CHAPTER 5: OPTIMAL ENERGY MANAGEMENT

1 Overview

In this chapter we study the optimal energy management problem. Optimal energy management involves the control of power flow between a network of generators, storage, and loads to optimize some performance criterion. Example performance criteria include minimizing fuel consumption, maximizing energy efficiency, minimizing electric utility costs, or minimizing carbon emissions. Our key tool is dynamic programming (DP). Unlike the static optimization problems of CH4, we consider energy systems which evolve dynamically in time. That is, our decisions now impact the energy system's future state, and therefore future decisions. Such a multi-stage process is ideal for dynamic programming.

In this chapter, we also consider cases where the generation and/or loads are stochastic. Specifically, we are interested in cases where generation is random (e.g. solar or wind) and/or loads are random (e.g. building power consumption based on human behavior). In these cases, we cannot perfectly forecast generation and/or load. However, we may be able to characterize it statistically. In more concrete terms, we utilize a stochastic process model to describe how generation and/or load evolves in time. For this case, we introduce stochastic dynamic programming (SDP) as our main optimization tool.

The optimal energy management problem finds application in nanogrids (e.g. hybrid vehicles [1, 2], buses [3], and individual buildings [4, 5]), microgrids (e.g. UC San Diego campus [6] and military forward operating bases [7]), or macro grids (i.e. distribution circuits in power systems).

1.1 Chapter Organization

The remainder of this chapter is organized as follows:

1. Canonical Energy Management Problem
2. Optimal HEV Energy Management
3. Optimal PEV Charging Schedule
4. Dynamic Programming
5. Optimal Home Appliance Schedule
6. Stochastic Dynamic Programming

2 Canonical Energy Management Problem

The canonical energy management problem (EMP) is mathematically represented by

$$\begin{aligned}
\text{EMP:} \quad \min_{x(k),u(k)} \quad J &= \sum_{k=0}^{N-1} c_k(x_k, u_k) + c_N(x_N) && (1)\\
\text{s. to} \quad x_{k+1} &= f(x_k, u_k), \qquad k = 0, 1, \cdots, N-1 && (2)\\
x_0 &= x_{init} && (3)\\
g(x(k), u(k)) &\leq 0, \qquad k = 0, 1, \cdots, N && (4)\\
h(x(k), u(k)) &= 0, \qquad k = 0, 1, \cdots, N && (5)
\end{aligned}$$

Compare to the canonical nonlinear programming problem from CH4:

$$\begin{aligned}
\text{NLP:} \quad \min_{x} \quad & f(x), && (6)\\
\text{s. to} \quad & g(x) \leq 0, && (7)\\
& h(x) = 0. && (8)
\end{aligned}$$

Note the following critical differences in the EMP relative to the NLP:

- First, the EMP explicitly includes a discrete-time dynamical system (2) and its initial condition (3) as equality constraints. This encodes the fact that current decisions $u(k)$ impact future states $x(k+n)$ where $n > 0$.
- Second, the EMP optimizes over the state variables $x(k)$ and control variables $u(k)$, for all discrete times $k = 0, \cdots, N$. Note the states and controls are coupled by the dynamics (2).
- Third, the objective function (1) is comprised of the cost at each time instant denoted by $c_k(x_k, u_k)$, and a terminal cost denoted $c_N(x_N)$. Note that the instantaneous and terminal costs generally depend on the state and control.
- Fourth, the EMP includes inequality and equality constraints, similar to the NLP. However, they are now generally given pointwise-in-time.

In the following sections we shall discover that the EMP can be solved via the static optimization approaches of CH4, for special cases. The more general case can be solved using dynamic programming methods, which is ideally suited for the EMP.

3 Optimal HEV Energy Management

We begin our exposition of optimal control by examining the optimal energy management problem for hybrid electric vehicles (HEVs). Namely, we demonstrate how the HEV battery dynamics

can be incorporated into an optimization formulation. The goal is to determine the optimal power distribution between the engine and battery to meet driver power demand, such that total fuel consumption is minimized. In this example, we consider a very simple power flow model of HEV dynamics, which can provide a "first-order" approximation to the HEV energy management problem. Ultimately, this formulation results in a linear program.

We also assume a priori knowledge of driver power demand. That is, we somehow have clairvoyance and know power demand for all time steps $k = 0, \cdots, N$ beforehand. Besides simplifying the problem formulation, this assumption has two practical implications. First, all commercially sold vehicles are required to undergo fuel economy and emissions certification tests in which the driving schedule is standardized. As a consequence, manufacturers have been known to develop "cycle-beaters" that optimize HEV performance for pre-specified driving schedules. Second, the results provide a *benchmark* for comparing any other sub-optimal strategy. That is, performance can be assessed by its relative performance relative to the ideal case.

3.1 HEV Model

Consider the block digram in Fig. 1, which demonstrates power flow in a simple HEV model. (For pedagogical purposes, we ignore efficiency losses.) Namely, the engine produces power P_{eng}, the battery can store or release power P_{batt}, and both must provide the driver power demand P_{dem} to propel the vehicle. Consequently, we have the power conservation equality constraint

$$P_{eng}(k) + P_{batt}(k) = P_{dem}(k), \qquad \forall\, k = 0, \cdots, N \tag{9}$$

where positive P_{batt} corresponds to discharge power. The battery has energy storage dynamics given by state variable $E(k)$, which indicates the current energy (in kJ or kWh) stored in the battery. This can be formulated as

$$E(k+1) = E(k) - P_{batt}(k)\, \Delta t, \qquad \forall\, k = 0, \cdots, N-1 \tag{10}$$

$$E(0) = E_0 \tag{11}$$

where Δt is the sampled time step and E_0 is the initial battery energy level. Note that (10) represents the discrete-time battery dynamics. In addition, we must ensure net-zero energy transfer for the battery, since the battery operates as a buffer and cannot be depleted over the course of a driving event. That is, we must ensure the battery has energy buffering capacity for the next driving event. Mathematically, we write this as

$$E(N) = E(0) \tag{12}$$

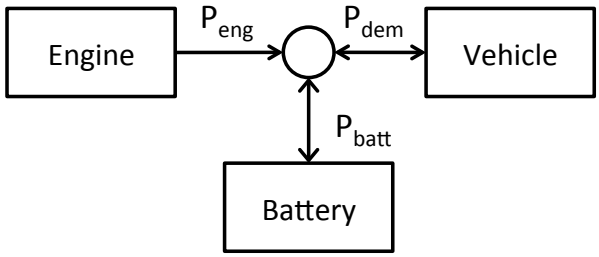

Figure 1: Block Diagram of Power Flow in Hybrid Electric Vehicle (HEV)

In reality, this will never be realized with equality exactly, so we typically relax this constraint into the following inequalities

$$0.95\ E(0) \leq E(N) \leq 1.05\ E(0) \tag{13}$$

In addition, we also assume the energy level $E(k)$, battery power $P_{batt}(k)$, and engine power $P_{eng}(k)$ are limited according to inequality constraints

$$E^{\min} \leq E(k) \leq E^{\max}, \qquad \forall\ k = 0, \cdots, N \tag{14}$$

$$P_{batt}^{\min} \leq P_{batt}(k) \leq P_{batt}^{\max}, \qquad \forall\ k = 0, \cdots, N \tag{15}$$

$$0 \leq P_{eng}(k) \leq P_{eng}^{\max}, \qquad \forall\ k = 0, \cdots, N \tag{16}$$

Recall the objective is to minimize fuel consumption. In the example we assume that fuel consumption is proportional to engine power. That is, total fuel consumption over the entire driving schedule is computed as

$$J = \sum_{k=0}^{N-1} \alpha \cdot P_{eng}(k)\ \Delta t \tag{17}$$

where α is a constant with units [g/(s-kW)], and J represents the total fuel consumption over the entire driving schedule.

3.2 LP Formulation

Collecting all the aforementioned equations produces an optimization program.

$$\min_{P_{batt}(k), P_{eng}(k), E(k)} \qquad J = \sum_{k=0}^{N-1} \alpha \cdot P_{eng}(k)\ \Delta t \tag{18}$$

with equality constraints

$$P_{eng}(k) + P_{batt}(k) \quad = \quad P_{dem}(k), \qquad \forall\ k = 0, \cdots, N \tag{19}$$

$$E(k+1) = E(k) - P_{batt}(k)\,\Delta t, \qquad \forall\, k = 0, \cdots, N-1 \tag{20}$$
$$E(0) = E_0 \tag{21}$$

and inequality constraints

$$0.95\,E(0) \leq E(N) \leq 1.05\,E(0), \tag{22}$$
$$E^{\min} \leq E(k) \leq E^{\max}, \qquad \forall\, k = 0, \cdots, N \tag{23}$$
$$P_{batt}^{\min} \leq P_{batt}(k) \leq P_{batt}^{\max}, \qquad \forall\, k = 0, \cdots, N \tag{24}$$
$$0 \leq P_{eng}(k) \leq P_{eng}^{\max}, \qquad \forall\, k = 0, \cdots, N \tag{25}$$

The decision variables are $P_{batt}(k), P_{eng}(k), E(k)$, for $k = 0, \cdots, N-1$ which are dynamically coupled by equality constraints (19)-(21).

At this point, one notices that P_{eng} can be eliminated from the program by solving (19) for P_{eng} and substituting the expression into the remaining equations/inequalities. This produces the reduced program

$$\min_{P_{batt}(k), E(k)} \qquad J = \sum_{k=0}^{N-1} \alpha \Delta t \cdot (P_{dem}(k) - P_{batt}(k)) \tag{26}$$

with equality constraints

$$E(k+1) = E(k) - P_{batt}(k)\,\Delta t, \qquad \forall\, k = 0, \cdots, N-1 \tag{27}$$
$$E(0) = E_0 \tag{28}$$

and inequality constraints

$$0.95\,E(0) \leq E(N) \leq 1.05\,E(0), \tag{29}$$
$$E^{\min} \leq E(k) \leq E^{\max}, \qquad \forall\, k = 0, \cdots, N \tag{30}$$
$$P_{batt}^{\min} \leq P_{batt}(k) \leq P_{batt}^{\max}, \qquad \forall\, k = 0, \cdots, N-1 \tag{31}$$
$$0 \leq P_{dem}(k) - P_{batt}(k) \leq P_{eng}^{\max}, \qquad \forall\, k = 0, \cdots, N \tag{32}$$

We have now obtained a linear program (LP) of the form

$$\min_{x} \qquad c^T x \tag{33}$$
$$\text{subject to:} \qquad Ax \leq b \tag{34}$$
$$A_{eq} x = b_{eq} \tag{35}$$

where the decision variable is given by

$$x = [P_{batt}(0), P_{batt}(1), \cdots, P_{batt}(N-1), E(0), E(1), \cdots, E(N-1), E(N)]^T \tag{36}$$

and we have $2N+1$ decision variables. The vectors and matrices c, A, b, A_{eq}, b_{eq} encode the remaining HEV problem parameters. Consequently, our immediate goal is to formulate c, A, b, A_{eq}, b_{eq} by properly arranging (26)-(32). It is easy to see that

$$c = [-\alpha\Delta t, \cdots, -\alpha\Delta t, \;\; 0, \ldots, 0]^T \tag{37}$$

Next we rewrite equality constraints (27)-(28) in matrix-vector form using x defined in (36).

$$\left[\begin{array}{ccccc|cccccc} \Delta t & 0 & 0 & \cdots & 0 & -1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & \Delta t & 0 & \cdots & 0 & 0 & -1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \Delta t & 0 & 0 & 0 & \cdots & -1 & 1 \\ \hline 0 & 0 & 0 & \cdots & 0 & 1 & 0 & 0 & \cdots & 0 & 0 \end{array}\right] \left[\begin{array}{c} P_{batt}(0) \\ P_{batt}(1) \\ P_{batt}(2) \\ \vdots \\ P_{batt}(N-1) \\ \hline E(0) \\ E(1) \\ E(2) \\ \vdots \\ E(N-1) \\ E(N) \end{array}\right] = \left[\begin{array}{c} 0 \\ 0 \\ \vdots \\ 0 \\ \hline E_0 \end{array}\right] \tag{38}$$

This provides matrices A_{eq} and b_{eq}. Similarly, we rewrite inequality constraints (30)-(32) in matrix-vector form:

$$\left[\begin{array}{ccccc|cccccc} 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & -1 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 1 \end{array}\right] \left[\begin{array}{c} P_{batt}(0) \\ P_{batt}(1) \\ P_{batt}(2) \\ \vdots \\ P_{batt}(N-1) \\ \hline E(0) \\ E(1) \\ E(2) \\ \vdots \\ E(N-1) \\ E(N) \end{array}\right] \leq \left[\begin{array}{c} -0.95\; E(0) \\ 1.05\; E(0) \end{array}\right] \tag{39}$$

