optimize*: Given pairs of grid conditions (load and solar generation) and their optimal inverter dispatches computed, the aforesaid approaches learn dispatch rules using linear or kernel-based regression [1], [2].

This work combines machine learning tools with physical grid models, and advocates a kernel-based approach for designing inverter control rules. The contribution is on two fronts: First, the design of inverter control rules is posed as a multi-task learning problem. Each inverter rule is modeled as a nonlinear function of control inputs. Rules are coupled through the electric grid to yield a system voltage profile. Using an approximate grid model, inverter rules are learned jointly so that they minimize a voltage regulation cost using anticipated load and solar generation scenarios. The precessor of this paper [23], [24] also adopts a kernel-based learning approach for designing nonlinear reactive power control rules. Each rule is described by a set of coefficients, one for each scenario. As a second contribution, we engineer the voltage regulation objective, so that the optimal rules are described by a few scenario coefficients. Such parsimonious representation of inverter rules saves communications. Numerical tests on a benchmark feeder showcase the advantages of nonlinear rules and explore the trade-off between voltage regulation and sparse rules. Contents of this work has been later submitted for IEEE transaction on smart grids [25].

Regarding notation, lower- (upper-) case boldface letters denote column vectors (matrices), while calligraphic symbols are reserved for sets. Symbol $^\top$ stands for transposition and $\|\mathbf{x}\|_2$ denotes the ℓ_2-norm of $\mathbf{x}$.

Chapter 2

Reactive Power Control

This chapter formulates the task of voltage regulation using inverters. Consider a distribution grid having $N+1$ buses served by the substation indexed by $n = 0$. Let v_n denote the voltage magnitude, and $p_n + jq_n$ the complex power injection at bus n. The active injection p_n is decomposed into $p_n = p_n^g - p_n^c$, where p_n^g is the solar generation and p_n^c the inelastic load at bus n. Reactive injections can be also expressed as $q_n = q_n^g - q_n^c$. Collect injections in N-length vectors:

$$\mathbf{p} = \mathbf{p}^g - \mathbf{p}^c \quad \text{and} \quad \mathbf{q} = \mathbf{q}^g - \mathbf{q}^c. \tag{2.1}$$

The reactive power injected by inverter n is constrained as

$$|q_n^g| \leq \bar{q}_n^g := \sqrt{(\bar{s}_n^g)^2 - (p_n^g)^2} \tag{2.2}$$

where $\bar{s}_n^g$ is the apparent power limit for inverter n; see [3].

Given loads $(\mathbf{p}^c, \mathbf{q}^c)$ and solar generation $\mathbf{p}^g$, voltage regulation aims at optimally setting $\mathbf{q}^g$ such that voltage deviations are kept minimal. To formally describe this task, one has to deal with the nonlinear power flow equations relating voltages to power injections. Trading modeling accuracy for computational tractability, we resort to the linearized model [26]

$$\mathbf{v} \simeq \mathbf{R}\mathbf{p} + \mathbf{X}\mathbf{q} + v_0\mathbf{1} \tag{2.3}$$

where $\mathbf{v} := [v_1 \; \dots \; v_N]^\top$ and matrices $(\mathbf{R}, \mathbf{X})$ depend on the feeder. These matrices are assumed to be known and if unknown, methods like the one inTKC19 can be used. Model (2.3) can be derived by linearizing the power flow equations around the flat voltage profile. In fact, the linearization can be performed at any system state $\mathbf{v}_0$, yet matrices $\mathbf{R}$ and $\mathbf{X}$ would then depend on the state $\mathbf{v}_0$; see [22]. From (2.1) and (2.3), the vector of voltage deviations from its nominal value can be approximated as

$$\mathbf{v} - v_0\mathbf{1} = \mathbf{X}\mathbf{q}^g + \mathbf{y} \tag{2.4}$$

where $\mathbf{y} := \mathbf{R}(\mathbf{p}^g - \mathbf{p}^c) - \mathbf{X}\mathbf{q}^c$ and $\mathbf{1}$ is a vector of all ones.

The goal here is to design the inverter injections $\mathbf{q}^g$ so that bus voltage magnitudes remain within regulation limits. The ANSI-C.84.1 standard dictates that service (load) voltages should remain within $\pm 5\%$ per unit (pu). However, our grid model of (2.4) stops at the level of distribution transformers. A distribution (pole or pad-mounted) transformer may be serving several residential customers. Each customer is typically connected to the distribution transformer through a triplex cable, which incurs a voltage drop between the transformer and the service voltage: Suppose a customer is connected to a 50 kVA, 7200-240/120 V center-tapped transformer via a 1/0 AA 100-ft triplex cable. The customer runs a constant-current load of 10 kVA at the nominal voltage of 120 V with 0.9 lagging power factor. If load currents are equally distributed among the three supplies (two 120 V and one 240 V), the service voltage drops by 1.5% pu. If the load is distributed among supplies non-uniformly, the service voltage can drop by even 3.5% pu. Due to this, the current practice is to maintain voltages at distribution transformers within $\pm 3\%$ pu, to ensure that service voltages remain within $\pm 5\%$ pu; see exercises of [27].

Given loads, solar generation, and grid parameters, the goal is to decide $\mathbf{q}^g$ to regulate voltage while satisfying the apparent power constraints of (2.2). The setpoints for reactive power injections from inverters can be found as the minimizer

$$\tilde{\mathbf{q}}^g := \arg\min_{\mathbf{q}^g \in \mathcal{Q}} \Delta(\mathbf{q}^g; \mathbf{y}). \tag{2.5}$$

The set $\mathcal{Q} \subseteq \mathbb{R}^N$ captures the constraints in (2.2) for all n; and $\Delta(\mathbf{q}^g; \mathbf{y})$ is a voltage regulation objective. A typical choice for Δ is the sum of squared voltage deviations [16], [19], [22]

$$\Delta_s(\mathbf{q}^g; \mathbf{y}) := \sum_{n=1}^{N} (v_n - v_0)^2 = \|\mathbf{X}\mathbf{q}^g + \mathbf{y}\|_2^2. \tag{2.6}$$

Alternatively, the utility may want to maintain voltages within the range of $(1 \pm \epsilon)v_0$ for say $\epsilon = 0.03$. Then, a pertinent objective is [17]

$$\Delta_\epsilon(\mathbf{q}^g; \mathbf{y}) := \sum_{n=1}^{N} [v_n - v_0]_\epsilon = \sum_{n=1}^{N} \left[\mathbf{e}_n^\top (\mathbf{X}\mathbf{q}^g + \mathbf{y})\right]_\epsilon \tag{2.7}$$

where $\mathbf{e}_n$ is the n-th canonical vector of length N, and the operator $[\cdot]_\epsilon$ is defined as

$$[x]_\epsilon := \begin{cases} 0 & , \ |x| \leq \epsilon \\ |x| - \epsilon & , \ \text{otherwise} \end{cases} . \tag{2.8}$$

Function Δ_ϵ returns zero when all voltages are within limits. Otherwise, it increases linearly with voltage excursions; see [17] for distributed solvers of (2.5) with $\Delta = \Delta_\epsilon$.

It is worth noticing that $\mathbf{X}$ depends only on the network and the linearization point, whereas the set $\mathcal{Q}$ and vector $\mathbf{y}$ depend on the variable loads and solar generation, collectively denoted as $\boldsymbol{\chi} := [(\mathbf{p}^c)^\top \ (\mathbf{q}^c)^\top \ (\mathbf{p}^g)^\top]^\top$.

Ideally, the reactive control process entails three steps:

S1) Each bus communicates its (p_n^g, p_n^c, q_n^c) to the operator.

S2) The operator solves (2.5) knowing the current $\boldsymbol{\chi}$.

S3) The operator sends the optimal setpoints $\tilde{\mathbf{q}}^g$ to inverters.

Under variable solar generation, the process has to be repeated on a per-minute basis. Observe that *S1)* establishes N inverter-to-utility communication links, and *S3)* requires another N utility-to-inverter links. Running this process for multiple feeders can become a computationally and communication-wise challenging task.

To adaptively adjust inverter setpoints based on $\boldsymbol{\chi}_t$, affine control rules in the form of $\mathbf{q}^g(\boldsymbol{\chi}_t)$ have been suggested in [19], [20], [21]. Based on these rules, the reactive injection of inverter n is expressed as an affine function over a subvector of $\boldsymbol{\chi}_t$. The premise is to design the rule in a quasi-stationary fashion, but apply it in real-time. We extend linear to *nonlinear* control rules enjoying varying cyber requirements after briefly reviewing the toolbox of kernel-based learning.

Chapter 3

Preliminaries on Kernel-based Learning

Given pairs $\{(z_s, y_s)\}_{s=1}^{S}$ of features z_s belonging to a measurable space $\mathcal{Z}$ and target values $y_s \in \mathbb{R}$, kernel-based learning aims at finding a function or mapping $f : \mathcal{Z} \rightarrow \mathbb{R}$. From all possible options of arbitrarily complex functions, one needs to select a specific family where f belongs. Kernel-based learning postulates that f lies in the function space [28]

$$\mathcal{H}_{\mathcal{K}} := \left\{ f(z) = \sum_{s=1}^{\infty} K(z, z_s) a_s, \;\; a_s \in \mathbb{R} \right\}. \tag{3.1}$$

This is the space of functions that can be expressed as linear combinations of a given kernel (basis) function $K : \mathcal{Z} \times \mathcal{Z} \rightarrow \mathbb{R}$ evaluated at arbitrary points z_s. When $K(\cdot, \cdot)$ is a symmetric positive definite function, then $\mathcal{H}_{\mathcal{K}}$ becomes a reproducing kernel Hilbert space (RKHS) whose members have finite norm $\|f\|_{\mathcal{K}}^2 := \sum_{s=1}^{\infty} \sum_{s'=1}^{\infty} K(z_s, z_{s'}) a_s a_{s'}$; see [29]. Some options for the kernel function K are provided under *Examples 1–2* in chapter 4.1.

Learning f from data $\{(z_s, y_s)\}_{s=1}^{S}$ can be formulated as the regularization task [28], [30]

$$\min_{f \in \mathcal{H}_{\mathcal{K}}, b} \frac{1}{S} \sum_{s=1}^{S} L\left(f(z_s), b; y_s\right) + \mu \|f\|_{\mathcal{K}} \tag{3.2}$$

where b is an intercept term. When it comes to regression, typical choices for the data-fitting loss L include the least-squares (LS) fit $(y_s - f(z_s) - b)^2$, or the ϵ-insensitive loss $[y_s - f(z_s) - b]_\epsilon$. The second term in (3.2) ensures $f \in \mathcal{H}_\mathcal{K}$ and facilitates generalization over unseen data [29]. Parameter $\mu > 0$ balances fitting versus generalization, and is tuned via cross-validation: *i)* problem (3.2) is solved for a specific μ using 4/5 of the data; *ii)* the learned function is validated on the unused 1/5 of the data; *iii)* the process is repeated 5 times to calculate the average fitting error for this μ; and *iv)* the μ attaining the best fit is selected; see [28] for details.

The advantage of confining f to lie in the RKHS $\mathcal{H}_\mathcal{K}$ is that the functional optimization of (3.2) can be equivalently posed as an minimization problem over a finite-dimensional vector: The celebrated Representer's Theorem asserts that the solution to (3.2) admits the form [28]

$$f(z) = \sum_{s=1}^{S} K(z, z_s) a_s. \tag{3.3}$$

In other words, the minimizer of (3.2) is described only by S rather than infinitely many a_s's. Based on (3.3), evaluating $f(z)$ at the given data provides $\mathbf{f} = \mathbf{K}\mathbf{a}$, where $\mathbf{f} := [f(z_1) \ \ldots \ f(z_S)]^\top$; matrix $\mathbf{K} \in \mathbb{S}_{++}^S$ is the *kernel matrix* with entries $[\mathbf{K}]_{s,s'} := K(z_s, z_{s'})$; and $\mathbf{a} := [a_1 \ \ldots \ a_S]^\top$.

