2017 American Control Conference
Sheraton Seattle Hotel
May 24–26, 2017, Seattle, USA

Dynamic Energy Management with Scenario-Based Robust MPC

Matt Wytock, Nicholas Moehle and Stephen Boyd

Abstract— **We present a simple, practical method for managing the energy produced and consumed by a network of devices. Our method is based on (convex) model predictive control. We handle uncertainty using a robust model predictive control formulation that considers a finite number of possible scenarios. A key attribute of our formulation is the encapsulation of device details, an idea naturally implemented with object-oriented programming. We introduce an open-source Python library implementing our method and demonstrate its use in planning and control at various scales in the electrical grid: managing a smart home, shared charging of electric vehicles, and integrating a wind farm into the transmission network.**

I. Introduction

Optimization has long been central to the management of the electrical grid, in the form of the economic dispatch of generation in the transmission network [1], [2]. As the generation mix shifts to renewable resources, there is a greater need for more extensive planning extending down to the distribution network and behind the meter of individual consumers. In addition, networked devices are now enabling greater coordination and control of loads than has previously been feasible. As with dispatch of generators in the transmission network, optimization provides a disciplined mechanism for formulating and minimizing total system cost in these new scenarios, scheduling how devices should produce or consume power while respecting physical system constraints.

At the same time, planning in energy problems often involves intrinsic uncertainty around future supply and demand, which is increasing with greater renewable integration and a focus on more granular domains. In these situations, model predictive control (MPC) [3] provides a natural framework for planning under uncertainty, leveraging optimization. Many sophisticated methods for robust MPC have been proposed, using a variety of models and approaches; see *e.g.*, [4], [5], [6], [7], [8]. In the context of energy management, it is often the case that there are several plausible scenarios (*e.g.*, generator or transmission outage, uncertain renewable generation, *etc.*). Our approach to robust MPC is to optimize the worst-case performance over scenarios that are explicitly provided as inputs to the optimization problem.

Our method, which we refer to as dynamic energy management (DEM), provides a general framework for optimization problems involving a network of devices exchanging energy. The basis of the framework is an extensible set of devices, which are themselves defined by small optimization problems. It encompasses the standard static and dynamic power flow problems, as well as new scenarios involving storage, and indeed any scenario involving devices that can be represented with a cost or utility function and constraints. This framework is a natural fit for object-oriented programming, with objects encapsulating the device-specific implementation details. For the purposes of this work, we restrict our attention to devices implemented with convex functions which allows robust and efficient algorithms from convex optimization to be applied.

There already exists several popular software packages for computing optimal power flow and analyzing electrical grid operations at various levels. Packages such as MATPOWER [9] and PSAT [10] compute optimal power flow and provide other tools for analyzing the transmission network. GridLAB-D [11] and GridSpice [12] extend into modeling the distribution network, with a focus on producing accurate simulations using detailed physical models of devices. These tools are open source; commercial software packages with similar features, such as PSS from Siemens [13], also exist. In general, the focus of these software packages is on highly accurate physical device simulation, rather than optimization. To the extent optimization is supported, it is typically restricted to a standard problem formulations, such as optimal power flow, with limited support for extension to new scenarios. In contrast, this work focuses on developing an extensible device model for the purposes of optimization and a general framework for handling uncertainty.

This paper is organized as follows. In §II, we describe the general optimization framework for devices exchanging power over a network and describe how generators, loads, and other common devices fit into this model. In §III, we extend the framework to handle uncertainty using model predictive control with robust optimization and multiple plausible scenarios. In §IV, we provide numerical examples demonstrating how the dynamic energy management framework can be applied to a variety of problems.

II. Network model

The network model consists of three abstractions: *devices*, *nets* and *terminals*. Devices are defined by mathematical functions that map power flow into and out of the device to operating cost. Nets are connection points at which power is exchanged among devices; each device has one or more terminals which connects that Device to one or more nets. At each net, power balance is enforced and a price assigned, allowing any number of devices to exchange power at a single point. Specifying the device cost functions, nets and the connectivity between devices and nets defines the complete specification of the network.