,

$$\left[\begin{array}{ccccc|ccccc} 0 & 0 & 0 & \cdots & 0 & -1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & -1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & -1 \\ \hline 0 & 0 & 0 & \cdots & 0 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 1 \end{array}\right] \left[\begin{array}{c} P_{batt}(0) \\ P_{batt}(1) \\ P_{batt}(2) \\ \vdots \\ P_{batt}(N-1) \\ \hline E(0) \\ E(1) \\ E(2) \\ \vdots \\ E(N-1) \\ E(N) \end{array}\right] \leq \left[\begin{array}{c} -E^{\min} \\ -E^{\min} \\ \vdots \\ -E^{\min} \\ \hline E^{\max} \\ E^{\max} \\ \vdots \\ E^{\max} \end{array}\right] \tag{40}$$

,

$$\left[\begin{array}{ccccc|ccccc} -1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & -1 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & -1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ \hline 1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 & 0 & 0 & \cdots & 0 & 0 \end{array}\right] \left[\begin{array}{c} P_{batt}(0) \\ P_{batt}(1) \\ P_{batt}(2) \\ \vdots \\ P_{batt}(N-1) \\ \hline E(0) \\ E(1) \\ E(2) \\ \vdots \\ E(N-1) \\ E(N) \end{array}\right] \leq \left[\begin{array}{c} -P_{batt}^{\min} \\ -P_{batt}^{\min} \\ \vdots \\ -P_{batt}^{\min} \\ \hline P_{batt}^{\max} \\ P_{batt}^{\max} \\ \vdots \\ P_{batt}^{\max} \end{array}\right] \tag{41}$$

,

$$\left[\begin{array}{ccccc|ccccc} 1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ \hline -1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & -1 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & -1 & 0 & 0 & 0 & \cdots & 0 & 0 \end{array}\right] \left[\begin{array}{c} P_{batt}(0) \\ P_{batt}(1) \\ P_{batt}(2) \\ \vdots \\ P_{batt}(N-1) \\ \hline E(0) \\ E(1) \\ E(2) \\ \vdots \\ E(N-1) \\ E(N) \end{array}\right] \leq \left[\begin{array}{c} P_{dem}(0) \\ P_{dem}(1) \\ \vdots \\ P_{dem}(N-1) \\ \hline P_{eng}^{\max} - P_{dem}(0) \\ P_{eng}^{\max} - P_{dem}(1) \\ \vdots \\ P_{eng}^{\max} - P_{dem}(N-1) \end{array}\right] \tag{42}$$

Concatenating the matrices, we obtain A and b. Consequently, the entire HEV management problem has been encoded into matrices c, A, b, A_{eq}, b_{eq}, and can be solved with any linear program solver.

3.3 Results

We consider the power demand schedule $P_{dem}(k),\ k = 0, \cdots, N-1$ shown in Fig. 2, which has been generated from the Urban Dynamic Driving Schedule (UDDS). In the UDDS cycle, there are $N = 1369$ time-steps. As a result, our LP will have 2,739 decision variables. Also consider model parameters α = 0.1 g/(s-kW), $\Delta t = 1$ sec., E_0 = 0.6 kWh = 2.16 MJ = 50%, $E^{\min}$ = 1.296 MJ = 30%, $E^{\max}$ = 3.024 MJ = 70% $P_{batt}^{\min}$ = -15 kW, $P_{batt}^{\max}$ = +15 kW, $P_{eng}^{\max}$ = 35 kW.

The results are provided in Fig. 3. The minimum total fuel consumption is 220.8 g over the entire UDDS cycle. The battery state-of-charge (SOC) stays between 30% and 70%. In addition, the final SOC is nearly equal to the initial SOC. Finally, one may see how the battery is generating most of the effort, which is intuitive since engine power consumes fuel, which is being minimized. Moreover, both engine and battery remain within their power limits through the UDDS cycle. Finally, recall that there are 2,739 decision variables. On a MacBook Pro laptop with 2.7 GHz Intel Core i7 with 4GB of RAM, the Matlab LP solver `linprog` required only 0.2 seconds.

3.4 Remarks

Readers interested to learn more about HEV/PHEV energy management may refer to [1, 2, 8–11] and the references therein. This problem can be made more realistic in several respects: (i) model power conversion efficiencies, (ii) more detailed models of HEV powertrain components (e.g. battery, electric machines, engine), (iii) assume only statistical knowledge of the drive cycle,

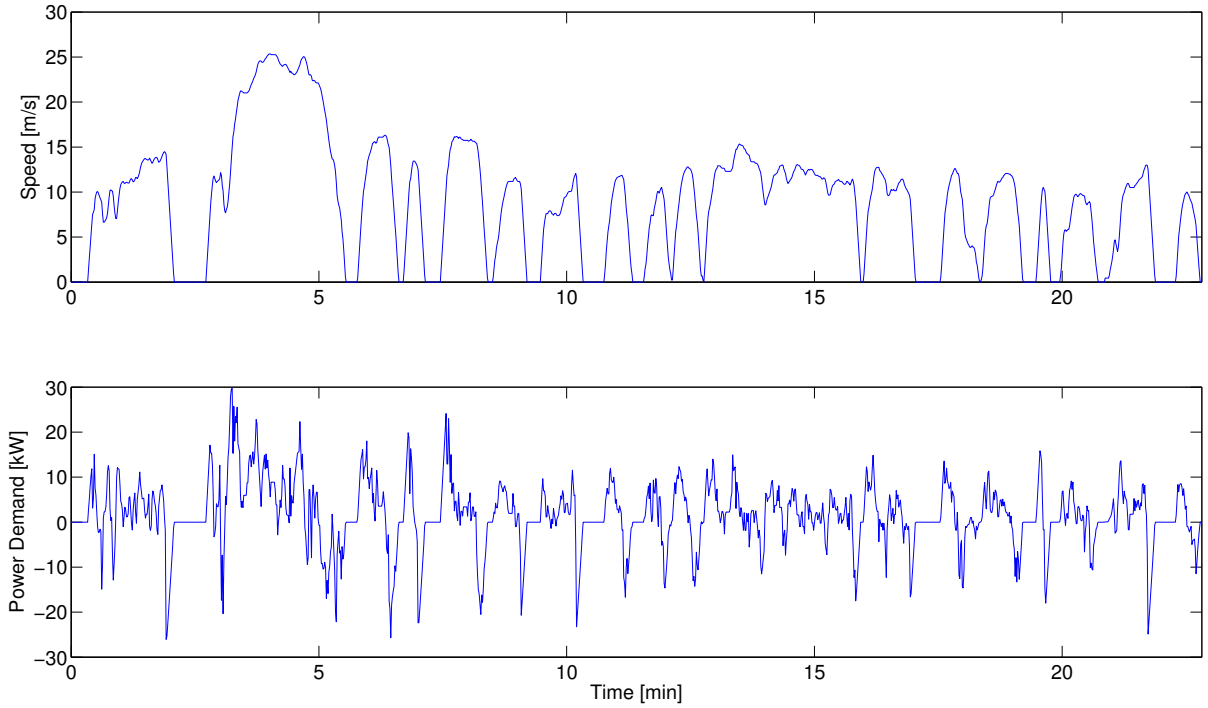

Figure 2: Power Demand for a Toyota Prius undergoing UDDS cycle.

(iv) optimize battery size, (v) optimize emissions, etc. Implementing these features will necessarily complicate the optimization formulation, and require solvers more complex than LPs.

4 Optimal PEV Charge Scheduling

Next we examine optimal control formulations for scheduling charge of plug-in electric vehicles (PEVs), under time-varying price signals. The objective is to determine a charging schedule that minimizes total electricity cost, while ensuring sufficient charge is delivered to satisfy mobility needs. We shall assume (i) advance knowledge of electricity price, and (ii) advance knowledge of the necessary energy to satisfy mobility needs. Ultimately, this formulation results in a quadratic program (QP).

4.1 PEV Battery Model

We consider the following discrete-time equivalent circuit model of a battery

$$SOC(k+1) = SOC(k) + \frac{\Delta t}{Q_{cap}} I(k), \qquad k = 0, \cdots, N-1 \tag{43}$$

$$SOC(0) = SOC_0 \tag{44}$$

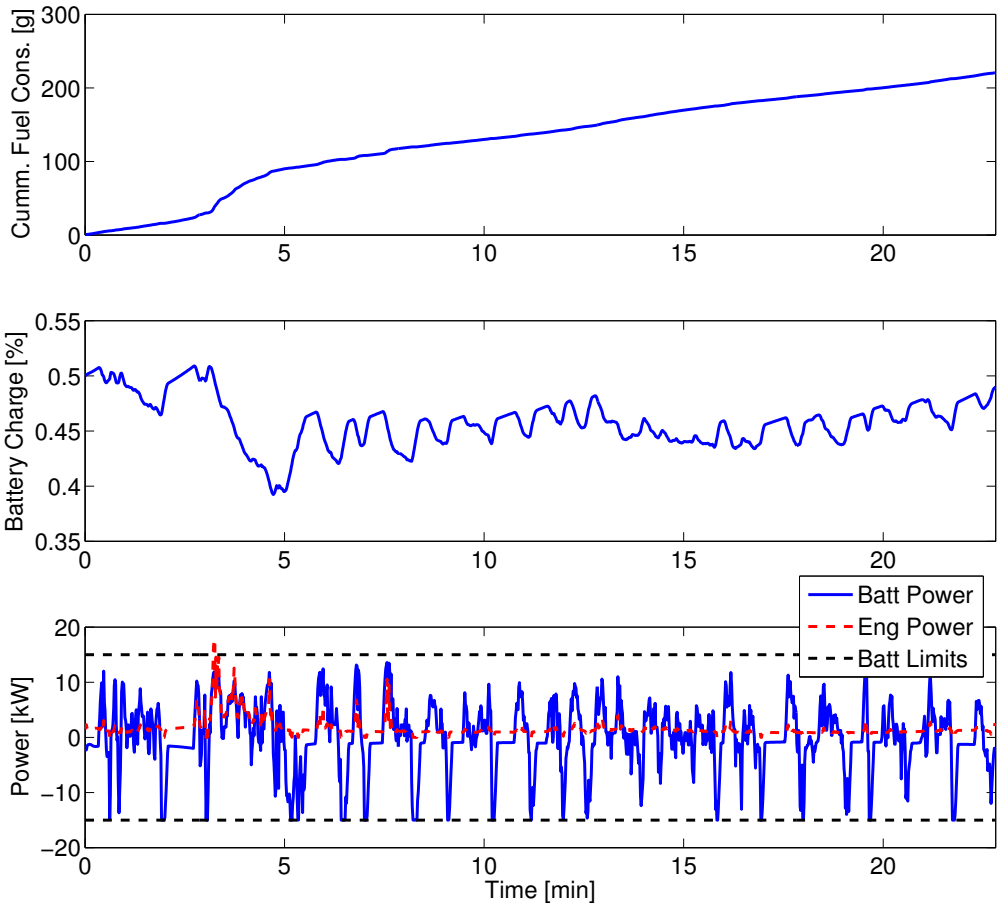

Figure 3: Results of Optimal Energy Management Problem, solved via LP. [TOP] Cumulative fuel consumption; [MIDDLE] Battery State-of-Charge (SOC); [BOTTOM] Battery and Engine Power.

$$V(k) \quad = \quad V_{oc} + R_{int} I(k) \tag{45}$$

where $SOC, \Delta t, Q_{cap}, I, SOC_0, V_{oc}, R_{int}$ are the battery state-of-charge, time-step, charge capacity [A-s], current, initial SOC, open circuit voltage, and internal resistance, respectively. Consequently, we can compute the charging power as

$$P(k) = I(k)V(k) = V_{oc} I(k) + R_{int} I^2(k) \tag{46}$$

As a result, the total charging cost is given by

$$J = \sum_{k=0}^{N-1} c(k) \Delta t V_{oc} I(k) + c(k) \Delta t R_{int} I^2(k) \tag{47}$$

where $c(k)$ is the time-varying electricity price [USD/kWh]. In addition, we have physical limits of the battery SOC and current

$$\begin{aligned} SOC^{\min} &\leq SOC(k) \leq SOC^{\max}, \qquad k = 0, \cdots, N \quad (48) \\ 0 &\leq I(k) \leq I^{\max}, \qquad k = 0, \cdots, N-1 \quad (49) \end{aligned}$$

Finally, the schedule must deliver sufficient charge to store the energy required to meet the mobility demands of the driver.

$$SOC(N) \geq \frac{E}{Q_{cap} V_{oc}} + SOC_{\min} \quad (50)$$

4.2 QP Formulation

Collecting all the aforementioned equations produces the optimization program

$$\min_{I(k), SOC(k)} \qquad J = \sum_{k=0}^{N-1} c(k) \Delta t \; V_{oc} I(k) + c(k) \Delta t R_{int} \; I^2(k) \quad (51)$$

with equality constraints

$$\begin{aligned} SOC(k+1) &= SOC(k) + \frac{\Delta t}{Q_{cap}} I(k), \qquad k = 0, \cdots, N-1 \quad (52) \\ SOC(0) &= SOC_0 \quad (53) \end{aligned}$$

and inequality constraints

$$\begin{aligned} SOC^{\min} \leq SOC(k) \leq SOC^{\max}, \qquad k = 0, \cdots, N \quad (54) \\ 0 \leq I(k) \leq I^{\max}, \qquad k = 0, \cdots, N-1 \quad (55) \\ SOC(N) \geq \frac{E}{Q_{cap} V_{oc}} + SOC_{\min} \quad (56) \end{aligned}$$

The decision variables are $I(k), SOC(k)$, for $k = 0, \cdots, N-1$ which are dynamically coupled by equality constraints (52).