From properties of the RKHS's, it holds that $\|f\|_\mathcal{K}^2 = \mathbf{a}^\top \mathbf{K} \mathbf{a}$; see [29]. For regression under an LS loss, the functional minimization in (3.2) becomes the vector optimization

$$\min_{\mathbf{a},b} \ \frac{1}{S} \|\mathbf{y} - \mathbf{K}\mathbf{a} - b\mathbf{1}\|_2^2 + \mu \|\mathbf{K}^{1/2}\mathbf{a}\|_2 \tag{3.4}$$

where $\mathbf{K}^{1/2}$ is the square root of $\mathbf{K}$ and $\mathbf{y} := [y_1 \ \cdots \ y_S]^\top$.

It is worth stressing that (3.3) applies not only to the given data $\{z_s\}_{s=1}^S$, but any $z_{s'} \in \mathcal{Z}$. Evaluating $f(z)$ requires knowing the $(\mathbf{a}, b)$ minimizing (3.4), and being able to evaluate the kernel $K(z, z_s)$ for $s = 1, \ldots, S$. We next use kernel-based learning to develop nonlinear inverter control rules.

Chapter 4

Kernel-based Control Policies

The reactive injection by inverter n is modeled by the rule

$$q_n^g(\mathbf{z}_n) = f_n(\mathbf{z}_n) + b_n \tag{4.1}$$

whose ingredients $(f_n, \mathbf{z}_n, b_n)$ are explained next.

Control inputs: Vector $\mathbf{z}_n \in \mathcal{Z}_n \subseteq \mathbb{R}^{M_n}$ is the input to control rule for inverter n. This vector may include load, solar generation, and/or line flow measurements collected locally or remotely. For a purely local rule, this input can be selected as

$$\mathbf{z}_n := [\bar{q}_n^g \quad (p_n^c - p_n^g) \quad q_n^c]^\top \tag{4.2}$$

where the first entry $\bar{q}_n^g$ relates to the apparent power constraint and has been defined in (2.2). The voltage v_n could also be appended in $\mathbf{z}_n$; however the stability of the resultant control loop is hard to analyze even when f_n is linear; see e.g., [31], [17], [18], [22], [1].

Selecting the controller structure, i.e., the content for each $\mathbf{z}_n$, can affect critically the performance of this control scheme. Ideally, each inverter rule can be fed all uncertain quantities, that is the three numbers in the right-hand side of (4.2) across all buses. In that case, the input vectors $\mathbf{z}_n$ become all equal and of size $3N$. However, this incurs the communication burden of broadcasting $3N$ values in real time. Hybrid setups with $\mathbf{z}_n$'s carrying a combination of local and remote data can be envisioned. To eliminate the effect of this trade-off between communications and performance, this work assumes that the content of $\mathbf{z}_n$'s is prespecified. The task of input selection could be possibly pursued along the lines of sparse linear or polynomial regression [22], [1], [32]; and automatic relevance determination [33, Sec. 6.4].

Control function: Selecting the form of f_n is the second design task. To leverage kernel-based learning, the inverter rule f_n is postulated to lie in the RKHS

$$\mathcal{H}_{\mathcal{K}_n} := \left\{ f_n(\mathbf{z}_n) = \sum_{s=1}^{\infty} K_n(\mathbf{z}_n, \mathbf{z}_{n,s}) a_{n,s},\ \ a_{n,s} \in \mathbb{R} \right\} \tag{4.3}$$

determined by the kernel function $K_n : \mathcal{Z}_n \times \mathcal{Z}_n \to \mathbb{R}$.

Linear rules can be designed by selecting the linear kernel $K_n(\mathbf{z}_{n,s}, \mathbf{z}_{n,s'}) = \mathbf{z}_{n,s}^\top \mathbf{z}_{n,s'}$. *Non-linear* rules can be designed by selecting for example a polynomial kernel $K_n(\mathbf{z}_{n,s}, \mathbf{z}_{n,s'}) = \left(\mathbf{z}_{n,s}^\top \mathbf{z}_{n,s'} + \gamma\right)^\beta$ or a Gaussian kernel $K_n(\mathbf{z}_{n,s}, \mathbf{z}_{n,s'}) = \exp\left(-\|\mathbf{z}_{n,s} - \mathbf{z}_{n,s'}\|_2^2/\gamma\right)$ with design parameters $\beta > 0$ and $\gamma > 0$; see [28].

Intercept $b_n \in \mathbb{R}$: Although it could be incorporated into f_n by augmenting $\mathbf{z}_n$ with a constant entry of 1, it is kept separate to avoid its penalization through $\|f\|_{\mathcal{K}_n}$ [28].

4.1 Learning rules from scenario data

The rules of (4.1) can be learned from scenario data indexed by $s \in \mathcal{S}$ with $\mathcal{S} := \{1, \ldots, S\}$. Scenario s consists of the control inputs $\mathbf{z}_{n,s}$ for $n \in \mathcal{N}$, and the associated vector $\mathbf{y}_s := \mathbf{R}(\mathbf{p}_s^g - \mathbf{p}_s^c) - \mathbf{X}\mathbf{q}_s^c$ defined in (2.4). Evaluating rule n of (4.1) under scenario s yields the inverter response $q_{n,s}^g := q_n^g(\mathbf{z}_{n,s})$. Let us collect the outputs $q_{n,s}^g$ from all inverters into vector $\mathbf{q}_s^g$. Note that the goal is not to fit $\mathbf{y}_s$ by $\mathbf{q}_s^g$, but to minimize the voltage deviations

$\mathbf{X}\mathbf{q}_s^g + \mathbf{y}_s$. The control functions $\{f_n\}_{n=1}^N$ and the intercepts $\{b_n\}_{n=1}^N$ accomplishing this goal can be found via the functional minimization

$$\begin{aligned} \min \quad & \frac{1}{S}\sum_{s=1}^{S} \Delta\left(\mathbf{q}_s^g; \mathbf{y}_s\right) + \mu \sum_{n=1}^{N} \|f_n\|_{\mathcal{K}_n} \\ \text{over} \quad & q_{n,s}^g = f_n(\mathbf{z}_{n,s}) + b_n, \ \forall n, s \\ & \{f_n \in \mathcal{H}_{\mathcal{K}_n}\}, \mathbf{b} := [b_1 \ \cdots \ b_N]^\top \\ \text{s.to} \quad & |q_{n,s}^g| \leq \bar{q}_{n,s}^g, \ \forall n, s \end{aligned} \tag{4.4}$$

where Δ is a voltage regulation objective [cf. (2.6)–(2.7)].

Remark 4.1. The proposed approach is related to [1]–[2], where inverter rules are also trained using machine learning. However, the aforementioned works proceed in two steps: They first solve a sequence of OPF problems similar to (2.5) to find the optimal inverter setpoints $\tilde{\mathbf{q}}^g$ under different scenarios. Secondly, they learn the mapping between controller inputs $\{\mathbf{z}_{n,s}\}_{s\in\mathcal{S}}$ and optimal setpoints $\{\tilde{q}_{n,s}^g\}$ decided by the OPF problems. During this process, they also select which inputs are more effective to be communicated to inverters. The mapping is learned via linear or kernel-based regression. On the other hand, the approach proposed here consolidates the OPF and the learning steps into a single step: The advantage is that the OPF decisions of (4.4) are taken under the explicit practical limitation that q_n^g can only be a function of $\mathbf{z}_n$, since inverter n will not have access to the complete grid conditions. To get some intuition, suppose ones designs linear control rules of known input structure using the single-step approach of (4.4) with $\mu = 0$ and the two-step approach of [1]–[2]. The single-step approach yields rules R_1, and the two-step approach yields rules R_2. Let us evaluate R_1 and R_2 on the training scenarios. Rules R_2 are not necessarily feasible per scenario $s \in \mathcal{S}$, whereas rules R_1 are. Moreover, rules R_2 do not necessarily coincide with the minimizers of (2.5). For the sake of comparison, let us assume that rules R_2 turn out to be feasible per scenario, and hence feasible for (4.4). Being the minimizers of (4.4), rules R_1 attain equal or smaller voltage deviation cost compared to R_2 over the training data. Numerical tests in chapter 6 corroborate the advantage of R_1 over R_2 for $\mu > 0$ and during the operational phase as well.

Different from (3.2), the optimization in (4.4) entails learning multiple functions (one per inverter). Since inverter injections affect voltages feeder-wise, inverter rules are naturally coupled through Δ in (4.4). Similar multi-function setups can be found in collaborative filtering or multi-task learning [29], [34].

Fortunately, Representer's Theorem can be applied successively over n in (4.4). Therefore, each rule n is written as

$$f_n(\mathbf{z}_n) = \sum_{s=1}^{S} K_n(\mathbf{z}_n, \mathbf{z}_{n,s}) a_{n,s}. \tag{4.5}$$

Once the coefficients $\{a_{n,s}\}$ have been found, rule $\{f_n\}$ can be evaluated for any $\mathbf{z}_n$. Similar to (3.3), evaluating rule f_n over the scenario data $\{\mathbf{z}_{n,s}\}_{s=1}^{S}$ gives

$$\mathbf{f}_n = \mathbf{K}_n \mathbf{a}_n, \quad \forall n \tag{4.6}$$

where $[\mathbf{K}_n]_{s,s'} = K_n(\mathbf{z}_{n,s}, \mathbf{z}_{n,s'})$ for $s, s' = 1, \ldots, S$, and $\mathbf{a}_n := [a_{n,1} \ \cdots \ a_{n,S}]^\top$. The RKHS norms can be written as

$$\|f_n\|_{\mathcal{K}_n} = \sqrt{\mathbf{a}_n^\top \mathbf{K}_n \mathbf{a}_n}, \quad \forall n. \tag{4.7}$$

In this way, the functional minimization in (4.4) is cast as a vector minimization over $\{\mathbf{a}_n\}_{n=1}^{N}$ and $\mathbf{b}$. The exact form of this minimization and its properties for different Δ are discussed later in chapter 5. For now, let us clarify how the kernel functions $K_n(\cdot, \cdot)$ effect different rule forms.

Example 1: Affine rules. The linear kernel $K_n(\mathbf{z}_{n,s}, \mathbf{z}_{n,s'}) = \mathbf{z}_{n,s}^\top \mathbf{z}_{n,s'}$ yields affine rules. The sought functions can be written as

$$f_n(\mathbf{z}_n) = \mathbf{z}_n^\top \mathbf{w}_n, \qquad \forall n. \tag{4.8}$$

Given scenario data $\mathbf{z}_{n,s}$ and $\mathbf{y}_s$ for $n \in \mathcal{N}$ and $s \in \mathcal{S}$, we would like to find $\{\mathbf{w}_n, b_n\}_n$ through (4.4). Collect the input data for inverter n in the $M_n \times S$ matrix $\mathbf{Z}_n := [\mathbf{z}_{n,1} \ \cdots \ \mathbf{z}_{n,S}]$. According to Representer's Theorem, the optimal $\mathbf{w}_n$ can be expressed as $\mathbf{w}_n = \mathbf{Z}_n \mathbf{a}_n$ for some $\mathbf{a}_n$. Evaluating the control rule for any input $\mathbf{z}_{n,s}$ yields

$$q_n(\mathbf{z}_{n,s}) = f_n(\mathbf{z}_{n,s}) + b_n = \mathbf{z}_{n,s}^\top \mathbf{Z}_n \mathbf{a}_n + b_n.$$

Evaluating the rule at the input data yields (4.6) with $\mathbf{K}_n = \mathbf{Z}_n^\top \mathbf{Z}_n$. The squared function norm is $\|f_n\|_{\mathcal{K}_n}^2 = \|\mathbf{w}_n\|_2^2 = \mathbf{a}_n^\top \mathbf{Z}_n^\top \mathbf{Z}_n \mathbf{a}_n = \mathbf{a}_n^\top \mathbf{K}_n \mathbf{a}_n$.