This research was supported in part by the DARPA XDATA program and a grant from MISO

Matt Wytock, Nicholas Moehle and Stephen Boyd are with the Electrical Engineering Department, Stanford University, Stanford, CA 94305

The DEM optimal power flow (OPF) problem is

$$\begin{array}{ll} \text{minimize} & f(p) \\ \text{subject to} & Ap = 0 \end{array} \qquad (1)$$

with the decision variable p representing the power flow across all terminals. We may consider the *static* OPF problem in which case p is indexed by terminal (*i.e.*, p is vector) or the *dynamic* OPF problem in which case p is indexed by terminal and time (*i.e.*, p is a matrix). In either case, the constraint $Ap = 0$ ensures power conservation across nets at each time point, with A being the net-terminal adjacency matrix. The objective function $f(p)$ is the sum of device cost functions and thus is separable across devices. We define f to be extended valued, with an infinite cost corresponding to an infeasible power flow.

In addition to power schedules, solving the DEM optimization problem produces prices at each net and time. In particular, the dual variables associated with the power balance constraints are locational marginal prices (LMPs), representing the cost of additional power at a particular time and place in the network [14]. These prices may form the basis of an energy payment scheme, with devices paying or receiving compensation for power produced or consumed. For example, in the economic dispatch of generation in the transmission network, LMPs determine the price that load pays to generators. When congestion occurs, the price differs across the network, resulting in surpluses associated with individual transmission lines. Financial transmission rights (FTRs) securitize these surplus payments and are sold at auction or simply given to load serving entities in order to hedge congestion costs [15].

In the remainder of this section, we give brief, high-level descriptions of how devices of various types fit into this model. The complete specification and implementation of a common set of devices is provided as part of the open-source library, which is presented in §IV. In addition, more complete mathematical descriptions and discussion of many device models, including convex relaxations for nonconvex costs and constraints, can be found in [16].

Generation. Generators are single-terminal devices with negative power flow, representing generation. Conventional generators, which produce energy from fossil fuels (*e.g.*, natural gas, coal), tend to have marginal costs dominated by fuel costs. In addition, conventional generators have operating constraints determined by maximum capacity, ramping considerations, and a minimum operating level.

Renewable generators (*e.g.*, wind, solar) have a variable maximum capacity determined by resource availability and typically have zero marginal cost. As these generators are built with power electronics, they can ramp up and down quickly and can typically curtail in periods of overgeneration.

Generation can also be represented abstractly with a grid connection which is appropriate certain scenarios, such as modeling a single home or neighborhood. In this case, power is provided by the utility which typically provides a flat rate in \$/kWh, represented with a linear cost. Incorporating more sophisticated utility rate plans is also possible, including tiered rate structures, time-of-use rates, demand charges and demand response incentives.

Load. Loads are single-terminal devices with positive power flow, representing consumption. In conventional power flow problems, loads are treated as fixed which is represented in this framework with a device containing a single equality constraint.

Flexible loads (*e.g.*, electric vehicle chargers, curtailable lighting, *etc.*) provide a utility function specifying the value of power, along with constraints on their consumption. For example, an electric vehicle charger specifies the value of energy stored in the EV battery, and will typically prefer faster charging in order to ensure departure flexibility. In addition, an EV charging device limits the maximum charging rate and the total amount of energy consumed, in accordance with available storage capacity in the battery.

Thermal loads (*e.g.*, HVAC, refrigerator, water heater, *etc.*) include additional state in the form of internal temperatures and a model of how energy consumption affects these temperatures. The model can specify hard constraints on temperatures, or a utility function that allows for temperature targets to be relaxed in periods of high energy prices.

Interface devices. Transmission lines and other interface devices have two or more terminals and flow power from different points or convert electricity to different forms. These devices have a maximum capacity and may include models of the losses associated with conversion or transmission. For example, in the standard optimal power flow problem, transmission lines connect variable generation to fixed load, specifying the capacity limits in the transmission network. In the distribution network, transformers impose limits on power as well as losses as they step down voltage from distribution lines for consumption by customers.