We have now obtained a quadratic program (QP) of the form

$$\begin{aligned} \min_{x} \qquad & \frac{1}{2} x^T Q x + R^T x \quad (57) \\ \text{subject to:} \qquad & Ax \; \leq \; b \quad (58) \\ & A_{eq} x \; = \; b_{eq} \quad (59) \end{aligned}$$

where the decision variable is given by

$$x = [I(0), I(1), \cdots, I(N-1), SOC(0), SOC(1), \cdots, SOC(N-1), SOC(N)]^T \tag{60}$$

and we have we have $2N+1$ decision variables. The vectors and matrices $Q, R, A, b, A_{eq}, b_{eq}$ encode the remaining problem parameters. Consequently, our immediate goal is to formulate $Q, R, A, b, A_{eq}, b_{eq}$ by properly arranging (51)-(56).
It is easy to see that

$$Q = \text{diag}\left([2R_{int}\Delta t\ c(0), 2R_{int}\Delta t\ c(1) \cdots, 2R_{int}\Delta t\ c(N-1),\ \ 0, \ldots, 0]\right), \tag{61}$$

$$R = [V_{oc}\Delta t\ c(0), V_{oc}\Delta t\ c(1) \cdots, V_{oc}\Delta t\ c(N-1),\ \ 0, \ldots, 0]^T. \tag{62}$$

Next we write equality constraints (52)-(53) into matrix-vector form using x defined in (60).

$$\left[\begin{array}{ccccc|cccccc} -\frac{\Delta t}{Q_{cap}} & 0 & 0 & \cdots & 0 & -1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & -\frac{\Delta t}{Q_{cap}} & 0 & \cdots & 0 & 0 & -1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & -\frac{\Delta t}{Q_{cap}} & 0 & 0 & 0 & \cdots & -1 & 1 \\ \hline 0 & 0 & 0 & \cdots & 0 & 1 & 0 & 0 & \cdots & 0 & 0 \end{array}\right] \left[\begin{array}{c} I(0) \\ I(1) \\ I(2) \\ \vdots \\ I(N-1) \\ \hline SOC(0) \\ SOC(1) \\ SOC(2) \\ \vdots \\ SOC(N-1) \\ SOC(N) \end{array}\right] = \left[\begin{array}{c} 0 \\ 0 \\ \vdots \\ 0 \\ \hline SOC_0 \end{array}\right] \tag{63}$$

This provides matrices A_{eq} and b_{eq}. Similarly, we write inequality constraints (54)-(56) into matrix-

vector form using x defined in (60).

$$\left[\begin{array}{ccccc|ccccc} 0 & 0 & 0 & \cdots & 0 & -1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & -1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & -1 \\ \hline 0 & 0 & 0 & \cdots & 0 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 1 \end{array}\right] \left[\begin{array}{c} I(0) \\ I(1) \\ I(2) \\ \vdots \\ I(N-1) \\ \hline SOC(0) \\ SOC(1) \\ SOC(2) \\ \vdots \\ SOC(N-1) \\ SOC(N) \end{array}\right] \leq \left[\begin{array}{c} -SOC^{\min} \\ -SOC^{\min} \\ \vdots \\ -SOC^{\min} \\ \hline SOC^{\max} \\ SOC^{\max} \\ \vdots \\ SOC^{\max} \end{array}\right] \tag{64}$$

,

$$\left[\begin{array}{ccccc|ccccc} -1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & -1 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & -1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ \hline 1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 & 0 & 0 & \cdots & 0 & 0 \end{array}\right] \left[\begin{array}{c} I(0) \\ I(1) \\ I(2) \\ \vdots \\ I(N-1) \\ \hline SOC(0) \\ SOC(1) \\ SOC(2) \\ \vdots \\ SOC(N-1) \\ SOC(N) \end{array}\right] \leq \left[\begin{array}{c} 0 \\ 0 \\ \vdots \\ 0 \\ \hline I^{\max} \\ I^{\max} \\ \vdots \\ I^{\max} \end{array}\right] \tag{65}$$

,

$$\left[\begin{array}{ccccc|ccccc} 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & -1 \end{array}\right] \left[\begin{array}{c} I(0) \\ I(1) \\ I(2) \\ \vdots \\ I(N-1) \\ \hline SOC(0) \\ SOC(1) \\ SOC(2) \\ \vdots \\ SOC(N-1) \\ SOC(N) \end{array}\right] \leq -\frac{E}{Q_{cap} V_{oc}} - SOC^{\min} \tag{66}$$

Concatenating the matrices, we obtain A and b. Consequently, the entire PEV charge scheduling problem has been encoded into matrices $Q, R, A, b, A_{eq}, b_{eq}$, and can be solved with any QP solver.

4.3 Results

We consider the time-varying price $c(k),\ k = 0, \cdots, N-1$ shown in Fig. 4. In the 24 hour period, there are $N = 96$ time-steps. As a result, our QP will have 193 decision variables. Also consider model parameters Δt = 15 min, Q_{cap} = 13.8 A-hr, $V_{oc} = 363V$, R_{int} = 1.1, SOC_0 = 0.2, $SOC^{\min}$ = 0.1, $SOC^{\max}$ = 0.9, $I^{\max}$ = 9.66 A, E = 14.4 MJ. In these results we also assume the PEV is only plugged in between 16:00 and 24:00 hours.

5 Dynamic Programming

In the previous sections have we solved multi-stage decision processes via LP and QP. For these simple problems, LP and QP formulations are tractable to solve. However, one can easily imagine more complex problems that render millions of decision variables that do not have linear or quadratic structures. Consider the famous "traveling salesmen" problem shown in Fig. 5. The goal is to find the shortest path to loop through N cities, ending at the origin city. Due to the number of constraints, possible decision variables, and nonlinearity of the problem structure, the traveling salesmen problem is notoriously difficult to solve.

It turns out that a more efficient solution method exists, specifically designed for multi-stage decision processes, known as dynamic programming. The basic premise is to break the problem into simpler subproblems. This structure is inherent in multi-decision processes.

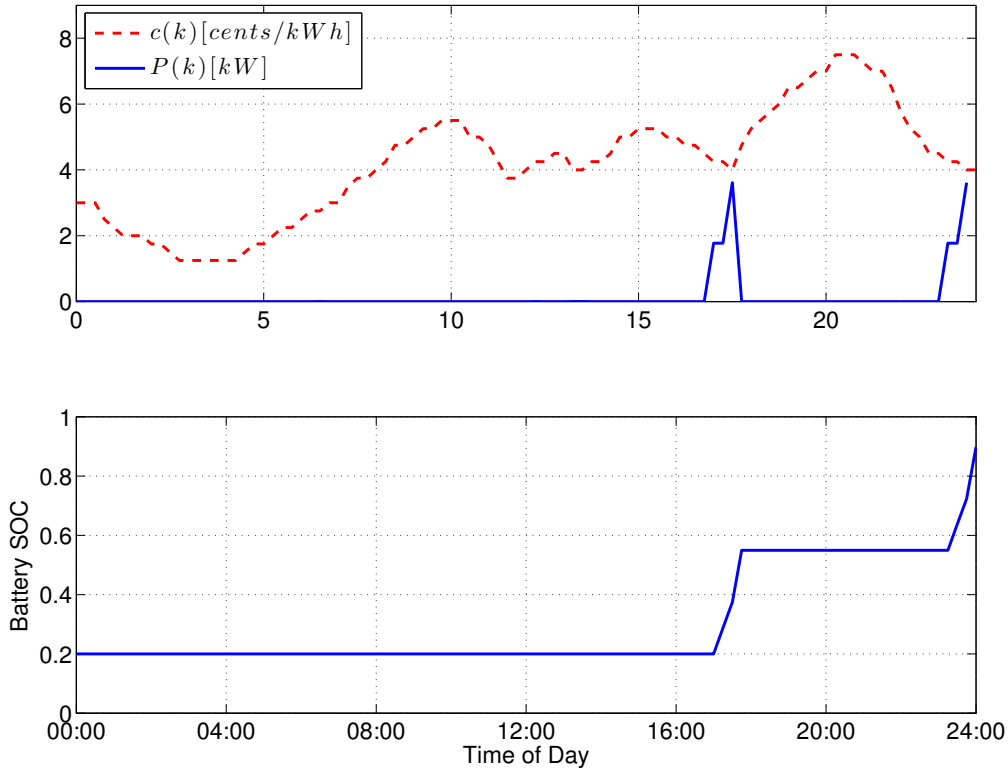

Figure 4: Results for Optimal PEV Charge Schedule.

5.1 Principle of Optimality

Consider a multi-stage decision process, i.e. an equality constrained NLP with dynamics

$$\min_{x(k),u(k)} \quad J = \sum_{k=0}^{N-1} c_k(x_k, u_k) + c_N(x_N) \tag{67}$$

$$\text{s. to} \quad x_{k+1} = f(x_k, u_k), \qquad k = 0, 1, \cdots, N-1 \tag{68}$$

$$x_0 = x_{init} \tag{69}$$

where k is the discrete time index, x_k is the state at time k, u_k is the control decision applied at time k, N is the time horizon, $c_k(\cdot,\cdot)$ is the instantaneous cost, and $c_N(\cdot)$ is the final or terminal cost.

In words, the principle of optimality is the following. Assume at time step k, you know all the future optimal decisions, i.e. $u^*(k+1), u^*(k+2), \cdots, u^*(N-1)$. Then you may compute the best solution for the current time step, and pair with the future decisions. This can be done recursively by starting from the end $N-1$, and working your way backwards.

Mathematically, the principle of optimality can be expressed precisely as follows. Define $V_k(x_k)$

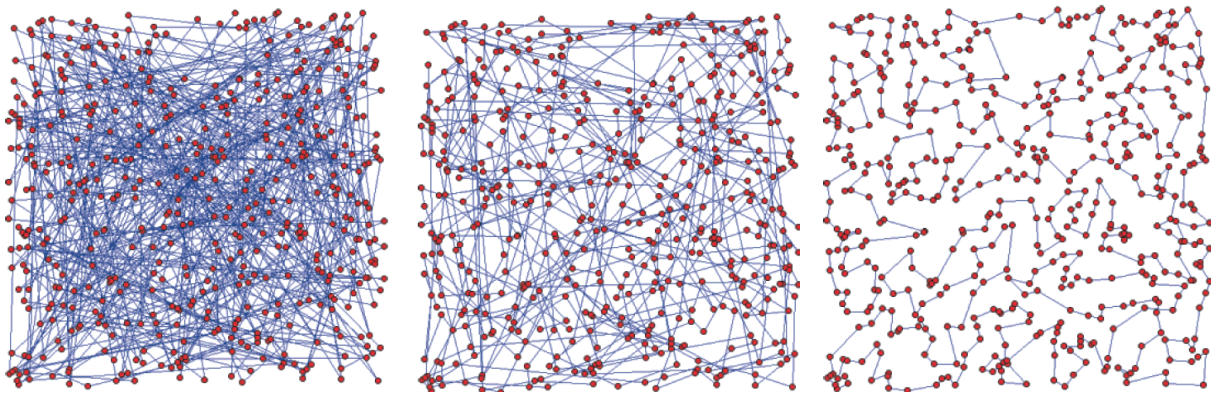

Figure 5: Random (left), suboptimal (middle), and optimal solutions (right).

as the optimal "cost-to-go" (a.k.a. "value function") from time step k to the end of the time horizon N, given the current state is x_k. Then the principle of optimality can be written in recursive form as:

$$V_k(x_k) = \min_{u_k} \left\{ c_k(x_k, u_k) + V_{k+1}(x_{k+1}) \right\}, \qquad k = 0, 1, \cdots, N-1 \tag{70}$$

with the boundary condition

$$V_N(x_N) = c_N(x_N) \tag{71}$$

The admittedly awkward aspects are:

1. You solve the problem backward!
2. You solve the problem recursively!

Let us illustrate this with an example.