Example 2: Non-linear rules. For non-linear rules, transform the input $\mathbf{z}_{n,s}$ to vector $\boldsymbol{\phi}_{n,s} := \phi_n(\mathbf{z}_{n,s})$ via a non-linear mapping $\phi_n : \mathbb{R}^{M_n} \to \mathbb{R}^{\Phi_n}$. The entries of $\boldsymbol{\phi}_{n,s}$ could be for example all the first- and second-order monomials formed by the entries of $\mathbf{z}_{n,s}$. The dimension Φ_n of $\boldsymbol{\phi}_{n,s}$ can be finite (e.g., polynomial kernels) or infinite (Gaussian kernels) [33]. Then, the control function

$$f_n(\mathbf{z}_n) = \boldsymbol{\phi}_n^\top \mathbf{w}_n \tag{4.9}$$

with $\mathbf{w}_n \in \mathbb{R}^{\Phi_n}$ is *non-linear* in $\mathbf{z}_n$. The developments of Example 1 carry over to Example 2 by using $\mathbf{K}_n = {}_n^\top \, {}_n$ and replacing $\mathbf{Z}_n$ by ${}_n := [\boldsymbol{\phi}_{n,1} \ \cdots \ \boldsymbol{\phi}_{n,S}]$. Depending on the mapping ϕ_n, the vectors $\boldsymbol{\phi}_{n,s}$ may be of finite or infinite length [28]. The critical point is that f_n does not depend on $\boldsymbol{\phi}_{n,s}$'s directly, but only on their inner products $\boldsymbol{\phi}_{n,s}^\top \boldsymbol{\phi}_{n,s'}$ for any s and s'. These products can be easily calculated through the kernel function as $\boldsymbol{\phi}_{n,s}^\top \boldsymbol{\phi}_{n,s'} = K_n(\mathbf{z}_{n,s}, \mathbf{z}_{n,s'})$; see [28].

Since the constraints in (4.4) are enforced for the scenario data, the learned rules do not necessarily satisfy these constraints for all $\mathbf{z}_{n,s}$ with $s \notin \{1, \ldots, S\}$. This limitation appears also in scenario-based and chance-constrained designs [20]. Once a control rule is learned, in real-time t, it can be heuristically projected within $[-\bar{q}^g_{n,t}, +\bar{q}^g_{n,t}]$ as

$$\mathcal{P}_{\bar{q}^g_{n,t}}\left[q^g_{n,t}\right] := \max\left\{\min\left\{q^g_{n,t}, \bar{q}^g_{n,t}\right\}, -\bar{q}^g_{n,t}\right\}.$$

4.2 Implementing reactive control rules

Our control scheme involves four steps; see also Fig. 4.1:

T1) The utility collects scenario data $\mathbf{z}_{n,s}$ for all n and s.

T2) The utility designs rules by solving (4.4); see chapter 5.

T3) Each inverter n receives $S+1$ data $(\mathbf{a}_n, b_n)$ from the utility, which describe f_n.

T4) Over the next 30 minutes and at real time t, each inverter n will be collecting $\mathbf{z}_{n,t'}$ and applying the rule

$$\mathcal{P}_{\bar{q}^g_{n,t'}}\left[\sum_{s=1}^{S} K_n(\mathbf{z}_{n,t'}, \mathbf{z}_{n,s})a_{n,s} + b_n\right]. \tag{4.10}$$

The aforesaid process is explicated next. Regarding *T1)*, scenario data should be as representative as possible for the grid conditions anticipated over the following 30-min control period. One option would be to use load and solar generation forecasts. A second option would be to use historical data from the previous day and same time, if they representative of today's conditions. A third alternative would be to use the most recent grid conditions known to the utility. For example, if smart meter data are collected every 30 min anyway, they can be used in lieu of forecasts for the next control period.

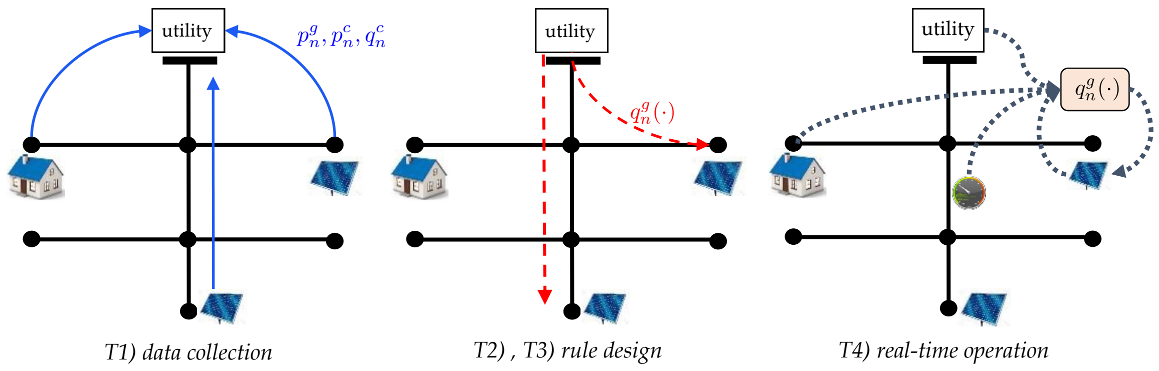

Figure 4.1: Implementing reactive power control rules. *Left:* Data are collected from buses. *Center*: utility designs rules and downloads rules to inverters. *Right:* Inverters follow control rules fed by local and/or remote data.

The numerical tests of chapter 6 adopt the third option and use the minute-based grid conditions observed over the last 30-minutes as $S = 30$ scenarios to train the inverter rules for the upcoming 30-minute interval. Obviously, the number of training scenarios S does not have to coincide with the length of the control period measured in minutes. These two parameters relate to loading conditions; feeder details; availability and quality of scenario data; communication and computational resources. Selecting their optimal values goes beyond the scope of this work.

During *T4)*, inverter n has already received $(\mathbf{a}_n, b_n)$ and $\{\mathbf{z}_{n,s}\}_{s=1}^S$ during *T3)*. Each $\mathbf{z}_n$ may consist of local data and a few active flow readings collected from major lines or transformers. If the entries of $\mathbf{z}_n$ are all local, the rule can be applied with no communication. Otherwise, the non-local entries of $\mathbf{z}_n$ have to be sent to inverter n. If non-local inputs are shared among inverters, broadcasting protocols can reduce the communication overhead.

Remark 4.2. Suppose each inverter n knows the training data $\mathbf{z}_{n,s}$ for $s \in \mathcal{S}$. Function f_n can be described in two ways: Either through (4.5) using the data described under T3); or through (4.8)–(4.9) via $\mathbf{w}_n$. For the second way, vector $\mathbf{w}_n$ has M_n entries in the linear case and Φ_n entries in the nonlinear case. For the linear case, if $M_n < S + 1$, representing f_n

through (4.8) by $\mathbf{w}_n$ is more parsimonious. Representation (4.5) becomes advantageous only when $\Phi_n \gg S + 1$ under the nonlinear case.

Chapter 5

Support Vector Reactive Power Control

This chapter converts (4.4) to a vector minimization and explores different options for Δ. From (4.6), the output of inverter n across all S scenarios is $\mathbf{K}_n\mathbf{a}_n + b_n\mathbf{1}$. Then, the apparent power constraints in (4.4) can be written as

$$-\bar{\mathbf{q}}_n^g \leq \mathbf{K}_n\mathbf{a}_n + b_n\mathbf{1} \leq \bar{\mathbf{q}}_n^g,\ \forall n \tag{5.1}$$

where $\bar{\mathbf{q}}_n^g := [\bar{q}_{n,1}^g \ \cdots \ \bar{q}_{n,S}^g]^\top$. Moreover, the vector of voltage deviations can be expressed as

$$\begin{aligned}\mathbf{X}\mathbf{q}_s^g + \mathbf{y}_s &= \mathbf{X}\left(\sum_{n=1}^{N}\mathbf{e}_n q_{n,s}^g\right) + \mathbf{y}_s \\ &= \sum_{n=1}^{N}\mathbf{x}_n\mathbf{e}_s^\top\mathbf{K}_n\mathbf{a}_n + \sum_{n=1}^{N} b_n\mathbf{x}_n + \mathbf{y}_s\end{aligned} \tag{5.2}$$

where $\mathbf{x}_n$ is the n-th column of $\mathbf{X}$. Substituting (4.7) and (5.1)–(5.2), the optimization in (4.4) can be posed as a second-order cone program (SOCP) over $\{\mathbf{a}_n\}_{n\in\mathcal{N}}$ and $\mathbf{b}$.

Nonetheless, solving (4.4) with $\Delta = \Delta_\epsilon$ yields optimal $\mathbf{a}_n$'s with several non-zero entries. This means that to describe rule n by (4.10), the utility needs to communicate the entire vector $\mathbf{a}_n$ during *T3)*. If scenarios $\{\mathbf{z}_{n,t}\}_{t=1}^T$ are not known by the inverter, they have to be communicated along with $\mathbf{a}_n$ as well. The number of scenarios T may be large when learning rules under complex feeder setups. A related approach for minimizing a convex combination of Δ_s and power losses has been suggested in the conference precursor of this work [23, 24], but inherits the same difficulty of non-sparse $a_{n,t}$'s. As commented earlier, one of the contributions of this paper is designing parsimonious reactive power representations which helps reduce the communication overhead.

Inspired by support vector machines (SVM), we engineer Δ to obtain inverter rules described by possibly fewer scenarios: Promoting sparse $\mathbf{a}_n$'s alleviates the communication overhead during step *T3)*. To this end, we put forth the cost

$$\Delta_\tau(\mathbf{q}^g;\mathbf{y}) := [\|\mathbf{X}\mathbf{q}^g + \mathbf{y}\|_2]_\tau \tag{5.3}$$

for some $\tau > 0$. If scenario s yields a vector of voltage deviations $\mathbf{X}\mathbf{q}_s^g + \mathbf{y}_s$ with ℓ_2-norm smaller than τ, this scenario incurs no cost. If $\|\mathbf{X}\mathbf{q}_s^g + \mathbf{y}_s\|_2 > \tau$, the voltage regulation penalty grows with $\|\mathbf{X}\mathbf{q}_s^g + \mathbf{y}_s\|_2$. The cost in (5.3) can be expressed as an SOCP over the slack variable d

$$\Delta_\tau(\mathbf{q}^g;\mathbf{y}) := \min_{d\geq 0} \left\{d : \|\mathbf{X}\mathbf{q}^g + \mathbf{y}\|_2 \leq d + \tau\right\}.$$

Applying the same epigraph trick for the function norms, problem (4.4) can be solved as the

SOCP

$$
\begin{aligned}
\min \quad & \frac{1}{S}\mathbf{d}^\top \mathbf{1} + \mu \boldsymbol{\gamma}^\top \mathbf{1} && \text{(5.4a)}\\
\text{over} \quad & \{\mathbf{q}_s^g\}, \{\mathbf{a}_n\}, \mathbf{b}, \mathbf{d} \geq \mathbf{0}, \boldsymbol{\gamma} && \text{(5.4b)}\\
\text{s.to} \quad & (5.1), (5.2) && \text{(5.4c)}\\
& \|\mathbf{K}_n^{1/2}\mathbf{a}_n\|_2 \leq \gamma_n, \quad \forall n && \text{(5.4d)}\\
& \|\mathbf{X}\mathbf{q}_s^g + \mathbf{y}_s\|_2 \leq d_s + \tau, \quad \forall s && \text{(5.4e)}
\end{aligned}
$$

where $\mathbf{d} := [d_1 \ \cdots \ d_S]^\top$ and $\boldsymbol{\gamma} := [\gamma_1 \ \cdots \ \gamma_N]^\top$. The variables $\mathbf{q}_s^g$ can be eliminated using the substitutions of (5.2). Solving (5.4) takes $\mathcal{O}\left(N^{3.5}T^3\right)$ operations with interior point-based solvers [35]. However, the advantage of inverter control rules is that (5.4) is not solved in real time. If standard interior point-based solvers are not scalable to larger grids, one may resort to (distributed) first-order algorithms; warm-start initializations; and cutting-plane methods.