III. Control and uncertainty

A. Model predictive control

In this section, we discuss how the DEM framework can be used for planning and control. Model predictive control, also known as receding-horizon control, is a feedback control technique that naturally incorporates optimization [3]. The simplest version is *certainty-equivalent* MPC: at each time step, we replace unknown quantities with predictions and solve the optimization problem to produce a power flow schedule. We then execute the first action in the schedule. At the next time step, we repeat this process, incorporating the latest information available into our prediction models.

We perform the following steps to construct power flow schedule covering time intervals $t, \dots, t+T$:

1) *Predict.* Predict unknown quantities to form an estimate of the device objective functions.
2) *Optimize.* Solve problem (1) over time horizon $t, \dots, t+T$ to obtain an optimal power schedule $p^\star$.
3) *Execute.* Instruct all devices to follow the first step of the power flow schedule, $p_t^\star$.

We then repeat this procedure, incorporating new information, at time $t+1$.

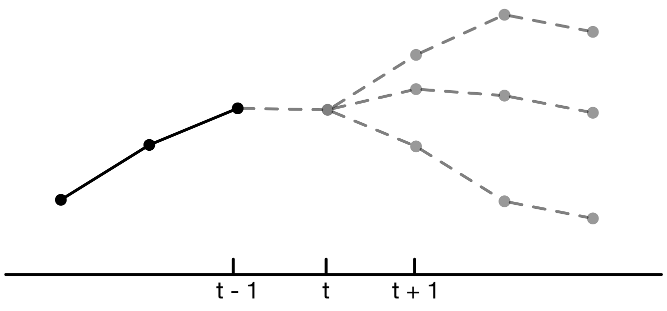

Fig. 1. Example of historical actions (solid line) and schedule (dashed lines) at time t, under multiple scenarios. At time t, the schedule for each scenario is constrained to be equal, but may diverge at time $t+1, \ldots, t+T$.

B. Scenario-based robust MPC

Due to feedback, certainty-equivalent MPC often performs surprisingly well even with simple prediction models. However, in many energy management applications, there exist several plausible scenarios which we would like to explicitly incorporate. This is accomplished by introducing a *scenario generator*, which generates multiple scenarios in place of single predictions used by certainty-equivalent MPC. The architecture is agnostic to the nature of the scenario generator, possibilities include sampling from a statistical model or a rule incorporating domain-specific knowledge, *e.g.*, adding redundancy to transmission line or generator outages. Given multiple scenarios as input, $s = 1, \ldots, S$, we formulate the robust OPF problem by minimizing the the cost in the worst case,

$$\begin{array}{ll} \text{minimize} & \max_s f^{(s)}(p^{(s)}) \\ \text{subject to} & Ap^{(s)} = 0, \forall s \\ & p^{(s)} \text{ is equal at time } t, \ \forall s. \end{array} \tag{2}$$

The decision variables are $p^{(1)}, \ldots, p^{(S)}$, power flow schedules for each scenario. The second constraint ensures that that the first step (time t) is equal across the schedules for all scenarios, an example is shown in Fig. 1. We note that minimizing worst case cost is only one possible formulation, see *e.g.*, [17] §7.5.

The outer loop of the robust MPC procedure proceeds similarly to certainty-equivalent MPC with small modification. In the prediction step, we now employ the scenario generator to provide a set of objective functions, one for each scenario. In the optimization step, we solve problem (2), resulting in a single instruction for time t, which is provided for execution.

IV. Numerical examples

In this section, we present examples demonstrating the application of the DEM framework to various energy management problems. In the first example, we manage networked devices in a smart home supplied only by solar generation and storage, demonstrating planning with several devices including HVAC load, which must regulate the home temperature despite limited solar power. In the shared EV charging example, we demonstrate the sharing of distribution resources and discuss payment schemes that arise fro2m prices associated with nets in the DEM framework. The wind farm integration example demonstrates planning under uncertainty using robust MPC, with a statistical model for future wind power production generating wind power scenarios which are used to co-optimize storage and gas generation to provide a firm power output.