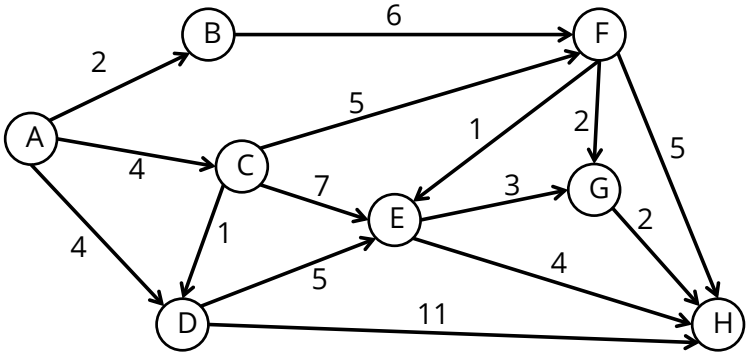

Figure 6: Network for shortest path problem in Example 5.1.

Example 5.1 (Shortest Path Problem)**.** Consider the network shown in Fig. 6. The goal is to find the shortest path from node A to node H, where path length is indicated by the edge numbers.

Let us define the cost-to-go as $V(i)$. That is, $V(i)$ is the shortest path length from node i to node H. For example $V(H) = 0$. Let $c(i,j)$ denote the cost of traveling from node i to node j. For example, $c(C,E) = 7$. Then $c(i,j) + V(j)$ represents the cost of traveling from node i to node j, and then from node j to H along the shortest path. This enables us to write the principle of optimality equation and boundary conditions:

$$V(i) = \min_{j \in N_i^d} \{c(i,j) + V(j)\} \tag{72}$$

$$V(H) = 0 \tag{73}$$

where the set N_i^d represents the nodes that descend from node i. For example $N_C^d = \{D, E, F\}$. We can solve these equations recursively, starting from node H and working our way backward to node A as follows:

$$\begin{aligned}
V(G) &= c(G,H) + V(H) = 2 + 0 = 2 \\
V(E) &= \min\{c(E,G) + V(G), c(E,H) + V(H)\} = \min\{3+2, 4+0\} = 4 \\
V(F) &= \min\{c(F,G) + V(G), c(F,H) + V(H), c(F,E) + V(E)\} \\
&= \min\{2+2, 5+0, 1+4\} = 4 \\
V(D) &= \min\{c(D,E) + V(E), c(D,H) + V(H)\} = \min\{5+4, 11+0\} = 9 \\
V(C) &= \min\{c(C,F) + V(F), c(C,E) + V(E), c(C,D) + V(D)\} \\
&= \min\{5+4, 7+4, 1+9\} = 9 \\
V(B) &= c(B,F) + V(F) = 6 + 4 = 10 \\
V(A) &= \min\{c(A,B) + V(B), c(A,C) + V(C), c(A,D) + V(D)\} \\
&= \min\{2+10, 4+9, 4+9\} = 12
\end{aligned}$$

Consequently, we arrive at the optimal path A $\rightarrow$ B $\rightarrow$ F $\rightarrow$ G $\rightarrow$ H.

Example 5.2 (Optimal Consumption and Saving)**.** This example is popular among economists for learning dynamic programming, since it can be solved by hand. Consider a consumer who lives over periods $k = 0, 1, \cdots, N$ and must decide how much of a resource they will consume or save during each period.

Let u_k be the consumption in each period and assume consumption yields utility $\ln(u_k)$ over each period. The natural logarithm function models a "diseconomies of scale" in marginal value when increasing resource consumption. Let x_k denote the resource level in period k, and x_0 denote the initial resource level. At any given period, the resource level in the next period is given by $x_{k+1} = x_k - u_k$. We also constrain the resource level to be non-negative. The consumer's

decision problem can be written as

$$\max_{x_k,u_k} \quad J = \sum_{k=0}^{N-1} \ln(u_k) \tag{74}$$

$$\text{s. to} \quad x_{k+1} = x_k - u_k, \qquad k = 0, 1, \cdots, N-1 \tag{75}$$

$$x_k \geq 0, \qquad k = 0, 1, \cdots, N \tag{76}$$

Note that the objective function is not linear nor quadratic in decision variables x_k, u_k. It is, in fact, concave in u_k. The equivalent minimization problem would be $\min_{x_k,u_k} -\ln(u_k)$ which is convex in u_k. Moreover, all constraints are linear. Consequently, convex programming is one solution option. Dynamic programming is another. In general DP does not require the convex assumptions, and – in this case – can solve the problem analytically.

First we define the value function. Let $V_k(x_k)$ denote the maximum total utility from time step k to terminal time step N, where the resource level in step k is x_k. Then the principle of optimality equations can be written as:

$$V_k(x_k) = \max_{u_k \leq x_k} \left\{ \ln(u_k) + V_{k+1}(x_{k+1}) \right\}, \qquad k = 0, 1, \cdots, N-1 \tag{77}$$

with the boundary condition that represents zero utility can be accumulated after the last time step.

$$V_N(x_N) = 0 \tag{78}$$

We now solve the DP equations starting from the last time step and working backward. Consider $k = N-1$,

$$\begin{aligned} V_{N-1}(x_{N-1}) &= \max_{u_{N-1} \leq x_{N-1}} \left\{ \ln(u_{N-1}) + V_N(x_N) \right\} \\ &= \max_{u_{N-1} \leq x_{N-1}} \left\{ \ln(u_{N-1}) + 0 \right\} \\ &= \ln(x_{N-1}) \end{aligned}$$

In words, the optimal action is to consume all remaining resources, $u^*_{N-1} = x_{N-1}$. Moving on to $k = N-2$,

$$\begin{aligned} V_{N-2}(x_{N-2}) &= \max_{u_{N-2} \leq x_{N-2}} \left\{ \ln(u_{N-2}) + V_{N-1}(x_{N-1}) \right\} \\ &= \max_{u_{N-2} \leq x_{N-2}} \left\{ \ln(u_{N-2}) + V_{N-1}(x_{N-2} - u_{N-2}) \right\} \\ &= \max_{u_{N-2} \leq x_{N-2}} \left\{ \ln(u_{N-2}) + \ln(x_{N-2} - u_{N-2}) \right\} \\ &= \max_{u_{N-2} \leq x_{N-2}} \ln\left(u_{N-2}(x_{N-2} - u_{N-2}) \right) \\ &= \max_{u_{N-2} \leq x_{N-2}} \ln\left(x_{N-2}u_{N-2} - u^2_{N-2} \right) \end{aligned}$$

Since $\ln(\cdot)$ is a monotonically increasing function, maximizing its argument will maximize its value. Therefore, we focus on finding the maximum of the quadratic function w.r.t. u_{N-2} embedded inside the argument of $\ln(\cdot)$. It's straight forward to find $u^*_{N-2} = \frac{1}{2}x_{N-2}$. Moreover, $V_{N-2}(x_{N-2}) = \ln(\frac{1}{4}x^2_{N-2})$. Continuing with $k = N-3$,

$$\begin{aligned} V_{N-3}(x_{N-3}) &= \max_{u_{N-3}\leq x_{N-3}} \{\ln(u_{N-3}) + V_{N-2}(x_{N-2})\} \\ &= \max_{u_{N-3}\leq x_{N-3}} \{\ln(u_{N-3}) + V_{N-2}(x_{N-3} - u_{N-3})\} \\ &= \max_{u_{N-3}\leq x_{N-3}} \left\{\ln(u_{N-3}) + \ln\left(\frac{1}{4}(x_{N-3} - u_{N-3})^2\right)\right\} \\ &= \max_{u_{N-3}\leq x_{N-3}} \left\{\ln(u_{N-3}) + \ln\left((x_{N-3} - u_{N-3})^2\right)\right\} - \ln(4) \\ &= \max_{u_{N-3}\leq x_{N-3}} \ln\left(x^2_{N-3}u_{N-3} - 2x_{N-3}u^2_{N-3} + u^3_{N-3}\right) - \ln(4) \end{aligned}$$

Again, we can focus on maximizing the argument of $\ln(\cdot)$. It's simple to find that $u^*_{N-3} = \frac{1}{3}x_{N-3}$. Moreover, $V_{N-3}(x_{N-3}) = \ln(\frac{1}{27}x^3_{N-3})$. At this point, we recognize the pattern

$$u^*_k = \frac{1}{N-k}x_k, \qquad k = 0, \cdots, N-1 \tag{79}$$

One can use induction to prove this hypothesis indeed solves the recursive principle of optimality equations. Equation (80) provides the optimal state feedback policy. That is, the optimal policy is written as a function of the current resource level x_k. If we write the optimal policy in open-loop form, it turns out the optimal consumption is the same at each time step. Namely, it is easy to show that (80) emits the policy

$$u^*_k = \frac{1}{N}\, x_0, \qquad \forall\, k = 0, \cdots, N-1 \tag{80}$$

As a consequence, the optimal action is to consume that same amount of resource at each time-step. It turns out one should consume $1/N \cdot x_0$ at each time step if they are to maximize total utility.

Remark 5.1. This is a classic example in resource economics. In fact, this example represents the non-discounted, no interest rate version of Hotelling's Law, a theorem in resource economics [12]. Without a discount/interest rate, any difference in marginal benefit could be arbitraged to increase net benefit by waiting until the next time-step. Embellishments of this problem get more interesting where there exists uncertainty about resource availability, extraction cost, future benefit, and interest rate. In fact, oil companies use this exact analysis to determine when to drill an oil field, and how much to extract.

6 Smart Appliance Scheduling

In this section, we utilize dynamic programming principles to schedule a smart dishwasher appliance. This is motivated by the vision of future homes with smart appliances. Namely, internet-connected appliances with local computation will be able to automate their procedures to minimize energy consumption, while satisfying homeowner needs.

Consider a smart dishwasher that has five cycles, indicated in Table 1. Assume each cycle requires 15 minutes. Moreover, each cycle must be run in order, possibly with idle periods in between. We also consider electricity price which varies in 15 minute periods, as shown in Fig. 7. The goal is to find the cheapest cycle schedule starting at 17:00 and ending at 24:00, with the requirement that the dishwasher completes all of its cycles by 24:00 midnight.