The coefficients $\mathbf{a}_n$'s minimizing (5.4) enjoy two types of sparsity, across inverters and across scenarios. To explain the first type of sparsity, express the second summand in the cost of (5.4) as $\mu\boldsymbol{\gamma}^\top\mathbf{1} = \mu\sum_{n=1}^N \|\mathbf{K}_n^{1/2}\mathbf{a}_n\|_2$. Having these non-squared ℓ_2-norms in the objective promotes block sparsity across n, in the sense that for larger μ, some vectors $\mathbf{a}_n$ may be set to zero. This effect is a direct consequence of block-sparse solutions encountered in group Lasso (G-Lasso)-formulations; see [36], [30], [34]. All inverters receive a reactive power setpoint b_n, but if the optimal $\mathbf{a}_n$ becomes zero, inverter n will not be changing its reactive injection in real-time. One may drop the intercept b_n from the control rule of (4.1) and the optimization of (5.4), and modify the feature vector as

$$\mathbf{z}_n' = [1 \ \mathbf{z}_n^\top]^\top. \tag{5.5}$$

Thus, obtaining $\mathbf{a}_n = \mathbf{0}$ from (5.4) enables inverter selection.

The next proposition studies the second type of sparsity; see the appendix for a proof.

Proposition 5.1. *Consider (4.4) with $\Delta = \Delta_\tau$ and its minimizer in (4.5). If $\|\mathbf{X}\mathbf{q}_s^g + \mathbf{y}_s\|_2 < \tau$ for scenario s at the optimum, then $a_{n,s} = 0$ for every inverter n with $|q_{n,s}^g| < \overline{q}_{n,s}^g$.*

Proposition 5.1 explains how Δ_τ promotes block sparsity across s: If scenario s does not experience severe voltage violations, the corresponding coefficients $a_{n,s}$ will be zero for all inverters n that have not reached their apparent power limit. Block sparsity across time identifies non-critical scenarios. Phrased in the SVM context, the so-termed 'support vectors' here correspond to scenarios with significant voltage deviations. Larger values of τ effect fewer critical scenarios.

These two forms of sparsity offer communication savings since the related $(a_{n,s}, \mathbf{z}_{n,s})$ do not need to be communicated to inverters. This enables training the rules for larger number of scenarios S at the same communication overhead. Note that for fixed (μ, τ), the sparsity of $\mathbf{a}_n$'s depends on the training data $\mathbf{y}_s$'s as well. If a particular sparsity goal is to be met, the utility has to solve (5.4) repeatedly for various values of μ and τ. Such computations can be significantly sped up by initializing an optimization algorithm for one value of τ to the minimizer obtained using the previous value of τ [28, Sec. 18.4]; however, such techniques will not be pursued here.

Different from Δ_τ, cost Δ_ϵ is not expected to yield as sparse $\mathbf{a}_n$'s. The next claim (proved in the appendix) explains that even if a single bus experiences voltage deviation larger than ϵ for scenario s, then $a_{n,s} \neq 0$ for all n. In other words, a voltage violation at a single bus for scenario s renders this scenario critical for *all* inverter rules.

Proposition 5.2. *Consider (4.4) with $\Delta = \Delta_\epsilon$ and its minimizer in (4.5). If $\|\mathbf{X}\mathbf{q}_s^g + \mathbf{y}_s\|_\infty > \epsilon$ for scenario s at the optimum, then $a_{n,s} \neq 0$ for all n.*

Chapter 6

Numerical Tests

The novel inverter rules were tested on the IEEE 123-bus feeder [37], converted to a single-phase grid as described in [38]. Residential load and solar data were extracted from the Pecan Street dataset as delineated next [4]. Minute-sampled active load and solar generation data were collected for June 1, 2013 between 8:00–16:00. We downloaded data from the first 123 Pecan Street nodes, after excluding nodes with empty data records. Regarding solar generation, unless stated otherwise, 75% of the buses had solar generation by excluding nodes with bus indexes that are multiples of 4.

Load data were scaled on a per bus basis so that their daily peak values matched 150% of the benchmark load. Since the Pecan Street data included only active power, we drew lagging power factors uniformly at random within $[0.9, 0.95]$ for each bus and kept them fixed across time. The scaling factors for active loads were also used for scaling solar data. To allow for reactive power compensation even at peak solar irradiance, inverters were over-sized by 10% providing an apparent power capacity of $\bar{s}_n^g = 1.1\bar{p}_n^g$ for all n; see [3].

Our numerical tests included six control schemes:

C1) The optimal reactive injections computed by (2.5) on a per-minute basis;

C2) The optimal reactive injections computed by (2.5) on a per-minute basis assuming a 2-minute communication delay;

C3) The fixed Watt-VAR control rules of [3, (12)–(14)];

C4) The rules of (4.4) for linear kernels and $\Delta = \Delta_\tau$;

Figure 6.1: The IEEE 123-bus feeder benchmark.

Table 6.1: Running Time for Solving (4.4) with $T = 30$

	C4)	*C5)*	*C6)*	*C7)*
Running time [min]	0.21	0.45	0.96	1.99

C5) The rules of (4.4) for Gaussian kernels and $\Delta = \Delta_\tau$;

C6) The rules of (4.4) for linear kernels and $\Delta = \Delta_\epsilon$; and

C7) The rules of (4.4) for Gaussian kernels and $\Delta = \Delta_\epsilon$.

The input $\mathbf{z}_n$ to inverter n consisted of local data as in (4.2). Each entry of $\mathbf{z}_n$ was centered by its daily mean and normalized by its daily standard deviation. To avoid rank deficiency, we added $10^{-3} \cdot \mathbf{I}_S$ to all kernel matrices.

Schemes *C1),C2)* were solved using SDPT3 and YALMIP with MATLAB [39, 40]. Schemes *C4)–C7)* were solved by invoking the MOSEK solver directly through MATLAB [41]. Tests were run on a 2.4 GHz Intel Core i5 laptop with 8 GB RAM. The average running time for solving (4.4) with $T = 30$ is given in Table 6.1. It should be emphasized that although the control rules were designed using the LDF grid model, the voltage deviations experienced by all control rules were tested using the full AC model.

During training, we used $T = 30$ scenarios to learn the SVM-based control rules of *C4)–C7)*. These scenarios comprised the load and solar data observed during the last 30 minutes. During validation, the inverter control rules were tested over the following 30 minutes. Parameters μ and γ were selected via 5-fold cross-validation. The ranges of τ and ϵ were empirically chosen to yield an average communication overhead similar to the one needed by the affine rule of (4.8) as discussed under Remark 4.2: An affine rule is described by $M_n + 1 = 4$ data per inverter. If only 10% of the entries of $\mathbf{a}_n$ are nonzero, then communicating $(\mathbf{a}_n, b_n)$ entails sending $0.1 \cdot S + 1 = 0.1 \cdot 30 + 1 = 4$ data as well. The sparsity of $\mathbf{a}_n$'s depends on input data along with the values of (τ, μ) or (ϵ, μ). These parameters were set so that $\mathbf{a}_n$'s had 10% nonzero entries on the average across time and buses.

We next explored the trade-off between voltage deviation and the sparsity of $\mathbf{a}_n$'s for *C4)*–*C7)*. The expectations from this test were two: *i)* voltage deviations are expected to increase for sparser $\mathbf{a}_n$'s; *ii)* schemes *C4)* and *C5)* should exhibit improved sparsity over *C6)* and *C7)*. To validate these hypotheses, we recorded the voltage deviations for 10 values of τ and ϵ for *C4)*–*C7)*. The average absolute voltage deviation and the average percentage of non-zero coefficients were calculated over the day and across buses, and are shown in Figure 6.2. From Figure 6.2, the value of τ yielding a sparsity of roughly 11% is $\tau = 0.001$. Figure 6.2 reveals three important points. First, voltage deviations increase as $\mathbf{a}_n$'s become sparser as expected. Second, for a given sparsity in $\mathbf{a}_n$'s, the rules obtained by Δ_τ exhibit smaller voltage deviations compared to the rules obtained by Δ_ϵ. Because of this, we focus on the performance of *C4)*–*C5)* for the rest of this chapter. Third, the Gaussian kernel-based rules attained lower voltage deviations than the related linear kernel-based rules.

We next tested the effect of μ on inverter selection and voltages. Larger values of μ are expected to set more $\mathbf{a}_n$'s to zero. To eliminate the inverters with $\mathbf{a}_n = \mathbf{0}$, the parameter b_n was appended in $\mathbf{a}_n$ as delineated in (5.5). For a fixed value of $\tau = 0.001$, for scheme *C4)*, the values of μ were obtained using cross-validation across the day. The control rules were designed again using 4 different values of μ. As expected, by increasing the value of μ, the number of all-zero $\mathbf{a}_n$'s and the corresponding voltage deviations were increased. Figure 6.3 depicts the absolute voltage deviation averaged over time for each inverter. Notice that the values of τ and μ were kept fixed, although the training data $\mathbf{y}_s$'s varied across the day. Due to this, the reported sparsity in Figure 6.2 is the average sparsity across time and inverters. Moreover, the number of inverters in Figure 6.3 is the average number of activated inverters

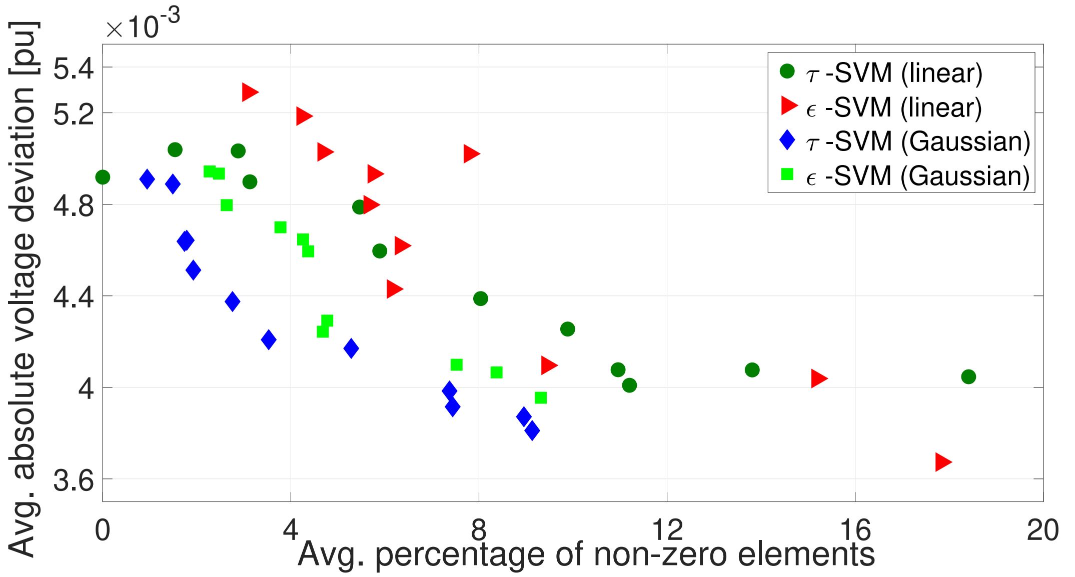

Figure 6.2: Average of absolute voltage deviation vs. sparsity for *C3)–C6)*.

across the day. Even though the values of μ and τ can be adjusted on a 30-min basis to meet specific sparsity requirements, we chose to keep them fixed to simplify the exposition. In fact, the rest of this chapter reports the worst-case instead of average voltage deviations across time and for each bus.