The dynamic energy management framework is implemented in an open-source library available at

```
http://github.com/cvxgrp/dem
```

along with the complete code and data for the examples. The library is an extension to CVXPY [18], providing device models for generators, loads (fixed, deferrable, curtailable, *etc.*), storage systems (*e.g.*, battery, pumped hydro), thermal devices (*e.g.*, HVAC, refrigerator, water heater) and transmission lines. In addition, the library provides tools for forming and solving the dynamic energy management problem, supporting both certainty-equivalent and robust MPC formulations.

A. Smart home

Our first example demonstrates a number of devices jointly scheduled by a smart home energy system. In this example, the home is self-sufficient, drawing power only from solar generation and a small storage system. In order to accommodate load in the home, tradeoffs must be made and energy consumption optimized for limited generation. In addition, solar power is most prevalent during the middle of the day, whereas peak consumption occurs in the evening.

The home contains the following devices:

- *Solar.* Power is provided by solar generation, which follows a typical diurnal profile and has a maximum output of 10 kW.
- *Battery.* A small amount of storage is provided by a battery with capacity of 6.4 kWh and a maximum charge/discharge rate of 3.3 kW.
- *Lights.* Lighting consumes a maximum of 100 W from 6-9am and 500 W from 6-10pm and is curtailable.
- *EV.* An electric vehicle arrives at 5pm and desires 10 kWh of charge, with maximum charging rate of 9.6 kW.
- *HVAC.* The HVAC system must maintain the internal temperature of the house at or below 78° F on a day that reaches 98.6° F externally. The maximum power consumption of the system is 10 kW.

The smart home example is constructed with the following Python code.

```
solar = Generator(power_max=p_solar)
battery = Storage(
  charge_max=3.3, energy_init=3.2*4,
  energy_max=6.4*4)
lights = FlexibleLoad(
  power_max=p_lights, gamma=0.1)
ev = FlexibleLoad(
  power_max=p_ev, energy_max=10*4,
  alpha=0.1)
hvac = ThermalLoad(
  power_max=10, temp_amb=temp,
```