6.1 Problem Formulation

Let us index each 15 minute time period by k, where $k = 0$ corresponds to 17:00 – 17:15, and $k = N = 28$ corresponds to 24:00–00:15. Let us denote the dishwasher state by $x_k \in \{0, 1, 2, 3, 4, 5\}$, which indicates the last completed cycle at the very beginning of each time period. The initial state is $x_0 = 0$. The control variable $u_k \in \{0, 1\}$ corresponds to either wait, $u_k = 0$, or continue to the next cycle, $u_k = 1$. We assume control decisions are made at the beginning of each period, and cost is accrued during that period. Then the state transition function, i.e. the dynamical relation, is given by

$$x_{k+1} = x_k + u_k, \qquad k = 0, \cdots, N-1 \tag{81}$$

Let c_k represent the time varying cost in units of USD/kWh. Let $p(x_k)$ represent the power required for cycle x_k, in units of kW. We are now positioned to write the optimization program

$$\begin{aligned} \min \quad & \sum_{k=0}^{N-1} \frac{1}{4} c_k \; p(x_{k+1}) \; u_k \\ \text{s. to} \quad & x_{k+1} = x_k + u_k, \qquad k = 0, \cdots, N-1 \\ & x_0 = 0, \\ & x_N = 5, \\ & u_k \in \{0, 1\}, \qquad k = 0, \cdots, N-1 \end{aligned}$$

6.2 Principle of Optimality

Next we formulate the dynamic programming equations. The cost-per-time-step is given by

$$g_k(x_k, u_k) \;\; = \;\; \frac{1}{4} c_k \; p(x_{k+1}) \; u_k, \tag{82}$$

$$= \frac{1}{4} c_k \, p(x_k + u_k) \, u_k, \qquad k = 0, \cdots, N-1 \tag{83}$$

Since we require the dishwasher to complete all cycles by 24:00, we define the following terminal cost:

$$g_N(x_N) = \begin{cases} 0 & : x_N = 5 \\ \infty & : \text{otherwise} \end{cases} \tag{84}$$

Let $V_k(x_k)$ represent the minimum cost-to-go from time step k to final time period N, given the last completed dishwasher cycle is x_k. Then the principle of optimality equations are:

$$\begin{aligned} V_k(x_k) &= \min_{u_k \in \{0,1\}} \left\{ \frac{1}{4} c_k \, p(x_k + u_k) u_k + V_{k+1}(x_{k+1}) \right\} \\ &= \min_{u_k \in \{0,1\}} \left\{ \frac{1}{4} c_k \, p(x_k + u_k) u_k + V_{k+1}(x_k + u_k) \right\} \\ &= \min \left\{ V_{k+1}(x_k), \;\; \frac{1}{4} c_k \, p(x_k + 1) + V_{k+1}(x_k + 1) \right\} \end{aligned} \tag{85}$$

with the boundary condition

$$V_N(5) = 0, \qquad V_N(i) = \infty \;\; \text{for} \;\; i \neq 5 \tag{86}$$

We can also write the optimal control action as:

$$u^*(x_k) = \arg \min_{u_k \in \{0,1\}} \left\{ \frac{1}{4} c_k \, p(x_k + u_k) u_k + V_{k+1}(x_k + u_k) \right\}$$

Equation (85) is solved recursively, using the boundary condition (86) as the first step. Next, we show how to solve this algorithmically in Matlab.

6.3 Matlab Implementation

The code below provides an implementation of the dynamic programming equations.

```
%% Problem Data
% Cycle power
p = [0; 1.5; 2.0; 0.5; 0.5; 1.0];

% Electricity Price Data
c = [12,12,12,10,9,8,8,8,7,7,6,5,5,5,5,5,5,6,7,7,8,9,9,10,11,11,...
     12,12,14,15,15,16,17,19,19,20,21,21,22,22,22,20,20,19,17,15,15,16,...
     17,17,18,18,16,16,17,17,18,20,20,21,21,21,20,20,19,19,18,17,17,...
     16,19,21,22,23,24,26,26,27,28,28,30,30,30,29,28,28,26,23,21,20,18,18,17,17,16,16];

```

	cycle	power
1	prewash	1.5 kW
2	main wash	2.0 kW
3	rinse 1	0.5 kW
4	rinse 2	0.5 kW
5	dry	1.0 kW

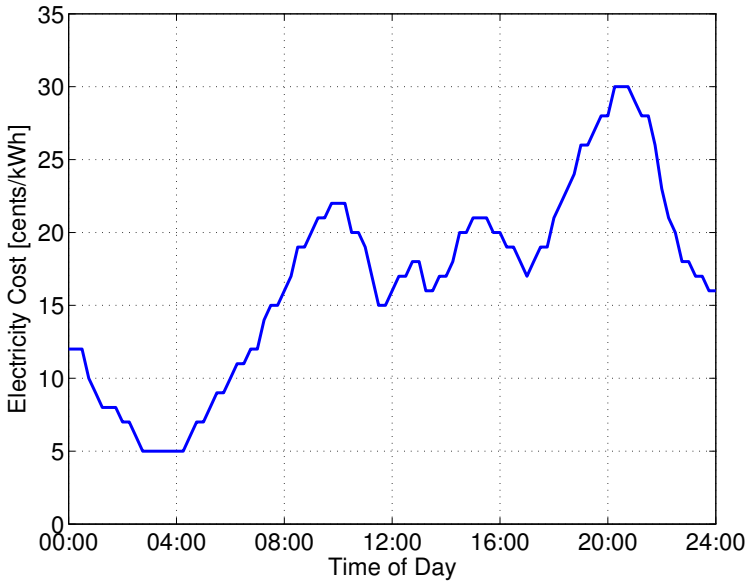

Figure 7 & Table 1: [LEFT] Dishwasher cycles and corresponding power consumption. [RIGHT] Time-varying electricity price. The goal is to determine the dishwasher schedule between 17:00 and 24:00 that minimizes the total cost of electricity consumed.

```
%% Solve DP Equations
% Time Horizon
N = 28;
% Number of states
nx = 6;

% Preallocate Value Function
V = inf*ones(N,nx);
% Preallocate control policy
u = nan*ones(N,nx);

% Boundary Condition
V(end, end) = 0;

% Iterate through time backwards
for k = (N-1):-1:1;

    % Iterate through states
    for i = 1:nx

        % If you're in last state, can only wait
        if(i == nx)
            V(k,i) = V(k+1,i);

```

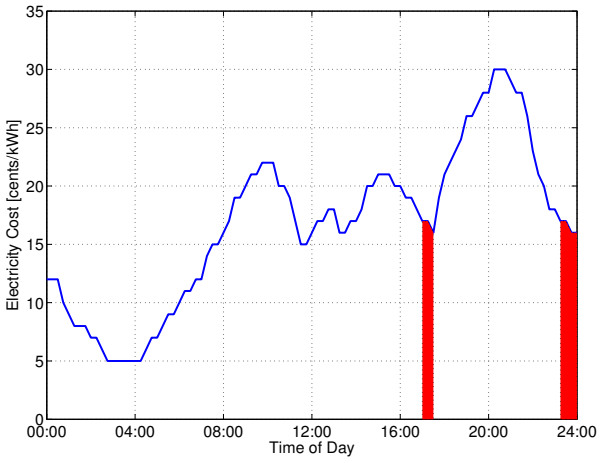

Figure 8: The optimal dishwasher schedule is to run cycles at 17:00, 17:15, 23:15, 23:30, 23:45. The minimum total cost of electricity is 22.625 cents.

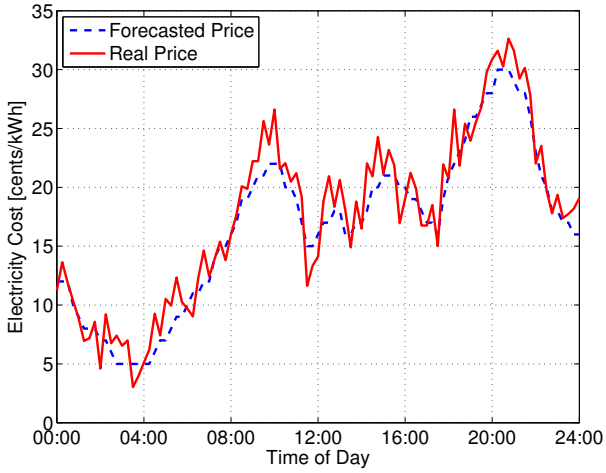

Figure 9: The true electricity price c_k can be abstracted as a forecasted price plus random uncertainty.

```
35          % Otherwise, solve Principle of Optimality
36          else
37                                  %Choose  u=0 ; u=1
38              [V(k,i),idx] = min([V(k+1,i); 0.25*c(69+k)*p(i+1) + V(k+1,i+1)]);
39
40              % Save minimizing control action
41              u(k,i) = idx-1;
42          end
43      end
44  end
```

Note the value function is solved backward in time (line 26), and for each state (line 29). The principle of optimality equation is implemented in line 38, and the optimal control action is saved in line 41. The variable `u(k,i)` ultimately provides the optimal control action as a function of time step k and dishwasher state i, namely $u_k^* = u^*(k, x_k)$.

6.4 Results

The optimal dishwasher schedule is depicted in Fig. 8, which exposes how cycles are run in periods of low electricity cost c_k. Specifically, the dishwasher begins prewash at 17:00, main wash at 17:15, rinse 1 at 23:15, rinse 2 at 23:30, and dry at 23:45. The total cost of electricity consumed is 22.625 cents.

6.5 Stochastic Dynamic Programming

The example above assumed the electricity price c_k for $k = 0, \cdots, N$ is known exactly a priori. In reality, the smart appliance may not know this price signal exactly, as demonstrated by Fig. 9. However, we may be able to anticipate it by forecasting the price signal, based upon previous data. We now seek to relax the assumption of perfect a priori knowledge of c_k. Instead, we assume that c_k is forecasted using some method (e.g. machine learning, neural networks, Markov chains) with some error with known statistics..

We shall now assume the true electricity cost is given by

$$c_k = \hat{c}_k + w_k \qquad , k = 0, \cdots, N-1 \tag{87}$$

where $\hat{c}_k$ is the forecasted price that we anticipate, and w_k is a random variable representing uncertainty between the forecasted value and true value. We additionally assume knowledge of the mean uncertainty, namely $\mathbb{E}[w_k] = \overline{w}_k$ for all $k = 0, \cdots, N$. That is, we have some knowledge of the forecast quality, quantified in terms of mean error.

Armed with a forecasted cost and mean error, we can formulate a stochastic dynamic programming (SDP) problem:

$$\begin{aligned}
\min \quad & J = \mathbb{E}_{w_k}\left[\sum_{k=0}^{N-1} \frac{1}{4}(\hat{c}_k + w_k)\, p(x_{k+1})\, u_k\right] \\
\text{s. to} \quad & x_{k+1} = x_k + u_k, \qquad k = 0, \cdots, N-1 \\
& x_0 = 0, \\
& x_N = 5, \\
& u_k \in \{0, 1\}, \qquad k = 0, \cdots, N-1
\end{aligned}$$

where the critical difference is the inclusion of w_k, a stochastic term. As a result, we seek to minimize the *expected* cost, w.r.t. to random variable w_k.

We now formulate the principle of optimality. Let $V_k(x_k)$ represent the expected minimum cost-to-go from time step k to N, given the current state x_k. Then the principle of optimality equations can be written as:

$$\begin{aligned}
V_k(x_k) &= \min_{u_k} \mathbb{E}\left\{g_k(x_k, u_k, w_k) + V_{k+1}(x_{k+1})\right\} \\
&= \min_{u_k \in \{0,1\}} \left\{\mathbb{E}\left[\frac{1}{4}(\hat{c}_k + w_k)\, p(x_{k+1}) u_k\right] + V_{k+1}(x_{k+1})\right\} \\
&= \min_{u_k \in \{0,1\}} \left\{\frac{1}{4}(\hat{c}_k + \overline{w}_k)\, p(x_{k+1}) u_k + V_{k+1}(x_k + u_k)\right\} \\
&= \min\left\{V_{k+1}(x_k),\ \ \frac{1}{4}(\hat{c}_k + \overline{w}_k)\, p(x_k + 1) + V_{k+1}(x_k + 1)\right\}
\end{aligned}$$

with the boundary condition

$$V_N(5) = 0, \qquad V_N(i) = \infty \text{ for } i \neq 5$$

These equations are deterministic, and can be solved exactly as before. The crucial detail is that we have incorporated uncertainty by incorporating a forecasted cost with uncertain error. As a result, we seek to minimize expected cost.

7 Stochastic Optimal Control

In renewable energy systems, such as solar, wind, and tidal power, the resource is fundamentally intermittent. This complicates the energy management problem, since the available generator power is unknown a priori. It is therefore impossible to achieve optimal energy management. However, we often have *statistics* about the available power. With appropriate formalisms to describe these statistics, we can integrate them into the optimal control formulation.

This concept is described schematically in Fig. 10. Namely, we abstract the intermittent solar, wind, tidal, etc. power as a random process, characterized by random state variable X_k. In Section 7.1, we introduce Markov chains as one way to model these random processes (see Ch. 12 of [13] for an in-depth exposition). In Section 7.2, we described how to incorporate Markov chain models into the dynamic programming formulation. This renders a so-called stochastic dynamic program (SDP).

7.1 Markov Chain Models

A Markov chain model is a dynamic system that undergoes transitions from one state to another on a state-space. Unlike deterministic dynamic systems $x_{k+1} = f(x_k)$, the process is random and each transition is characterized by statistics. Moreover, it contains the Markov property. The Markov property is that given the present state, the future and past states are independent. This

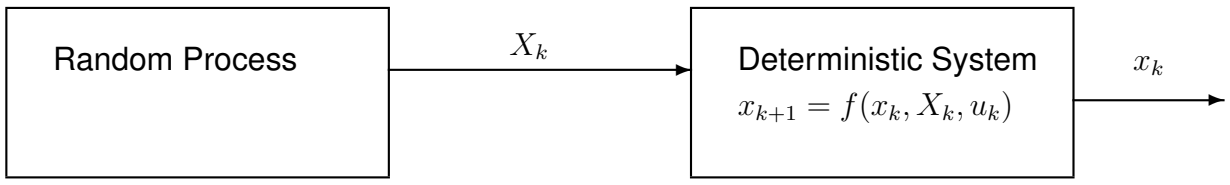

Figure 10: Block diagram of energy system with stochastic input generated from random process.