We next compared the proposed SVM-based control rules against the alternative schemes of *C1)–C3)*. To this end, voltage deviations were calculated between 8:00–16:00 for schemes *C1)–C5)*. Figures 6.4 and 6.5 demonstrate the average and the maximum voltage deviations over the test period. It can be observed from both figures that the Gaussian SVM-based rule performs better than *C1)–C3)* due to its ability to capture non-linear behaviors. Although *C3)* needs no communication, it violates the ANSI-C.84.1 standard voltage constraints. Furthermore, despite the high communication needed, scheme *C2)* shows no superiority in performance over *C5)* and corroborates the need for real-time response to system inputs.

In all previous tests, the rules were fed with locally recorded data. To evaluate the advantage of adding remote control inputs, we appended the values of active power flows on the lines

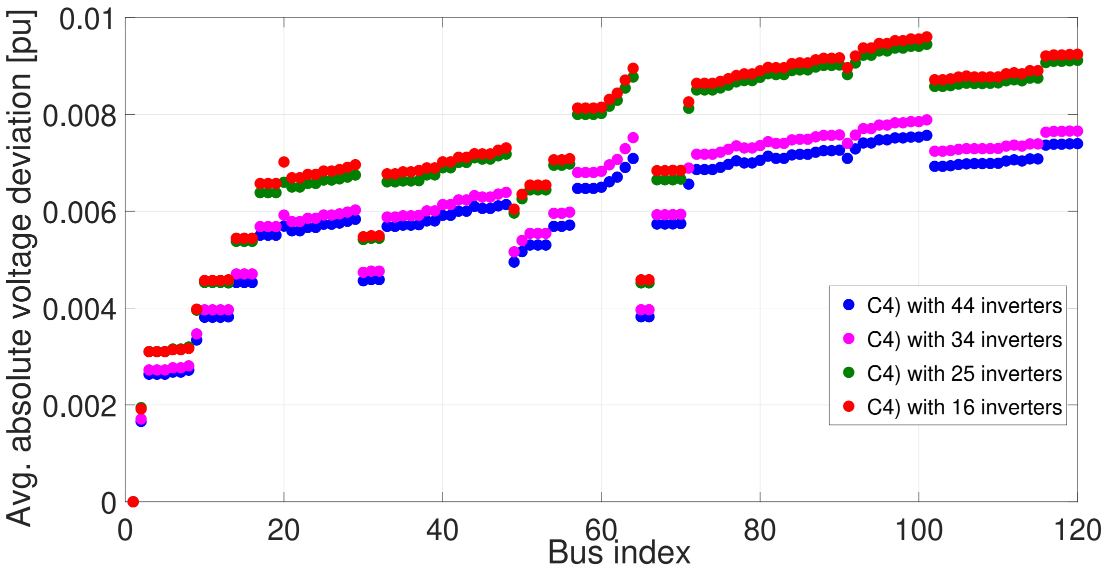

Figure 6.3: Maximum absolute voltage deviation over time for 75% penetration, obtained by the Gaussian SVM-based rules trained for $\Delta = \Delta_\tau$.

feeding buses 1, 16, and 51, to all input vectors $\mathbf{z}_n$. The daily maximum and the average voltage deviations attained by *C1)–C5)* are depicted in Figures 6.6 and 6.7, respectively. As expected, the results suggest that adding remote inputs to the rules improves the grid voltage profile at the expense of increased inter-network communication.

As mentioned in chapter 4.2, the length of the control period (in minutes) over which rules remain constant does not have to agree with the number of scenarios S used for training the rules. To evaluate how the control rules perform for longer control periods, Figure 6.8 compares the voltage deviations obtained by training rules using $S = 30$ scenarios, but keeping them unaltered over 30, 45, and 60 minutes. As expected, voltage regulation deteriorates as rules remain unchanged for longer periods.

All previous tests assumed solar penetration of 75%. We also tested the performance of *C1)–C5)* under penetrations of 50% and 25%. To simulate 50% penetration, solar generation and smart inverters were installed only in buses with even indexes. Likewise, to simulate 25% penetration, we considered buses whose indexes were multiples of 4. Figures 6.9 and 6.10

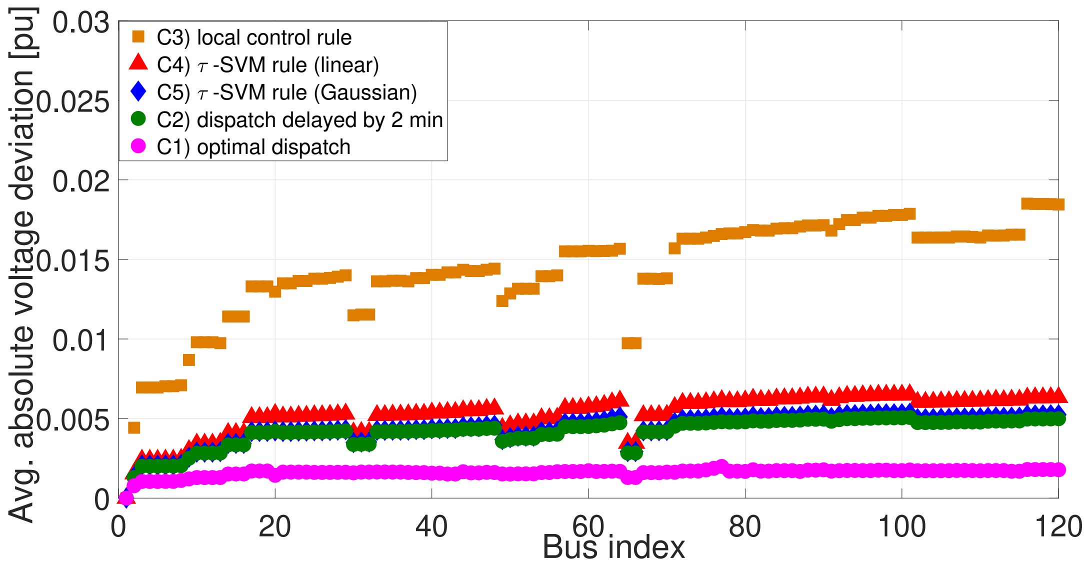

Figure 6.4: Absolute voltage deviation averaged over time for 75% penetration, obtained by the SVM-based rules trained for $\Delta = \Delta_\tau$.

depict the attained maximum absolute voltage deviations, which apparently decrease with decreasing solar penetration. For lower penetrations, the Gaussian-based rule preserves its superior voltage profile over the other schemes.

Schemes *C4)* and *C5)* were also tested under less communication by scaling down the sparsity in $\mathbf{a}_n$'s by a factor of 10: Voltage deviations were evaluated for $\tau = 0.03$ corresponding to 1.4% non-zero entries for $\mathbf{a}_n$'s on the average. Figure 6.11 demonstrates the maximum absolute voltage deviation for *C1)–C5)*. Even with fewer coefficients communicated, the voltage constraints of ANSI-C.84.1 were still satisfied.

The last set of numerical tests compares the developed single-step approach with the two-step approach of [1]–[2]; see also Remark 4.1. We used both approaches to design local linear control rules for the IEEE 13-bus feeder [27], under the voltage deviation cost $\Delta = \Delta_\epsilon$ with $\epsilon = 0.001$. The top and center panels of Fig. 6.12 show respectively the maximum and average voltage deviation per bus computed across time. The bottom panel shows the voltage deviation cost Δ_ϵ, time-averaged per control period. The bottom panel also shows

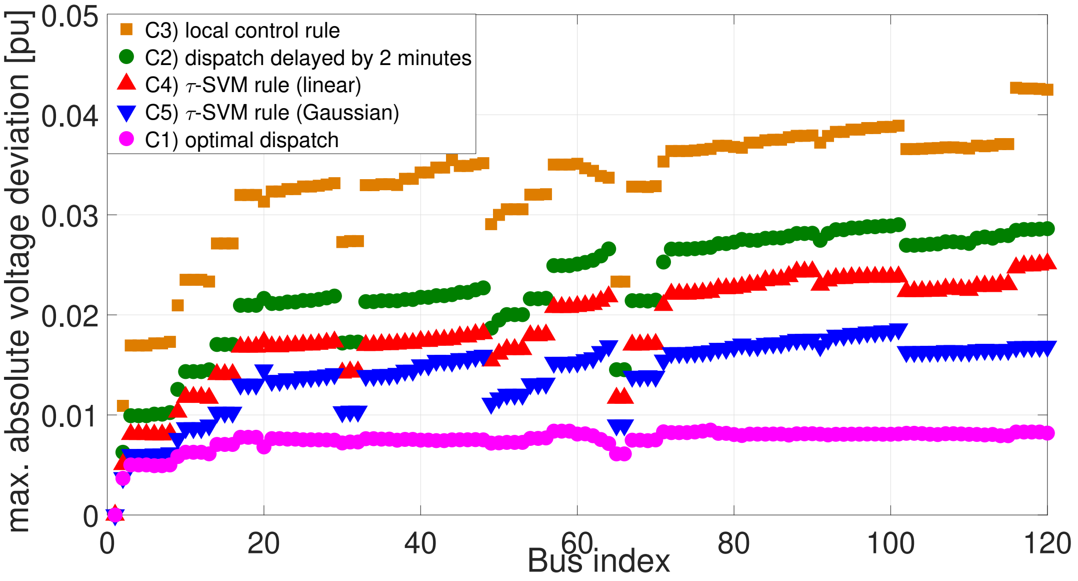

Figure 6.5: Maximum of absolute voltage deviation over time for 75% penetration, obtained by the SVM-based rules trained for $\Delta = \Delta_\tau$.

the voltage deviation cost Δ_τ with $\tau = 0.01$ attained upon training both rules using Δ_τ instead of Δ_ϵ. Similar results were obtained for other values of ϵ and τ. According to these tests, the single-step approach achieved: *1)* lower maximum per-bus voltage deviations; *2)* lower average per-bus voltage deviations; and *3)* smaller voltage deviation costs during the operational phase.

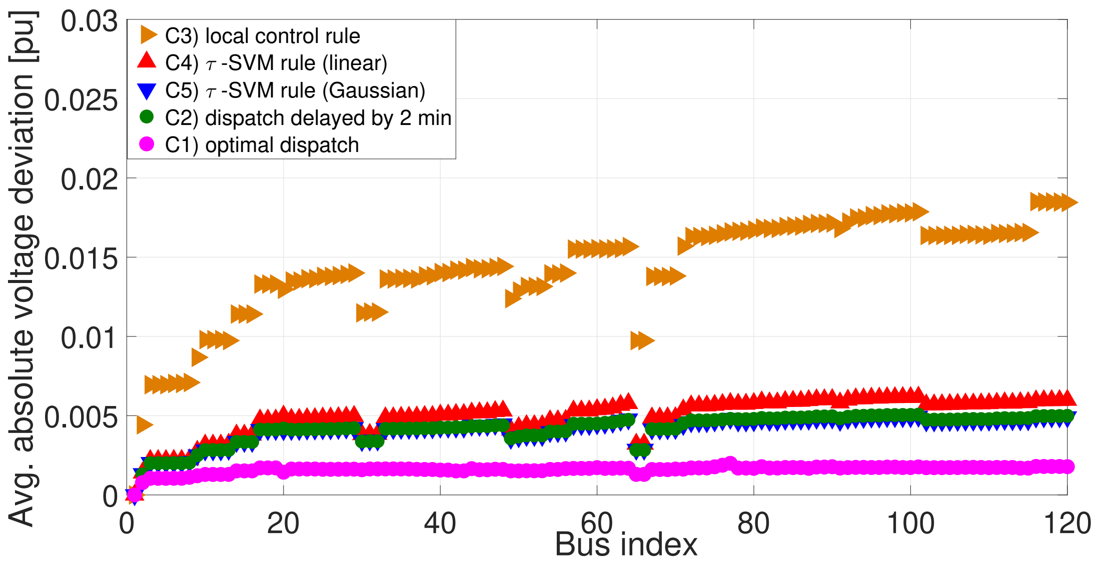

Figure 6.6: Absolute voltage deviation averaged over time for 75% penetration with remote inputs, and the SVM-based rules trained for Δ_τ.