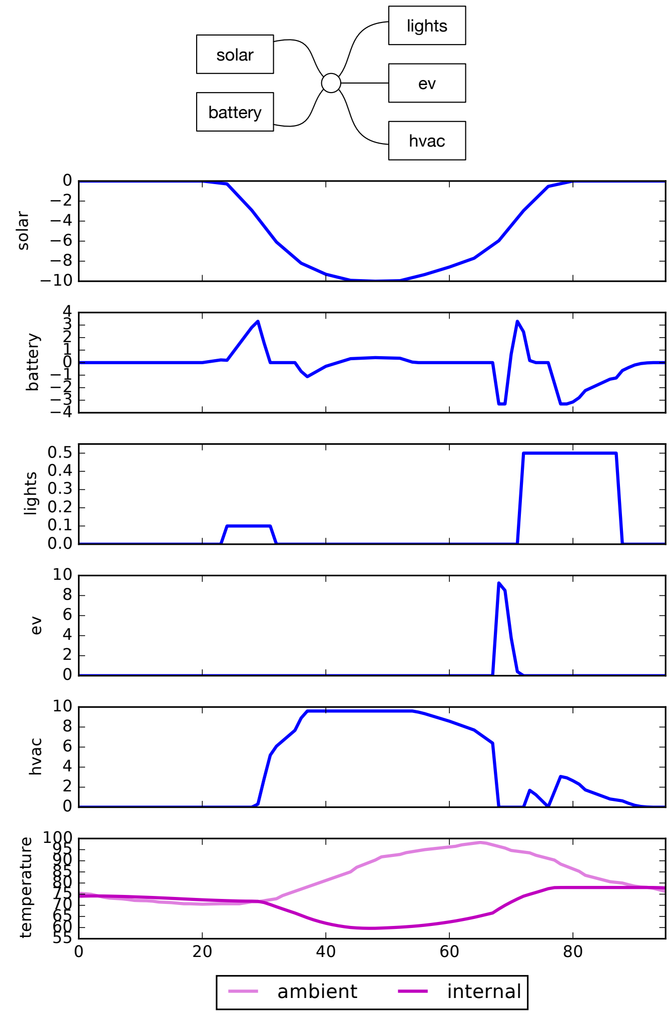

Fig. 2. Network for smart home example (top); power schedules along with ambient and internal temperature (bottom).

```
  temp_init=74, temp_max=78,
  efficiency=0.95, capacity=5,
  amb_conduct_coeff=0.3)
net = Net([solar.terminals[0],
           battery.terminals[0],
           lights.terminals[0]
           ev.terminals[0],
           hvac.terminals[0]])
network = Group(
  [solar, battery, lights, ev, hvac],
  [net])
network.optimize()
```

In the code above, `p_solar`, `p_lights`, and `p_ev` contain the time-varying maximum power output or consumption of the devices and `temp` contains the external temperature.

The results of optimizing a single day of energy usage in the smart home in 15 minute intervals are shown in Fig. 2. The HVAC system is able to maintain the internal temperature at or below 78° F, largely through pre-cooling the house when solar energy is prevalent in the middle of the day. During this period, the battery is also charged to full capacity. When the EV arrives at 5pm, energy from solar generation as well as battery storage is shifted to EV charging. Later in the evening, a small amount of energy is provided to the HVAC system in order to maintain the internal temperature. The lights remaining on and operating at full power, but the EV receives only about 5 kWh of the desired 10 kWh. The tradeoff between lights, EV and HVAC are represented in the utility functions associated with these devices and can be modified to accommodate user preferences, as desired.

B. Shared EV charging

The next example considers sharing oversubscribed EV charging resources. Due to flexibility in schedules and high maximum demand, it is natural to employ optimization to accommodate EV charging needs rather than over build physical infrastructure to accommodate peak demand. The model described here could be applied to a residential neighborhood or in a commercial setting in which a business provides charging to employees or customers.

In this scenario, Alice (A), Bob (B) and Carol (C) all would like to charge their EVs at the maximum rate of 9.6 kW for 3 hours, but the aggregate charging rate is limited to a maximum of 12 kW. In order to determine priority, each user specifies their personal value of charging via a utility function parameterized by $\gamma \geq 0$,

$$\sum_{\tau=t}^{t+T} \gamma q(\tau), \tag{3}$$

where $q(\tau)$ is the total charge received by time τ; *i.e.*, for power schedule $p_t, \dots, p_{t+T}$,

$$q(\tau) = \sum_{s=t}^{\tau} p_s. \tag{4}$$

The network representing this scenario is shown in Fig. 3 (top) and consists of two nets connected by a transformer with limited capacity. The EVs are connected on one side of the transformer and the grid, which provides power at a rate of \$0.1/kWh, is connected to the other.

The shared EV charging example is constructed with the following Python code.

```
p = 9.6     # 240A * 40V = 9.6 kW
e = p*3*12  # 3 hours of full charging
ev_a = FlexibleLoad(
  power_max=p_ev_a, energy_max=e,
  alpha=0.1/12)
ev_b = FlexibleLoad(
  power_max=p_ev_b, energy_max=e,
  alpha=1./12)
ev_c = FlexibleLoad(
  power_max=p_ev_c, energy_max=e,
  alpha=10./12)
trans = Transformer(power_max=12)
grid = Generator(beta=0.1, power_max=1000)
nets = [Net([ev_a.terminals[0],
```