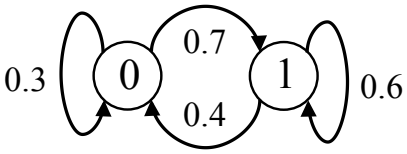

Figure 11: State transition diagram for simple Markov chain.

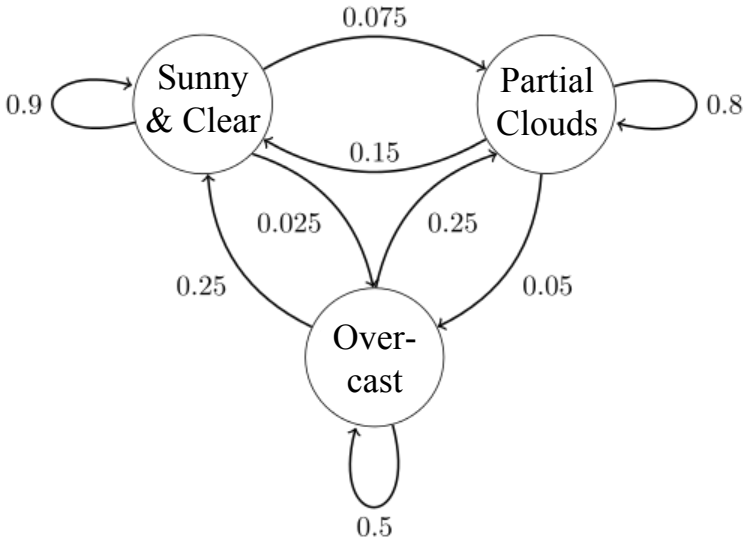

Figure 12: State-transition diagram for solar irradiation: High = sunny & clear; Medium = partial clouds; Low = overcast.

can be written mathematically as

$$\Pr\left[X_{k+1} = i_{k+1} | X_k = i_k, X_{k-1} = i_{k-1}, \cdots, X_0 = i_0\right] = \Pr\left[X_{k+1} = i_{k+1} | X_k = i_k\right] \tag{88}$$

where X_k is a random variable at time-step k and i_k is a value for the state. In other words, given the sequence of states $i_0, i_1, \cdot, i_k$, the condition probability of what X_{k+1} will be depends only on the value i_k.

Markov chains are often described by state transition diagram (i.e. directed graphs), such as Fig. 11, and probability transition matrices. Let us define the transition probability as

$$p_{ij} = \Pr\left[X_{k+1} = j | X_k = i\right] \tag{89}$$

In this example, $X_k \in \{0, 1\}$. Then p_{ij} provides the elements of the probability transition matrix, for row i and column j. That is

$$P = [p_{ij}] = \begin{bmatrix} 0.3 & 0.7 \\ 0.4 & 0.6 \end{bmatrix} \tag{90}$$

Note the row sum of a Markov chain transition matrix always sums to one. That is, the sum of all the probabilities leaving a state must be one:

$$\sum_j p_{ij} = \sum_j \Pr\left[X_{k+1} = j | X_k = i\right] = 1 \tag{91}$$

Example 7.1 (Solar Irradiation)**.** The state transition diagram in Fig. 12 represents the hypothetical

stochastic dynamics of solar irradiation, for a photovoltaic generator. According to the figure, a sunny & clear day is followed by another sunny & clear day 90% of the time, a partially cloudy day 7.5% of the time, and an overcast day 2.5% of the time. Labelling the state space $\{$1 = sunny & clear, 2 = partial clouds, 3 = overcast $\}$ the state transition matrix for this example is

$$P = \begin{bmatrix} 0.9 & 0.075 & 0.025 \\ 0.15 & 0.8 & 0.05 \\ 0.25 & 0.25 & 0.5 \end{bmatrix} \tag{92}$$

The distribution over states can be written as a stochastic row vector x with the relation $x^{(k+1)} = x^{(k)}P$. So if at time k the system is in state 2 (partial clouds), then three time periods later, at time $k+3$ the distribution is

$$\begin{aligned} x^{(k+3)} &= x^{(k+2)}P = \left(x^{(k+1)}P\right)P && (93)\\ &= x^{(k+1)}P^2 = \left(x^{(k)}P^2\right)P && (94)\\ &= x^{(k)}P^3 && (95)\\ &= \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0.9 & 0.075 & 0.025 \\ 0.15 & 0.8 & 0.05 \\ 0.25 & 0.25 & 0.5 \end{bmatrix}^3 && (96)\\ &= \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0.7745 & 0.17875 & 0.04675 \\ 0.3575 & 0.56825 & 0.07425 \\ 0.4675 & 0.37125 & 0.16125 \end{bmatrix} && (97)\\ &= \begin{bmatrix} 0.3575 & 0.56825 & 0.07425 \end{bmatrix}. && (98) \end{aligned}$$

In other words, if the sky is currently partially cloudy, then in three days the probability of a sunny day is 36%, a partially cloudy day is 57%, and an overcast day is 7.4%. Using the transition matrix it is possible to calculate, for example, the long-term fraction of days during which the sky is partially cloudy, or the average number of days it will take to go from a overcast day to a sunny day. Using the transition probabilities, the steady-state probabilities indicate that 62.5% of days will be sunny, 31.25% of days will be partially cloudy and 6.25% of days will be overcast, since:

$$\lim_{N\to\infty} P^N = \begin{bmatrix} 0.625 & 0.3125 & 0.0625 \\ 0.625 & 0.3125 & 0.0625 \\ 0.625 & 0.3125 & 0.0625 \end{bmatrix} \tag{99}$$

Reducibility: A state j is said to be accessible from state i (written $i \to j$) if a system started in state i has a non-zero probability of transitioning into state j at some point. Formally, state j is

accessible from state i if

$$\Pr\left[X_n = j | X_0 = i\right] > 0 \tag{100}$$

A state i is said to communicate with state j (written $i \leftrightarrow j$) if both $i \to j$ and $j \to i$. A set of states $\mathcal{C}$ is a communicating class if every pair of states in $\mathcal{C}$ communicates with each other, and no state in $\mathcal{C}$ communicates with any state not in $\mathcal{C}$. We call such communicating classes closed. A Markov chain is said to be **irreducible** if its state space is a single communicating class; in other words, it is possible to get from any state to any other state in the state-space, but it is not possible to leave the state-space.

Stationary distribution: If the Markov chain is a time-homogeneous (i.e. time-invariant) Markov chain, so that the process is described by a single time-independent matrix $P = [p_{ij}]$, then the vector π is called a stationary distribution if $\forall\ j \in \mathcal{S}$ it satisfies

$$0 \leq \pi_j \leq 1, \qquad \sum_{j \in \mathcal{S}} \pi_j = 1, \qquad \pi_j = \sum_{i \in \mathcal{S}} \pi_i p_{ij}. \tag{101}$$

The last equality can be written in matrix-vector form as

$$\pi = \pi P \tag{102}$$

which implies the stationary distribution π is the left eigenvector of P corresponding to the eigenvalue $\lambda = 1$. Consequently, stationary distributions are automatically determined when solving for the eigenvalues/eigenvectors of P.

7.2 Stochastic Dynamic Programming

Armed with the Markov chain modeling framework to incorporate statistics of the random process (e.g. solar irradiation, wind speed, homeowner power demand), we can now formulate a stochastic dynamic program (SDP). Consider the block diagram in Fig. 13. The deterministic system is given by the upper-right block, and is characterized by state x_k. The environment imposes some random variable X_k onto the system, described by the Markov chain model in the left block. The engineering design problem is to determine the control input u_k which minimizes some objective. The control will be synthesized as a control law. Namely, the control is the output of a mapping that depends on the current deterministic state x_k and stochastic state X_k.

We formalize this as a stochastic multi-stage decision process,

$$\min_{x_k, X_k, u_k} \quad J = \mathbb{E}\left[\sum_{k=0}^{N-1} g_k(x_k, X_k, u_k) + g_N(x_N)\right] \tag{103}$$

$$\text{s. to} \quad x_{k+1} = f(x_k, X_k = i, u_k), \qquad k = 0, 1, \cdots, N-1, \quad i \in \mathcal{S} \tag{104}$$

$$x_0 = x_0 \tag{105}$$

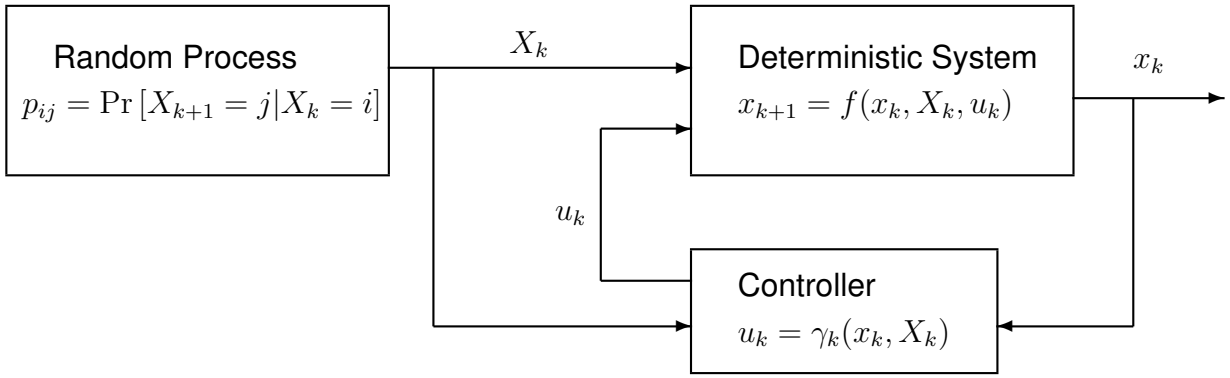

Figure 13: Block diagram of stochastic multi-stage decision process.

$$p_{ij} = \Pr[X_{k+1} = j | X_k = i], \qquad k = 0, 1, \cdots, N-1, \quad i, j \in \mathcal{S} \tag{106}$$

$$X_0 = i_0 \tag{107}$$

where k is the discrete time index, x_k is the deterministic state at time k, X_k is the stochastic state at time k, u_k is the control decision applied at time k, N is the time horizon, $g_k(\cdot,\cdot,\cdot)$ is the instantaneous cost, and $g_N(\cdot)$ is the final or terminal cost.

Now we define the value function. Let $V_k(x_k, X_k)$ be the expected cost-to-go from time step k to N, given the current states are x_k, X_k. The principle of optimality equation is

$$V_k(x_k, X_k) = \min_{u_k} \{g(x_k, X_k, u_k) + \mathbb{E}\, V_{k+1}(x_{k+1}, X_{k+1})\}, \qquad k = 0, 1, \cdots, N-1 \tag{108}$$

$$= \min_{u_k} \{g(x_k, X_k, u_k) + \mathbb{E}\, V_{k+1}(f(x_k, X_k, u_k), X_{k+1})\} \tag{109}$$

$$= \min_{u_k} \left\{ g(x_k, X_k, u_k) + \sum_{j \in \mathcal{S}} p_{ij} V_{k+1}(f(x_k, X_k, u_k), X_{k+1} = j) \right\} \tag{110}$$

where the argument of the $\min$ operator in the last line is completely a deterministic function of u_k. Moreover, the statistics of the random process are incorporated via transition probability p_{ij}. The boundary condition is given by

$$V_N(x_N, X_N) = g_N(x_N) \tag{111}$$

Finally, the optimal control is saved as

$$u_k^* = \gamma_k(x_k, X_k) = \arg\min_{u_k} \left\{ g(x_k, X_k, u_k) + \sum_{j \in \mathcal{S}} p_{ij} V_{k+1}(f(x_k, X_k, u_k), X_{k+1} = j) \right\} \tag{112}$$

As before, these equations are solved recursively going backwards in time. The critical difference here is that $V_{k+1}(\cdot,\cdot)$ must be recalled for all possible $X_{k+1} = j$ that might occur after X_k. Then