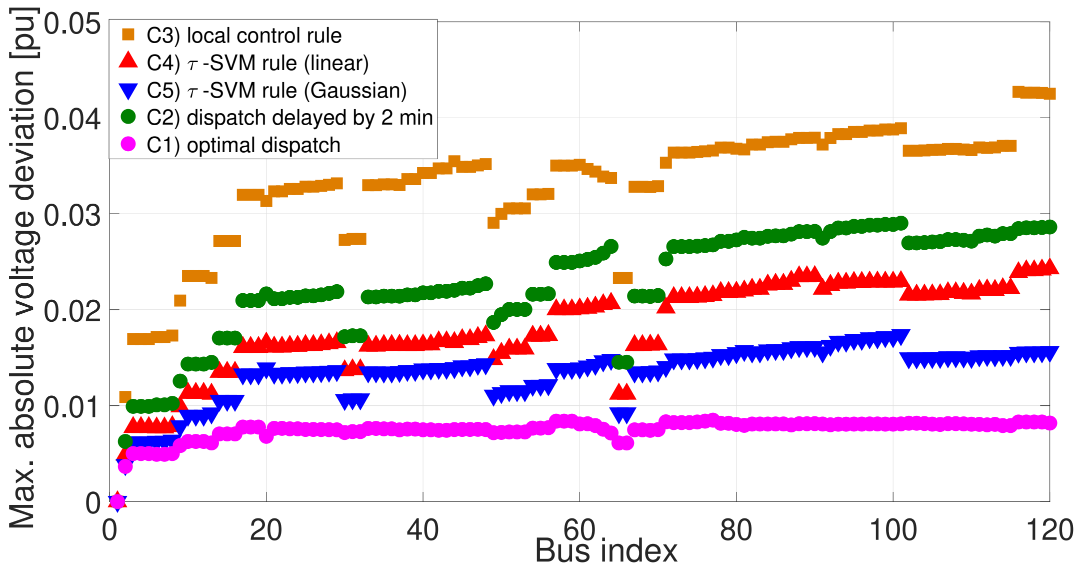

Figure 6.7: Maximum absolute voltage deviation over time for 75% penetration obtained with remote inputs and the SVM-based rules trained for $\Delta = \Delta_\tau$.

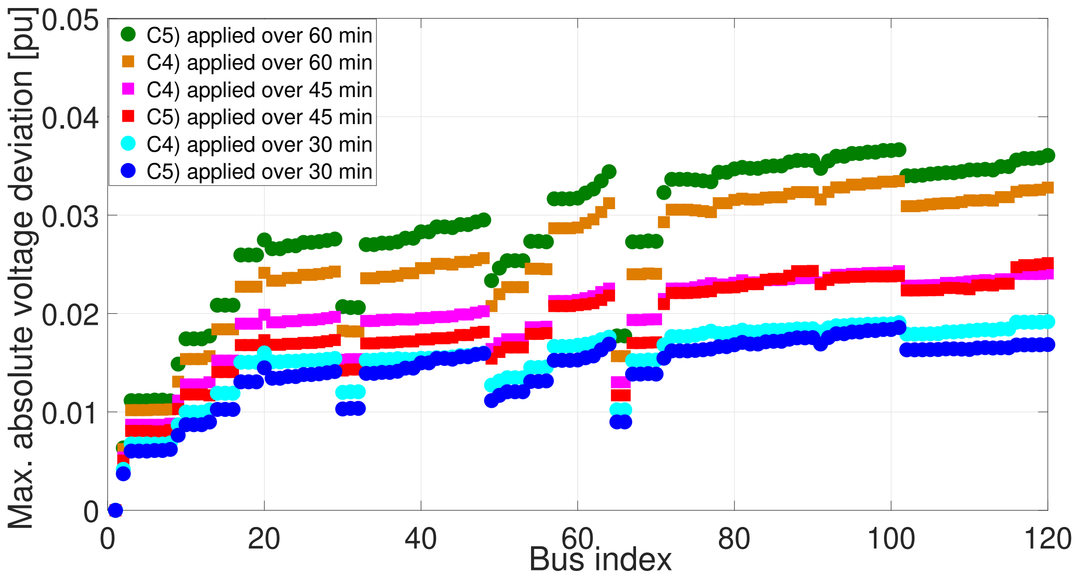

Figure 6.8: Maximum absolute voltage deviation over time for 75% penetration obtained using the SVM-based rules applied over 30, 45, and 60 minutes.

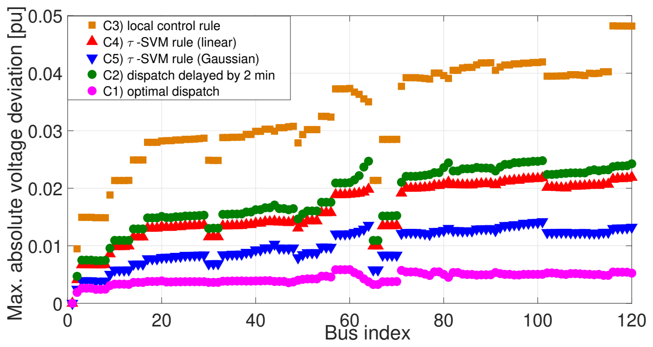

Figure 6.9: Maximum absolute voltage deviation over time for 50% penetration obtained using the SVM-based rules trained for $\Delta = \Delta_\tau$.

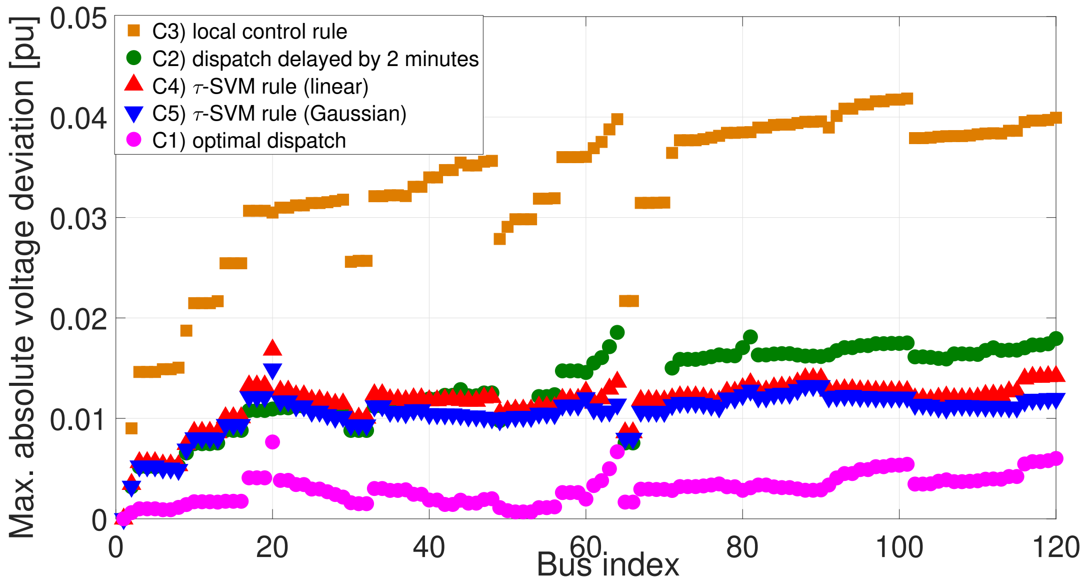

Figure 6.10: Maximum absolute voltage deviation over time for 25% penetration obtained using the SVM-based rules trained for $\Delta = \Delta_\tau$.

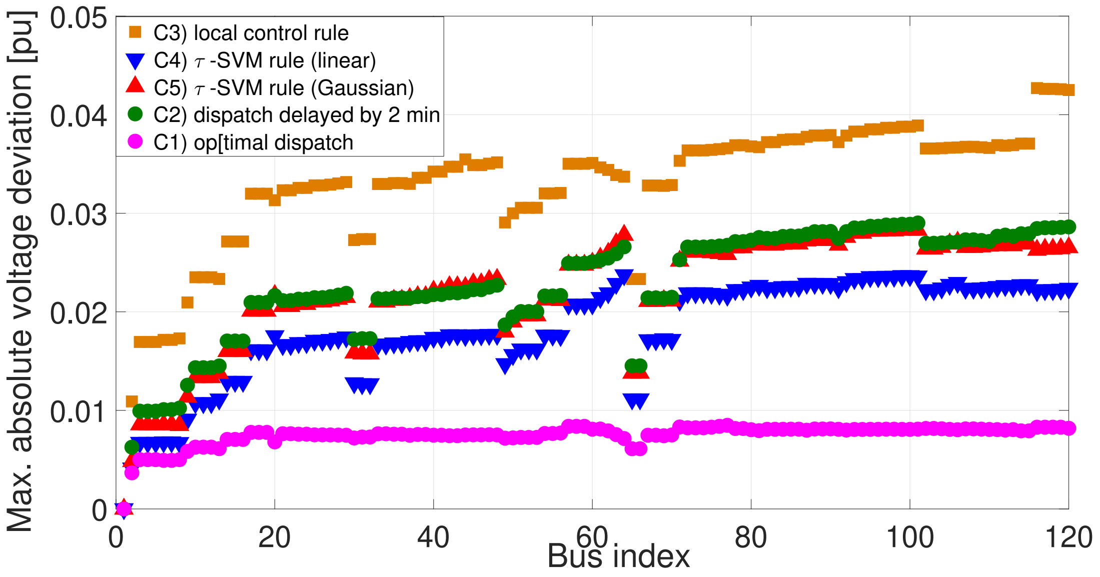

Figure 6.11: Max. absolute voltage deviation over time for 75% penetration obtained using SVM-based rules trained for $\Delta = \Delta_\tau$ with $\tau = 0.03$.

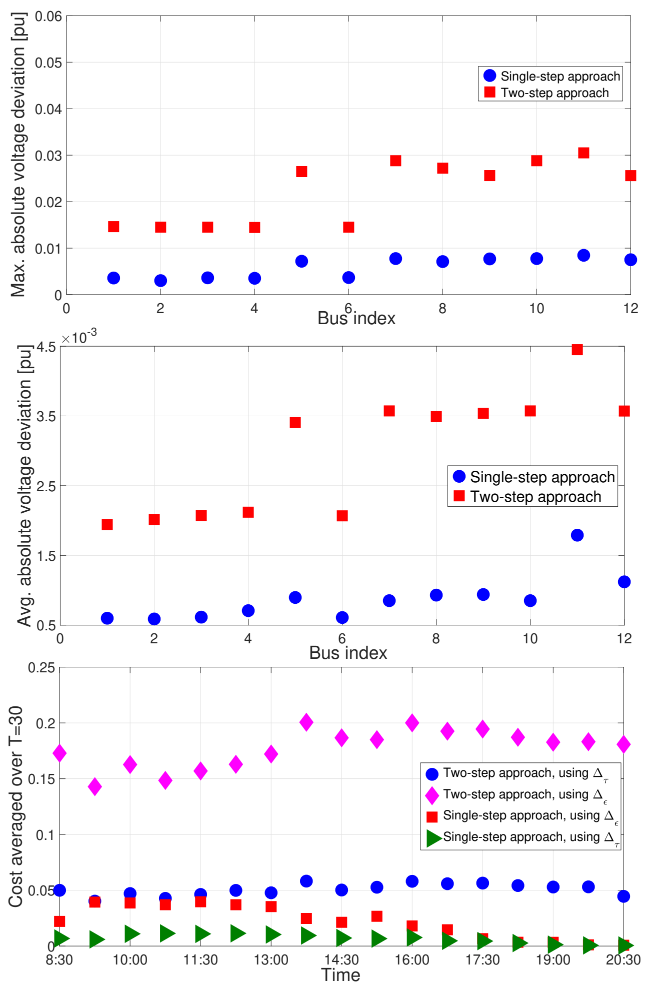

Figure 6.12: Comparison between the proposed single-step learning approach (rules R_1), and the two-step learning approach of [1]–[2] (rules R_2).