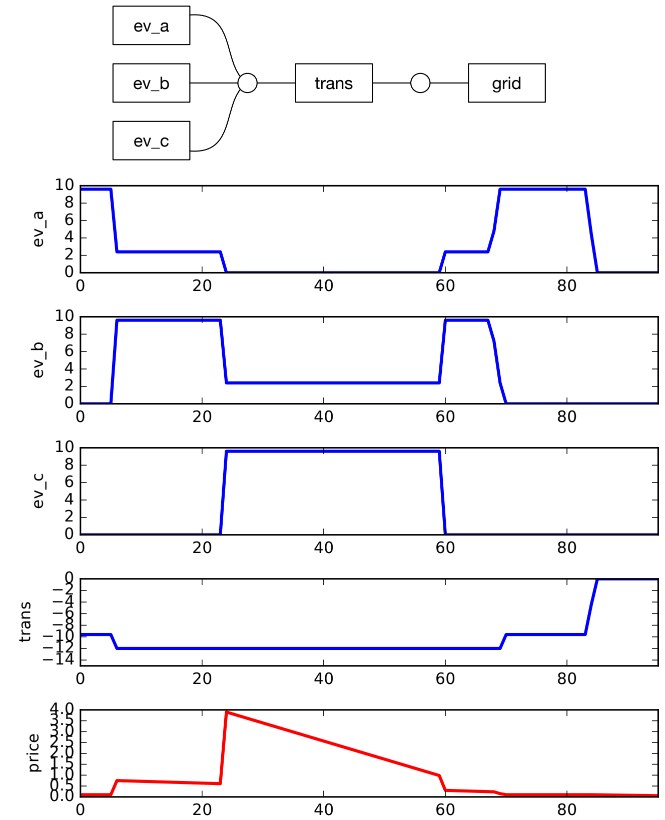

Fig. 3. Network for shared EV charging example (top); power schedules and price for leftmost net (bottom).

```
                ev_b.terminals[0],
                ev_c.terminals[0],
                trans.terminals[0]]),
          Net([grid.terminals[0],
                trans.terminals[1]])]
network = Group(
   [ev_a, ev_b, ev_c, trans, grid], nets)
network.optimize()
```

In the code above, p_ev_a, p_ev_b, and p_ev_c contain the maximum charging rates for each EV, which effectively encode the arrival time of each participant.

Fig. 3 (bottom) shows the resulting power schedules for the shared EV charging example. The utility functions are such that $\gamma_a \leq \gamma_b \leq \gamma_c$ and thus although Alice arrives first, her charging is reduced when Bob arrives. Carol, who places a much higher value on charging than the others, arrives last but immediately receives a full charge. During this period, the price at the net to which the EVs are connected is significantly higher than the utility price of \$0.1/kWh. This reflects the higher contention for the charging resource and specifically the marginal increase in utility that would result from greater charging capacity during this time. When Carol completes charging, Bob is able to resume at full speed, resulting in drop in price.

Applying the standard locational marginal pricing scheme to this scenario results in the participants paying more than the utility price. This is due to contention at the transformer; these surplus payments could naturally be assigned to the asset owner to be used toward infrastructure upgrades to resolve future contention. Or, they could be divided among the participants whose charging was curtailed as compensation for delay.

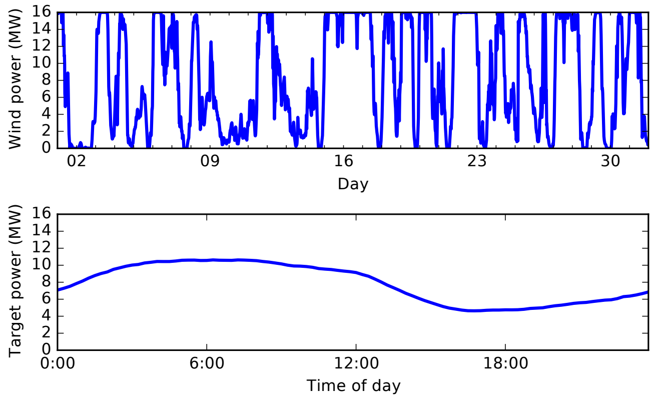

Fig. 4. Wind power over one month (top); daily contract for wind farm output (bottom).

C. Wind farm integration

We consider wind farm integration with data from the Wind Integration National Data Set [19]. The wind farm owner would like to provide a firm contract for energy production by augmenting the wind resource with storage and gas generation. We construct this contract using the average wind power output during each interval of the day, shown in Fig. 4, which also includes the actual wind power output for the wind farm.