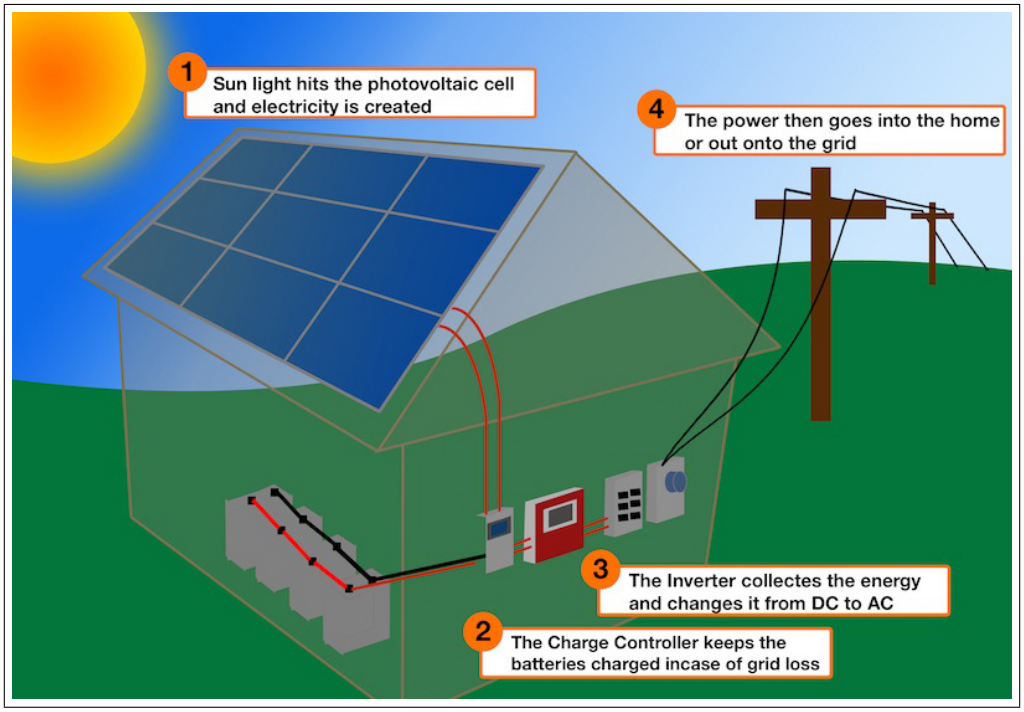

Figure 14: Schematic of photovoltaic generator with storage interfaced with the electricity grid.

we weight that cost-to-go with the corresponding transition probability p_{ij}, utilizing the definition of expectation [13].

Example 7.2 (Photovoltaic Generator with Storage)**.** We seek to develop an optimal energy management strategy for a photovoltaic (PV) generator with energy storage to maximize profit from selling energy to the grid. A schematic of this system is provided in Fig. 14. The key novelty here is that PV power is fundamentally intermittent, meaning we do not know it exactly a priori. We may, however, collect historical data of environmental conditions related to PV generation and develop a Markov chain model.

Suppose we have collected historical data on the local solar irradiation S and temperature T. Given this data, we propose Markov chain models for the stochastic evolution of each environmental variable

$$p_{ijk} = \Pr\left[S_{k+1} = S^j | S_k = S^i, k\right], \quad \forall\, S^i, S^j \in \mathcal{S}, \;\; k = 0, \cdots, N-1 \tag{113}$$

$$q_{lmk} = \Pr\left[T_{k+1} = T^m | T_k = T^l, k\right], \quad \forall\, T^l, T^m \in \mathcal{T}, \;\; k = 0, \cdots, N-1 \tag{114}$$

In words, the solar irradiation and temperature in the next time step are conditioned on the current solar irradiation and temperature, and current time period. Variables S^i, S^j are different levels of solar irradiation within a set $\mathcal{S}$. For example, the set $\mathcal{S} = \{0, 50, 100, \cdots, 1000\}$ W/m^2. Similarly, S^l, S^m are different levels of temperature within a set $\mathcal{T}$. For example, the set

$\mathcal{T} = \{0, 5, 10, \cdots, 40\}$ °C. Utilizing the photovoltaic cell model from [14], we find that PV power is a nonlinear function of S and T, namely

$$P_{pv,k} = f(S_k, T_k) \tag{115}$$

The conservation of power property for the entire system is given by the familiar power conservation equation,

$$P_{pv,k} + P_{batt,k} = P_{grid,k} \tag{116}$$

where $P_{batt,k}$ is the battery power (positive in discharge mode) and $P_{grid,k}$ is the power supplied to the grid. We also have the familiar battery energy storage dynamics

$$E_{k+1} = E_k - \Delta t \cdot P_{batt,k} \tag{117}$$

where E_k is the battery energy level. Finally, we have power and energy limits

$$-P_{batt}^{\max} \leq P_{batt,k} \leq P_{batt}^{\max}, \qquad k = 0, \cdots, N-1 \tag{118}$$

$$E^{\min} \leq E_k \leq E^{\max}, \qquad k = 0, \cdots, N-1 \tag{119}$$

$$E_N^{\min} \leq E_N \leq E_N^{\max} \tag{120}$$

We are now positioned to formulate the optimization program

$$\max_{P_{batt,k}, E_k, S_k, T_k} \quad J = \mathbb{E} \sum_{k=0}^{N-1} c_k \Delta t P_{grid,k} = \mathbb{E} \sum_{k=0}^{N-1} c_k \Delta t \left[f(S_k, T_k) + P_{batt,k} \right] \tag{121}$$

$$\text{s. to} \quad E_{k+1} = E_k - \Delta t \cdot P_{batt,k}, \qquad k = 0, \cdots, N-1, \tag{122}$$

$$p_{ijk} = \Pr\left[S_{k+1} = S^j | S_k = S^i, k \right], \qquad \forall\, S^i, S^j \in \mathcal{S}, \;\; k = 0, \cdots, N-1 \tag{123}$$

$$q_{lmk} = \Pr\left[T_{k+1} = T^m | T_k = T^l, k \right], \qquad \forall\, T^l, T^m \in \mathcal{T}, \;\; k = 0, \cdots, N-1 \tag{124}$$

$$-P_{batt}^{\max} \leq P_{batt,k} \leq P_{batt}^{\max}, \qquad k = 0, \cdots, N-1 \tag{125}$$

$$E^{\min} \leq E_k \leq E^{\max}, \qquad k = 0, \cdots, N-1 \tag{126}$$

$$E_N^{\min} \leq E_N \leq E_N^{\max} \tag{127}$$

Let the value function $V_k(E_k, S_k, T_k)$ be defined as the expected reward-to-go from time-step k to the end of the time horizon N, given that the current battery energy level, solar irradiation, and temperature are E_k, S_k, T_k, respectively. Then the principle of optimality is given by:

$$\begin{aligned} V_k(E_k, S_k, T_k) &= \max_{P_{batt,k} \in \mathcal{D}_k} \left\{ c_k \Delta t \left[f(S_k, T_k) + P_{batt,k} \right] + \mathbb{E}\, V_{k+1}(E_{k+1}, S_{k+1}, T_{k+1}) \right\} \\ &= \max_{P_{batt,k} \in \mathcal{D}_k} \left\{ c_k \Delta t \left[f(S_k, T_k) + P_{batt,k} \right] + \mathbb{E}\, V_{k+1}(E_k - \Delta t\; P_{batt,k}, S_{k+1}, T_{k+1}) \right\} \end{aligned}$$

$$= \max_{P_{batt,k} \in \mathcal{D}_k} \{c_k \Delta t \left[f(S_k, T_k) + P_{batt,k} \right] + \sum_{S^j \in \mathcal{S}} \sum_{T^m \in \mathcal{T}} p_{ijk}\, q_{lmk}\, V_{k+1}(E_k - \Delta t\, P_{batt,k}, S_{k+1} = S^j, T_{k+1} = T^m) \Bigg\} \quad (128)$$

where the maximization operator is subject to a time-varying admissible control set $\mathcal{D}_k$ characterized by

$$-P_{batt}^{\max} \leq P_{batt,k} \leq P_{batt}^{\max}, \qquad k = 0, \cdots, N-1 \quad (129)$$

$$E^{\min} \leq E_k \leq E^{\max}, \qquad k = 0, \cdots, N-1, \quad (130)$$

$$E_N^{\min} \leq E_N \leq E_N^{\max} \quad (131)$$

Plugging the battery dynamics (117) into (129), we arrive at two pairs of inequalities that bound $P_{batt,k}$ from above and below. At each time step, one of the upper/lower limits will dominate. It is easy to verify that at each step, $P_{batt,k}$ is bounded according to

$$\max \left\{ -P_{batt}^{\max}, \frac{1}{\Delta t}(E_k - E^{\max}) \right\} \leq P_{batt,k} \leq \min \left\{ P_{batt}^{\max}, \frac{1}{\Delta t}(E_k - E^{\min}) \right\}, \qquad k = 0, \cdots, N-1 \quad (132)$$

$$\max \left\{ -P_{batt}^{\max}, \frac{1}{\Delta t}(E_k - E_N^{\max}) \right\} \leq P_{batt,k} \leq \min \left\{ P_{batt}^{\max}, \frac{1}{\Delta t}(E_k - E_N^{\min}) \right\} \quad (133)$$

which characterizes the set $\mathcal{D}_k$. We also have the boundary condition

$$V_N(E_N, S_N, T_N) = 0, \qquad \forall\, E_N, S_N, T_N \quad (134)$$

Finally, the optimal control action is saved as

$$P_{batt,k}^* = \gamma_k \left(E_k, S_k, T_k \right) = \arg \max_{P_{batt,k} \in \mathcal{D}_k} \{c_k \Delta t \left[f(S_k, T_k) + P_{batt,k} \right] + \sum_{S^j \in \mathcal{S}} \sum_{T^m \in \mathcal{T}} p_{ijk}\, q_{lmk}\, V_{k+1}(E_k - \Delta t\, P_{batt,k}, S_{k+1} = S^j, T_{k+1} = T^m) \Bigg\} \quad (135)$$

To test this algorithm, we adopt measured solar irradiance data collected from the Indiana State Climate Office from May 2014, shown in Fig. 15. For exposition purposes, we will disregard temperature. We also adopt wholesale electricity price data adopted from CAISO, also shown in Fig. 15. Figure 16 provides the probability mass function and cumulative mass functions for the Indiana solar irradiance data. Not surprisingly, 50% of the time the solar irradiance is zero. All other irradiance levels are roughly evenly distributed.

We compute the transition probabilities in (113) by simply counting the number of transitions

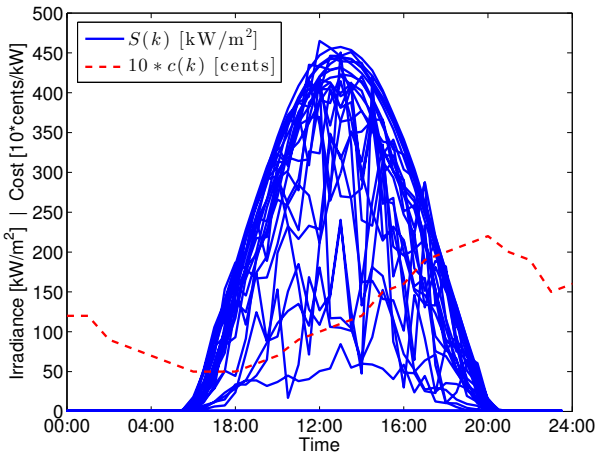

Figure 15: [BLUE] Raw solar irradiance data, adopted from Indiana State Climate Office from May 2014. [RED] Raw wholesale electricity price data adopted from CAISO.

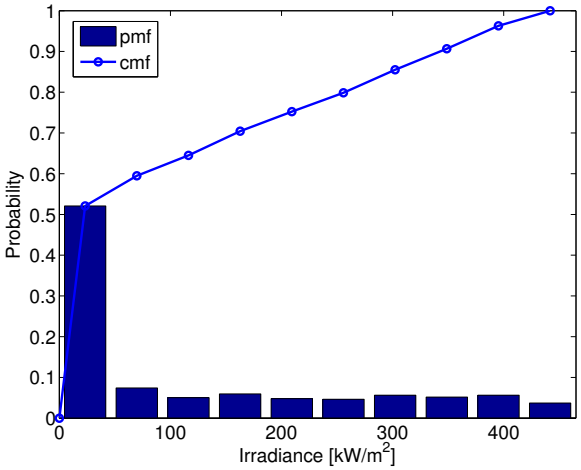

Figure 16: Probability and cumulative mass functions for the solar irradiance level, computed from raw data.

from one irradiance level to another, and dividing by the total number of transitions. This is done for each 30min segment of the day. The resulting transition probabilities for the Markov chain are summarized visually in Fig. 17. One clearly sees the irradiance is at the lowest level (zero) at midnight and 4am. It transitions to the lowest level in the next 30min with probability one, and all other levels with probability zero. This is intuitive. We can also see characteristics of sunrise and sunset for 8am and 4pm.