Figure 6.13: Voltage deviation costs time-averaged over each 30-min control interval.

Chapter 7

Conclusions

This thesis has put forth a novel approach for designing inverter control rules has been put forth. It relies on both data-based learning and physical grid modeling. Inverter rules are not learned independently using input/output pairs of the OPF problem. Instead, they are learned jointly by posing the related OPF problem as a multi-function learning task. Because of the way voltage deviations couple inverter outputs, the conventional support vector machine approach fails to yield sparse rule descriptions. We have engineered a voltage deviation cost to identify 'support scenarios,' that is a few scenarios with non-zero coefficients for most of inverter rules. The devised control rules were tested using on a benchmark feeder using the exact AC model. The novel scheme attained superior voltage regulation performance compared to preset local rules, and oftentimes comparable performance to an optimal inverter dispatch delayed by 2 minutes. The numerical tests have further corroborated the benefits of nonlinear rules with non-local inputs, and explored the trade-off between voltage regulation performance and sparsity. Finally, this work motivates several questions. On the implementation side, testing the novel formulations on multiphase grids along with capacitor banks, voltage regulators, and ZIP loads, is of practical interest. On the analytical side, chance-constrained formulations; studying the stability of nonlinear rules with voltages as inputs; using kernels to learn functions with constraints; and selecting non-local control inputs; are some open and interesting questions.

Bibliography

[1] R. Dobbe, O. Sondermeijer, D. Fridovich-Keil, D. Arnold, D. Callaway, and C. Tomlin, “Data-driven decentralized optimal power flow for distributed energy services in distribution grids.” [Online]. Available: https://arxiv.org/abs/1806.06790

[2] S. Karagiannopoulos, P. Aristidou, and G. Hug, “Data-driven local control design for active distribution grids using off-line optimal power flow and machine learning techniques,” *IEEE Trans. Smart Grid*, vol. PP, no. 99, pp. 1–1, 2019.

[3] K. Turitsyn, P. Sulc, S. Backhaus, and M. Chertkov, “Options for control of reactive power by distributed photovoltaic generators,” *Proc. IEEE*, vol. 99, no. 6, pp. 1063–1073, Jun. 2011.

[4] (2018) Pecan Street Inc. Dataport. [Online]. Available: https://dataport.cloud/

[5] Y. Agalgaonkar, B. Pal, and R. Jabr, “Stochastic distribution system operation considering voltage regulation risks in the presence of PV generation,” *IEEE Trans. Sustain. Energy*, vol. 6, no. 4, pp. 1315–1324, Oct. 2015.

[6] *IEEE 1547 Standard for Interconnecting Distributed Resources with Electric Power Systems*, IEEE Std., 2018. [Online]. Available: http://grouper.ieee.org/groups/scc21/1547/1547_index.html

[7] S. Low, “Convex relaxation of optimal power flow — Part II: Exactness,” *IEEE Trans. Control Netw. Syst.*, vol. 1, no. 2, pp. 177–189, Jun. 2014.

[8] G. Wang, V. Kekatos, A.-J. Conejo, and G. B. Giannakis, “Ergodic energy management

leveraging resource variability in distribution grids," *IEEE Trans. Power Syst.*, vol. 31, no. 6, pp. 4765–4775, Nov. 2016.

[9] Y. Zhang, N. Gatsis, and G. B. Giannakis, "Robust energy management for microgrids with high-penetration renewables," *IEEE Trans. Sustain. Energy*, vol. 4, no. 4, pp. 944–953, Oct. 2013.

[10] M. Farivar, R. Neal, C. Clarke, and S. Low, "Optimal inverter VAR control in distribution systems with high PV penetration," in *Proc. IEEE Power & Energy Society General Meeting*, San Diego, CA, Jul. 2012.

[11] B. Robbins, C. Hadjicostis, and A. Dominguez-Garcia, "A two-stage distributed architecture for voltage control in power distribution systems," *IEEE Trans. Power Syst.*, vol. 28, no. 2, pp. 1470–1482, May 2013.

[12] M. K. Singh, V. Kekatos, S. Taheri, and K. P. S. andChen Ching Liu, "Enforcing radiality constraints for DER-aided power distribution grid reconfiguration." [Online]. Available: https://arxiv.org/abs/1910.03020

[13] E. Dall'Anese, S. V. Dhople, and G. B. Giannakis, "Optimal dispatch of photovoltaic inverters in residential distribution systems," *IEEE Trans. Sustain. Energy*, vol. 5, no. 2, pp. 487–497, Dec. 2014.

[14] Q. Peng and S. Low, "Distributed algorithm for optimal power flow on a radial network," in *Proc. IEEE Conf. on Decision and Control*, Venice, Italy, Dec. 2014, pp. 167–172.

[15] M. Bazrafshan and N. Gatsis, "Decentralized stochastic optimal power flow in radial networks with distributed generation," *IEEE Trans. Smart Grid*, vol. 8, no. 2, pp. 787–801, Mar. 2017.

[16] B. Zhang, A. Dominguez-Garcia, and D. Tse, "A local control approach to voltage regulation in distribution networks," in *Proc. North American Power Symposium*, Manhattan, KS, Sep. 2013.

[17] N. Li, G. Qu, and M. Dahleh, "Real-time decentralized voltage control in distribution networks," in *Proc. Allerton Conf. on Comm., Control, and Computing*, Allerton, IL, Oct. 2014, pp. 582–588.

[18] V. Kekatos, L. Zhang, G. B. Giannakis, and R. Baldick, "Voltage regulation algorithms for multiphase power distribution grids," *IEEE Trans. Power Syst.*, vol. 31, no. 5, pp. 3913–3923, Sep. 2016.

[19] R. A. Jabr, "Linear decision rules for control of reactive power by distributed photovoltaic generators," *IEEE Trans. Power Syst.*, vol. 33, no. 2, pp. 2165–2174, Mar. 2018.

[20] K. S. Ayyagari, N. Gatsis, and A. F. Taha, "Chance-constrained optimization of distributed energy resources via affine policies," in *Proc. IEEE Global Conf. on Signal and Inf. Process.*, Montreal, Quebec, Canada, Nov. 2017.

[21] W. Lin and E. Bitar, "Decentralized stochastic control of distributed energy resources," *IEEE Trans. Power Syst.*, vol. 33, no. 1, pp. 888–900, Jan. 2018.

[22] K. Baker, A. Bernstein, E. Dall'Anese, and C. Zhao, "Network-cognizant voltage droop control for distribution grids," *IEEE Trans. Power Syst.*, vol. 33, no. 2, pp. 2098–2108, Mar. 2018.

[23] A. Garg, M. Jalali, V. Kekatos, and N. Gatsis, "Kernel-based learning for smart inverter control," in *Proc. IEEE Global Conf. on Signal and Inf. Process.*, Anaheim, CA, Dec. 2018.

[24] A. Garg, "Designing reactive power control rules for smart inverters using machine learning," Master's thesis, Virginia Tech, Blacksburg, US, Jun. 2018.

[25] M. Jalali, V. Kekatos, N. Gatsis, and D. Deka, "Designing reactive power control rules for smart inverters using support vector machines," *IEEE Transactions on Smart Grid*, pp. 1–1, 2019.

[26] S. Bolognani and F. Dorfler, "Fast power system analysis via implicit linearization of the power flow manifold," in *Proc. Allerton Conf. on Comm., Control, and Computing*, Allerton, IL, Sep. 2015, pp. 402–409.

[27] W. H. Kersting, *Distribution System Modeling and Analysis.* New York, NY: CRC Press, 2018.

[28] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer Series in Statistics, 2009.

[29] J. Abernethy, F. Bach, T. Evgeniou, and J.-P. Vert, "A new approach to collaborative filtering: Operator estimation with spectral regularization," *J. Machine Learning Res.*, vol. 10, pp. 803–826, 2009.

[30] J. A. Bazerque and G. B. Giannakis, "Nonparametric basis pursuit via sparse kernel-based learning," *IEEE Signal Process. Mag.*, vol. 12, pp. 112–125, Jul. 2013.

[31] M. Farivar, L. Chen, and S. Low, "Equilibrium and dynamics of local voltage control in distribution systems," in *Proc. IEEE Conf. on Decision and Control*, Florence, Italy, Dec. 2013, pp. 4329–4334.

[32] V. Kekatos and G. B. Giannakis, "Sparse Volterra and polynonial regression models: Recoverability and estimation," *IEEE Trans. Signal Process.*, vol. 59, no. 12, pp. 5907–5920, Dec. 2011.

[33] C. M. Bishop, *Pattern Recognition and Machine Learning.* New York, NY: Springer, 2006.

[34] V. Kekatos, Y. Zhang, and G. B. Giannakis, "Electricity market forecasting via low-rank multi-kernel learning," *IEEE J. Sel. Topics Signal Process.*, vol. 8, no. 6, pp. 1182–1193, Dec. 2014.

[35] M. S. Lobo, L. Vandenberghe, S. Boyd, and H. Lebret, "Applications of second-order cone programming," *Linear algebra and its applications*, vol. 284, no. 1, pp. 193–228, 1998.

[36] P. Ravikumar, J. Lafferty, H. Liu, and L. Wasserman, "Sparse additive models," *J. Roy. Stat. Soc. B*, vol. 71, no. 5, pp. 1009–1030, Oct. 2009.

[37] W. H. Kersting, "Radial distribution test feeders," in *Proc. Power Engineering Society Winter Meeting*, vol. 2, 2001, pp. 908–912.

[38] L. Gan, N. Li, U. Topcu, and S. Low, "On the exactness of convex relaxation for optimal power flow in tree networks," in *Proc. IEEE Conf. on Decision and Control*, Maui, HI, Dec. 2012, pp. 465–471.

[39] J. Lofberg, "A toolbox for modeling and optimization in MATLAB," in *Proc. of the CACSD Conf.*, 2004. [Online]. Available: http://users.isy.liu.se/johanl/yalmip/

[40] R. H. Tutuncu, K. C. Toh, and M. Todd, "Solving semidefinite-quadratic-linear programs using SDPT3," *Mathematical Programming Ser. B*, vol. 95, pp. 189–217, 2003.