The wind farm integration example is constructed with the following Python code.

```
load = FixedLoad(power=Parameter(T,K))
wind = Generator(
  alpha=0, beta=0,
  power_min=0, power_max=Parameter(T,K))
gas = Generator(
  alpha=0.02, beta=1,
  power_min=0.01, power_max=1)
battery = Storage(
  discharge_max=1, charge_max=1,
  energy_max=12, energy_init=Parameter(1))
net = Net([
  wind.terminals[0],
  gas.terminals[0],
  storage.terminals[0],
  output.terminals[0]])
```

In the code above, T represents the time horizon and K represents the number of scenarios, which we will use to handle uncertainty over future wind power production.

We generate S scenarios by using a simple autoregressive model to predict the wind power. More specifically, the

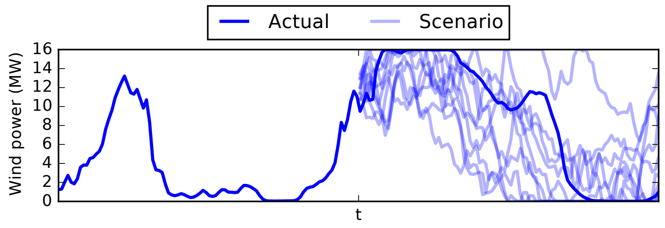

Fig. 5. Example of wind power scenarios.

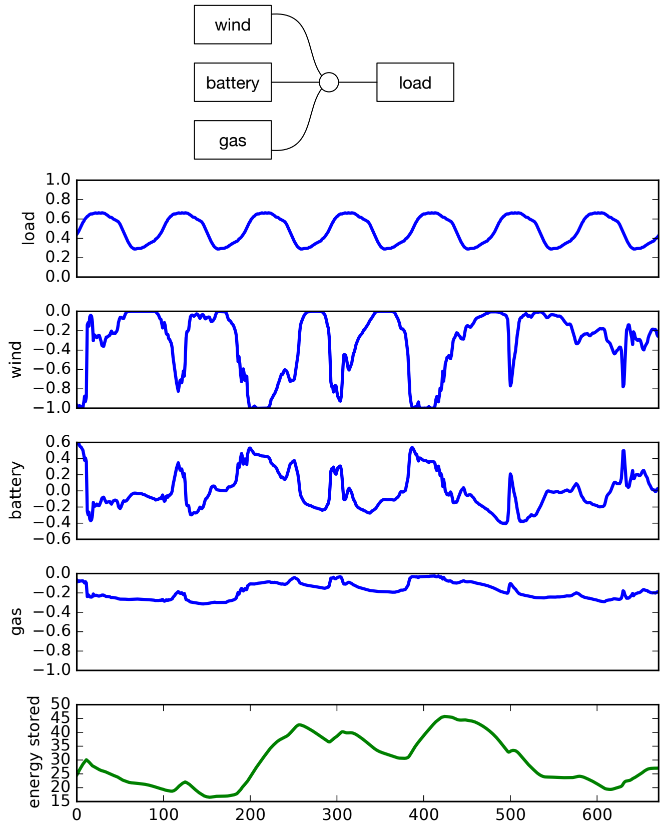

Fig. 6. Network for wind farm integration example (top); Robust MPC results showing executed actions and energy stored in battery (bottom).

predicted wind power $\hat{p}_t^{\mathrm{wind}}$ at time t is generated recursively according to

$$\hat{p}_t^{\mathrm{wind}} = \sum_{i=1}^{24} \alpha_i \hat{p}_{t-i}^{\mathrm{wind}} + \beta \bar{p}_t + \epsilon_t, \tag{5}$$

where $\bar{p}_t$ is the historical average of wind power production for the daily interval corresponding to t and ϵ_t is sampled according to a zero-mean normal distribution with standard deviation σ. The parameters σ, α and β are estimated from historical data. The recursion is initialized so that past predictions correspond to the known, measured values. Example wind power scenarios generated from this model are shown in Fig. 5.