After implementing the SDP algorithm described above, we arrive at an optimal control policy $P^*_{batt,k} = \gamma_k \left(E_k, S_k \right)$. This policy is tested on the 24 hour period shown in Fig. 18. We use the following parameters: $E^{\min}$ = 0 kWh, $E^{\max}$ = 10 kWh, $E_N^{\min}$ = 4.5 kWh, $E_N^{\max}$ = 5.5 kWh, E_0 = 5 kWh, $P_{batt}^{\min}$ = -10 kW, $P_{batt}^{\max}$ = +10 kW, Δt = 30 min.

The results are quite interesting, and clearly maximize profit. However, the exact solution is not simple to guess - thus motivating model/data-based optimization. Around 1am, the battery discharges and sells electricity to the grid to leverage relatively high price. Around 3am-4am, the system purchases electricity from the grid at relatively low price to charge the battery - thus performing arbitrage. As the sun rises and prices increase, the system sells electricity to the grid and depletes the battery. Around 12noon, there are relatively low prices, so the system purchases electricity and uses solar to charge the battery. Between 2pm-4pm, the prices increases slightly, the system sells all available energy from solar and storage. Interestingly, after 4pm the system uses the generated solar and purchased grid electricity to fully charge the battery. The reason to do so is the following. In the evening, 7pm-10pm, the prices are high. The system exploits this by selling stored battery energy during this period, even though the sun has set. Just before midnight, the prices are relatively low so the system returns the battery energy level to near 5 kWh.

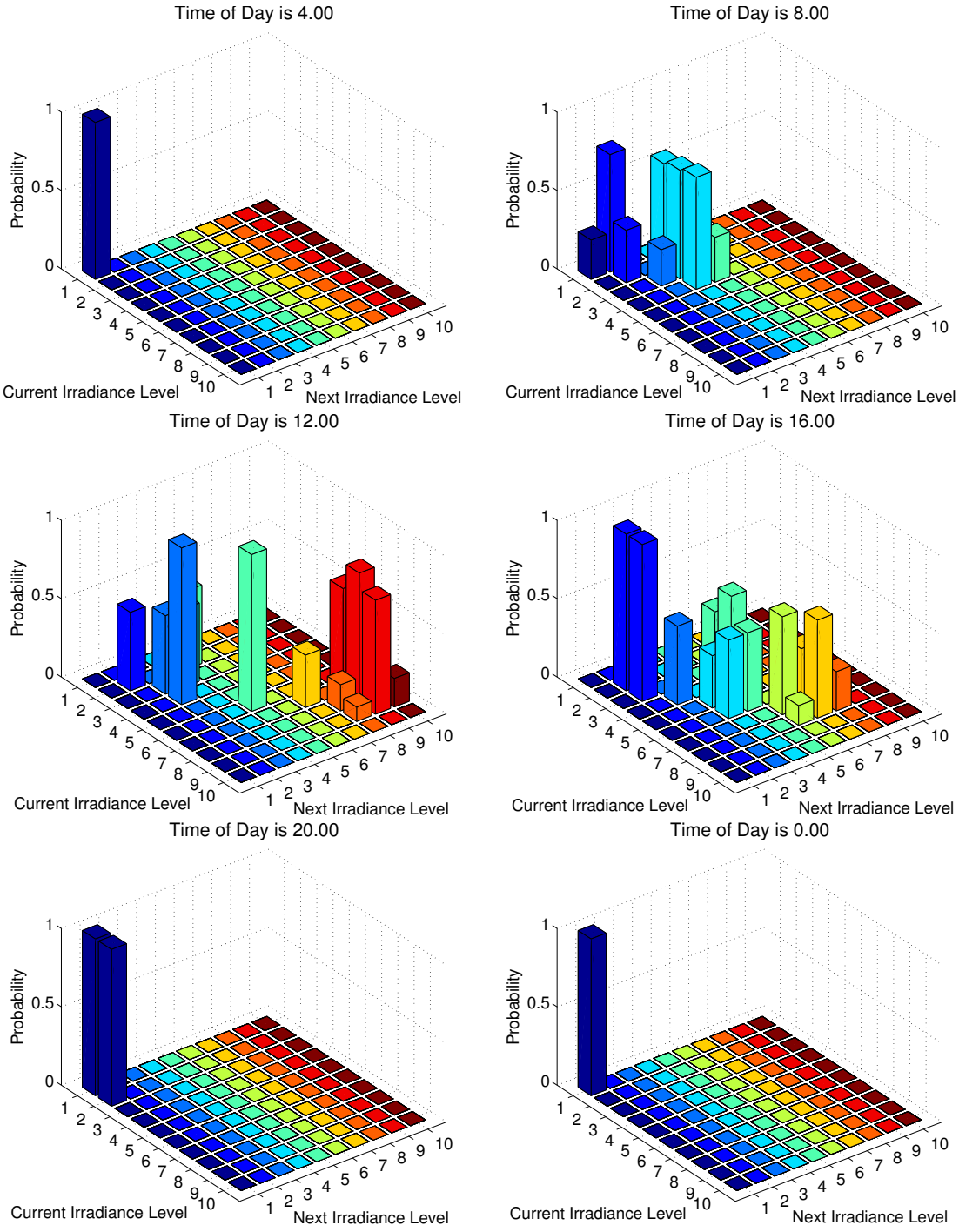

Figure 17: Transition probabilities identified from raw solar data in Fig. 15.

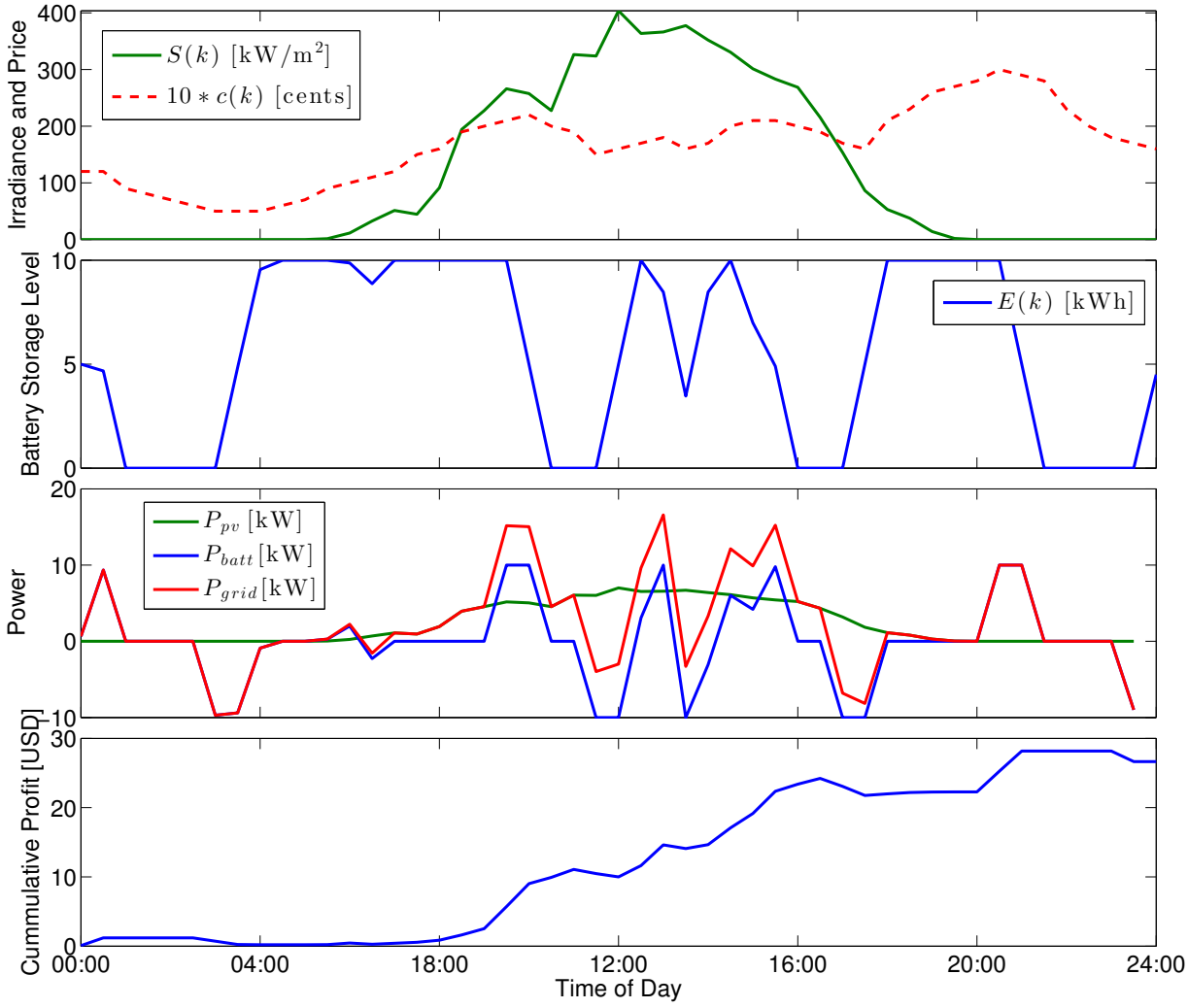

Figure 18: Sample Simulation with SDP-Optimized Control Law.

8 Model Predictive Control

9 Notes

You can learn more about optimal control in energy systems...

References

[1] S. Moura, H. Fathy, D. Callaway, and J. Stein, "A Stochastic Optimal Control Approach for Power Management in Plug-In Hybrid Electric Vehicles," IEEE Transactions on Control Systems Technology, vol. 19, no. 3, pp. 545 –555, 2011.

[2] S. Moura, J. Stein, and H. Fathy, "Battery-Health Conscious Power Management in Plug-In Hybrid Electric Vehicles via Electrochemical Modeling and Stochastic Control," IEEE Transactions on Control Systems Technology, vol. 21, no. 3, pp. 679–694, 2013.

[3] B. Egardt, N. Murgovski, M. Pourabdollah, and L. Johannesson Mardh, "Electromobility studies based on convex optimization: Design and control issues regarding vehicle electrification," Control Systems Magazine, vol. 34, no. 2, pp. 32–49, April 2014.

[4] C. Sun, F. Sun, and S. J. Moura, "Data enabled predictive energy management of a pv-battery smart home nanogrid," IEEE Transactions on Smart Grid, vol. PP, pp. 1–12, 2015.

[5] C. Marnay, G. Venkataramanan, M. Stadler, A. Siddiqui, R. Firestone, and B. Chandran, "Optimal technology selection and operation of commercial-building microgrids," IEEE Transactions on Power Systems, vol. 23, no. 3, pp. 975–982, Aug 2008.

[6] B. Washom, J. Dilliot, D. Weil, J. Kleissl, N. Balac, W. Torre, and C. Richter, "Ivory tower of power: Microgrid implementation at the university of california, san diego," IEEE Power and Energy Magazine, vol. 11, no. 4, pp. 28–32, July 2013.

[7] T. Ersal, C. Ahn, I. Hiskens, H. Peng, and J. Stein, "Impact of controlled plug-in evs on microgrids: A military microgrid example," in Power and Energy Society General Meeting, 2011 IEEE, July 2011, pp. 1–7.

[8] A. Malikopoulos, "Supervisory power management control algorithms for hybrid electric vehicles: A survey," IEEE Transactions on Intelligent Transportation Systems, vol. PP, no. 99, pp. 1–17, 2014.

[9] L. Serrao, G. Rizzoni, and S. Onori, "A comparative analysis of energy management strategies for hybrid electric vehicles," Journal of Dynamic Systems, Measurement, and Control, vol. 133, no. 3, p. 031012, 2011.

[10] J. Gonder and T. Markel, "Energy management strategies for plug-in hybrid electric vehicles," Energy, vol. 1, p. 0290, 2007.

[11] A. Sciarretta and L. Guzzella, "Control of hybrid electric vehicles," Control Systems Magazine, vol. 27, no. 2, pp. 60–70, April 2007.

[12] H. Hotelling, "Stability in competition," The Economic Journal, vol. 39, no. 153, pp. pp. 41–57, 1929.

[13] J. A. Gubner, Probability and random processes for electrical and computer engineers. Cambridge University Press, 2006.

[14] G. M. Masters, Renewable and efficient electric power systems. John Wiley & Sons, 2013.