[41] MOSEK, *The MOSEK optimization toolbox for MATLAB manual. Version 8.1.*, 2017. [Online]. Available: http://docs.mosek.com/8.1/toolbox/index.html

Appendices

Appendix A

Consider first the linear rules of (4.8), for which $q_n^g(\mathbf{z}_n) = \mathbf{z}_n^\top \mathbf{w}_n + b_n$ for all n. Problem (4.4) with $\Delta = \Delta_\tau$ can be reformulated as

$$\begin{aligned}
\min \quad & \frac{1}{T}\mathbf{s}^\top \mathbf{1} + \mu \boldsymbol{\gamma}^\top \mathbf{1} && && \text{(A.1a)}\\
\text{over} \quad & \{\mathbf{w}_n\}_{n=1}^N, \mathbf{b}, \mathbf{s} \geq \mathbf{0}, \boldsymbol{\gamma} && && \text{(A.1b)}\\
\text{s.to} \quad & -\bar{\mathbf{q}}_n^g \leq \mathbf{Z}_n^\top \mathbf{w}_n + b_n \mathbf{1} \leq \bar{\mathbf{q}}_n^g, && \forall n && \text{(A.1c)}\\
& \|\mathbf{w}_n\|_2 \leq \gamma_n, && \forall n && \text{(A.1d)}\\
& \|\mathbf{X}\mathbf{q}_t^g + \mathbf{y}_t\|_2 \leq s_t + \tau, && \forall t. && \text{(A.1e)}
\end{aligned}$$

Express voltage deviations at t in terms of $\mathbf{w}_n$'s and $\mathbf{b}$

$$\mathbf{X}\mathbf{q}_t^g + \mathbf{y}_t = \sum_{n=1}^N \mathbf{x}_n \mathbf{z}_{n,t}^\top \mathbf{w}_n + \sum_{n=1}^N b_n \mathbf{x}_n + \mathbf{y}_t.$$

Let us next introduce the Lagrange multipliers [35]:

- $\underline{\boldsymbol{\lambda}}_n \geq \mathbf{0}$ and $\overline{\boldsymbol{\lambda}}_n \geq \mathbf{0}$ corresponding to the linear inequalities in (A.1c) for all n;
- $(\mathbf{u}_n, \rho_n)$ related to constraint (A.1d) for all n; and
- $(\boldsymbol{\mu}_t, \sigma_t)$ related to constraint (A.1e) for all t.

Collect multipliers in $\mathbf{M} := [\boldsymbol{\mu}_1 \ \cdots \ \boldsymbol{\mu}_T] \in \mathbb{R}^{N\times T}$, and vectors $\boldsymbol{\rho} := [\rho_1 \ \cdots \ \rho_N]^\top$ and

$\boldsymbol{\sigma} := [\sigma_1 \ \cdots \ \sigma_T]^\top$. After some algebra, the Lagrangian of (A.1) can be written as

$$\begin{aligned} L = {} & \mathbf{s}^\top \left(\frac{1}{T}\mathbf{1} - \boldsymbol{\sigma} \right) + \boldsymbol{\gamma}^\top (\mu \mathbf{1} - \boldsymbol{\rho}) \\ & + \sum_{n=1}^{N} \mathbf{w}_n^\top \left[\mathbf{Z}_n \left(\overline{\boldsymbol{\lambda}}_n - \underline{\boldsymbol{\lambda}}_n - \mathbf{M}^\top \mathbf{x}_n \right) - \mathbf{u}_n \right] \\ & + \sum_{n=1}^{N} b_n \left[\left(\overline{\boldsymbol{\lambda}}_n - \underline{\boldsymbol{\lambda}}_n - \mathbf{M}^\top \mathbf{x}_n \right)^\top \mathbf{1} \right] \\ & - \sum_{n=1}^{N} \left(\overline{\boldsymbol{\lambda}}_n + \underline{\boldsymbol{\lambda}}_n \right)^\top \bar{\mathbf{q}}_n^g - \sum_{t=1}^{T} \boldsymbol{\mu}_t^\top \mathbf{y}_t - \tau \boldsymbol{\sigma}^\top \mathbf{1}. \end{aligned} \tag{A.2}$$

Minimizing L over the primal variables provides

$$\boldsymbol{\sigma} \leq \mathbf{1} \tag{A.3a}$$

$$\boldsymbol{\rho} = \mu \mathbf{1} \tag{A.3b}$$

$$\mathbf{u}_n = \mathbf{Z}_n \left(\overline{\boldsymbol{\lambda}}_n - \underline{\boldsymbol{\lambda}}_n - \mathbf{M}^\top \mathbf{x}_n \right), \qquad \forall n \tag{A.3c}$$

$$\left(\overline{\boldsymbol{\lambda}}_n - \underline{\boldsymbol{\lambda}}_n \right)^\top \mathbf{1} = \mathbf{x}_n^\top \mathbf{M} \mathbf{1}, \quad \forall n. \tag{A.3d}$$

From (A.3), the dual of (A.1) becomes the SOCP problem

$$\max \quad - \sum_{n=1}^{N} \left(\overline{\boldsymbol{\lambda}}_n + \underline{\boldsymbol{\lambda}}_n \right)^\top \bar{\mathbf{q}}_n^g - \sum_{t=1}^{T} \boldsymbol{\mu}_t^\top \mathbf{y}_t - \tau \boldsymbol{\sigma}^\top \mathbf{1} \tag{A.4a}$$

$$\text{over} \quad \{ \underline{\boldsymbol{\lambda}}_n, \overline{\boldsymbol{\lambda}}_n \}_{n=1}^{N}, \{ \boldsymbol{\mu}_t, \sigma_t \}_{t=1}^{T} \tag{A.4b}$$

$$\text{s.to} \quad \underline{\boldsymbol{\lambda}}_n \geq \mathbf{0}, \ \overline{\boldsymbol{\lambda}}_n \geq \mathbf{0}, \ \text{(A.3d)}, \quad \forall n \tag{A.4c}$$

$$\| \mathbf{Z}_n \left(\overline{\boldsymbol{\lambda}}_n - \underline{\boldsymbol{\lambda}}_n - \mathbf{M}^\top \mathbf{x}_n \right) \|_2 \leq \mu, \quad \forall n \tag{A.4d}$$

$$\| \boldsymbol{\mu}_t \|_2 \leq \sigma_t \leq 1, \quad \forall t. \tag{A.4e}$$

It is not hard to check that (A.1) and (A.4) are strictly feasible, so strong duality holds and both problems are solvable. The optimal primal and dual variables satisfy complementary slackness SOCPs; see [35, Sec. 4.1]. For constraints (A.1d) and (A.4d), these conditions identify three cases:

c1) If $\|\mathbf{w}_n\|_2 < \gamma_n$, then $\|\mathbf{u}_n\|_2 = \rho_n = 0$;

c2) If $\|\mathbf{u}_n\|_2 < \rho_n$, then $\|\mathbf{w}_n\|_2 = \gamma_n = 0$; or

c3) If $\|\mathbf{w}_n\|_2 = \gamma_n$ and $\|\mathbf{u}_n\|_2 = \rho_n$, then $\gamma_n \mathbf{u}_n = -\rho_n \mathbf{w}_n$.

Recall that $\rho_n = \mu > 0$ from (A.3b). Moreover, it is not hard to see that $\|\mathbf{w}_n\|_2 = \gamma_n$ at the optimum of (A.1). Then, case *c1)* cannot occur. The other two cases entail that $\mathbf{w}_n = \alpha_n \mathbf{u}_n$ for some $\alpha_n \leq 0$. Substituting $\mathbf{u}_n$ from (A.3c), and evaluating rule n at the tested scenarios gives

$$\begin{aligned}
\mathbf{q}_n^g &= \mathbf{Z}_n^\top \mathbf{w}_n + b_n \mathbf{1} \\
&= \alpha_n \mathbf{Z}_n^\top \mathbf{Z}_n \left(\overline{\boldsymbol{\lambda}}_n - \underline{\boldsymbol{\lambda}}_n - \mathbf{M}^\top \mathbf{x}_n\right) + b_n \mathbf{1} \\
&= \mathbf{K}_n \mathbf{a}_n + b_n \mathbf{1}.
\end{aligned}$$

Here we identify $\mathbf{K}_n = \mathbf{Z}_n^\top \mathbf{Z}_n$ and the coefficients in (4.10) as

$$\mathbf{a}_n := \alpha_n \left(\overline{\boldsymbol{\lambda}}_n - \underline{\boldsymbol{\lambda}}_n - \mathbf{M}^\top \mathbf{x}_n\right). \tag{A.5}$$

Focus now on the complementary slackness for (A.1e) and (A.4e). The equivalent to condition *c1)* reads now as:

c1') If $\|\mathbf{X}\mathbf{q}_t^g + \mathbf{y}_t\|_2 < s_t + \tau$, then $\|\boldsymbol{\mu}_t\|_2 = \sigma_t = 0$.

Suppose the optimal primal variables satisfy $\|\mathbf{X}\mathbf{q}_t^g + \mathbf{y}_t\|_2 < \tau$. Then $s_t = 0$ follows from (A.1), and *c1')* gives $\|\boldsymbol{\mu}_t\|_2 = \sigma_t = 0$. The t-th entry of $\mathbf{a}_n$ in (A.5) is

$$a_{n,t} = \alpha_n \left(\overline{\lambda}_{n,t} - \underline{\lambda}_{n,t} - \boldsymbol{\mu}_t^\top \mathbf{x}_n\right). \tag{A.6}$$

Complementary slackness for (A.1c) implies that $\overline{\lambda}_{n,t} = \underline{\lambda}_{n,t} = 0$ if $|q_{n,t}^g| < \bar{q}_{n,t}^g$ at the optimal, thus proving the claim for linear rules. The result in (A.6) holds for nonlinear rules too. The analysis carries over upon matching the length of $\mathbf{w}_n$ with the length of $\boldsymbol{\phi}(\mathbf{z}_n)$, and substituting $\mathbf{Z}_n^\top \mathbf{Z}_n$ by $\mathbf{K}_n$.

Rewrite (4.4) for $\Delta = \Delta_\epsilon$ as

$$\begin{aligned}
\min \quad & \frac{1}{T}\sum_{t=1}^{T} \mathbf{s}_t^\top \mathbf{1} + \mu \boldsymbol{\gamma}^\top \mathbf{1} && && \text{(A.7a)}\\
\text{over} \quad & \{\mathbf{w}_n\}_{n=1}^{N}, \mathbf{b}, \mathbf{s} \geq \mathbf{0}, \boldsymbol{\gamma} && && \text{(A.7b)}\\
\text{s.to} \quad & -\bar{\mathbf{q}}_n^g \leq \mathbf{Z}_n^\top \mathbf{w}_n + b_n \mathbf{1} \leq \bar{\mathbf{q}}_n^g, && \forall n && \text{(A.7c)}\\
& \|\mathbf{w}_n\|_2 \leq \gamma_n, && \forall n && \text{(A.7d)}\\
& -\mathbf{s}_t - \epsilon \mathbf{1} \leq \mathbf{X}\mathbf{q}_t^g + \mathbf{y}_t \leq \mathbf{s}_t + \epsilon \mathbf{1}, && \forall t. && \text{(A.7e)}
\end{aligned}$$

The Lagrangian multipliers of (A.7) are similar to shose of (A.1), except for $(\boldsymbol{\mu}_t, \sigma_t)$ being replaced by $(\underline{\boldsymbol{\mu}}_t, \overline{\boldsymbol{\mu}}_t)$ and collected in $\underline{\mathbf{M}} := [\underline{\boldsymbol{\mu}}_1 \ \cdots \ \underline{\boldsymbol{\mu}}_T]$ and $\overline{\mathbf{M}} := [\overline{\boldsymbol{\mu}}_1 \ \cdots \ \overline{\boldsymbol{\mu}}_T]$. Minimizing the Lagrangian of (A.7) over the primal variables yields

$$\mathbf{u}_n = \mathbf{Z}_n \left[\overline{\boldsymbol{\lambda}}_n - \underline{\boldsymbol{\lambda}}_n + \left(\overline{\mathbf{M}} - \underline{\mathbf{M}}\right)^\top \mathbf{x}_n\right], \quad \forall n.$$

Similar to Prop. 5.1, the t-th entry of $\mathbf{a}_n$ becomes

$$a_{n,t} = \overline{\lambda}_{n,t} - \underline{\lambda}_{n,t} + (\overline{\boldsymbol{\mu}}_t - \underline{\boldsymbol{\mu}}_t)^\top \mathbf{x}_n.$$

If the optimal primal variables satisfy $|\mathbf{X}\mathbf{q}_t + \mathbf{y}_t| \not< \epsilon\mathbf{1}$, then $\mathbf{s}_t \neq \mathbf{0}$ and accordingly, complementary slackness for (A.7e) implies that $\overline{\boldsymbol{\mu}}_t \neq \mathbf{0}$ or $\underline{\boldsymbol{\mu}}_t \neq \mathbf{0}$.