Fig. 6 shows the executed actions produced by robust MPC for a period of 7 days. At each time step, we generate $S = 10$ scenarios and plan with 15 minute intervals and a time horizon of 48 hours ($T = 192$). The augmented wind farm is able to meet the firm contract with the majority of wind variability handled by the battery, which charges during periods of excess capacity. The gas generator provides power at a sustained low level, which is desirable since running the gas generator at a higher level incurs higher marginal cost. Uncertainty causes robust MPC to produce a sustained small amount of energy from gas without fully charging or discharging the battery at any time, retaining flexibility in the system.

References

[1] T. Wu, M. Rothleder, Z. Alaywan, and A. D. Papalexopoulos, "Pricing energy and ancillary services in integrated market systems by an optimal power flow," *IEEE Transactions on power systems*, vol. 19, no. 1, pp. 339–347, 2004.

[2] Federal Energy Regulatory Commission, "Security constrained economic dispatch: definition, practices, issues and recommendations," Tech. Rep., 2006.

[3] J. Mattingley, Y. Wang, and S. Boyd, "Receding horizon control," *IEEE Control Systems*, vol. 31, no. 3, pp. 52–65, 2011.

[4] M. V. Kothare, V. Balakrishnan, and M. Morari, "Robust constrained model predictive control using linear matrix inequalities," *Automatica*, vol. 32, no. 10, pp. 1361–1379, 1996.

[5] M. Cannon and B. Kouvaritakis, "Optimizing prediction dynamics for robust MPC," *IEEE Transactions on Automatic Control*, vol. 50, no. 11, pp. 1892–1897, 2005.

[6] B. Ding, Y. Xi, M. T. Cychowski, and T. OMahony, "Improving off-line approach to robust MPC based-on nominal performance cost," *Automatica*, vol. 43, no. 1, pp. 158–163, 2007.

[7] A. Bemporad and M. Morari, "Robust model predictive control: A survey," in *Robustness in identification and control*. Springer, 1999, pp. 207–226.

[8] D. Limon, J. Bravo, T. Alamo, and E. Camacho, "Robust MPC of constrained nonlinear systems based on interval arithmetic," *IEE Proceedings-Control Theory and Applications*, vol. 152, no. 3, pp. 325–332, 2005.

[9] R. D. Zimmerman, C. E. Murillo-Sánchez, and R. J. Thomas, "Matpower: Steady-state operations, planning, and analysis tools for power systems research and education," *IEEE Transactions on power systems*, vol. 26, no. 1, pp. 12–19, 2011.

[10] F. Milano, "An open source power system analysis toolbox," *IEEE Transactions on Power systems*, vol. 20, no. 3, pp. 1199–1206, 2005.

[11] D. P. Chassin, K. Schneider, and C. Gerkensmeyer, "Gridlab-d: An open-source power systems modeling and simulation environment," in *2008 IEEE/PES Transmission and Distribution Conference and Exposition*, 2008.

[12] K. Anderson, J. Du, A. Narayan, and A. El Gamal, "Gridspice: A distributed simulation platform for the smart grid," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 4, pp. 2354–2363, 2014.

[13] P. Siemens, "Pss," *E 30.2 Program Operational Manual*, 2011.

[14] R. E. Bohn, M. C. Caramanis, and F. C. Schweppe, "Optimal pricing in electrical networks over space and time," *The Rand Journal of Economics*, pp. 360–376, 1984.

[15] J. Rosellón and T. Kristiansen, *Financial Transmission Rights*. Springer, 2013.

[16] M. Kraning, E. Chu, J. Lavaei, S. P. Boyd *et al.*, "Dynamic network energy management via proximal message passing." *Foundations and Trends in Optimization*, vol. 1, no. 2, pp. 73–126, 2014.

[17] N. Parikh, S. P. Boyd *et al.*, "Proximal algorithms." *Foundations and Trends in optimization*, vol. 1, no. 3, pp. 127–239, 2014.

[18] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.

[19] C. Draxl, A. Clifton, B.-M. Hodge, and J. McCaa, "The wind integration national dataset (WIND) toolkit," *Applied Energy*, vol. 151, pp. 355–366, 2015.