CHAPTER 2: MACHINE LEARNING

1 Overview

What is machine learning (ML)? From our perspective, it is quite simply *mathematical modeling from data*. Recall Chapter 1 on mathematical modeling. Recall that we outlined seven steps as a guide for creating mathematical models. Step 6 involves training the unknown system parameter from experimental data. This chapter dives deeply into this specific task. Note, the mathematical model itself may arise from either first-principles, or be a generic "black-box" format. In either case, our objective is to "fit" the unknown parameters (a.k.a. "weights" or "coefficients") from experimental data. In energy systems, we sometimes find ourselves rich in data yet poor in engineering insight. These techniques provide the engineer with various tools to generate system models from data.

Machine learning is, in fact, a convergence of several other intellectual fields that include optimization, statistics, control theory, and computer science. We have strategically introduced optimization before ML in this course, because ML can be viewed as an application of optimization. Historically, "machine learning" was coined by IBM computer scientist Arthur Samuel in 1959. Specifically, Arthur Samuel said the following about ML: "It gives computers the ability to learn without being explicitly programmed." In the 1980s – 1990s, the concept of artificial intelligence (AI) emerged as another re-branding of ML. Interestingly, AI and ML lost popularity throughout the 2000s. The main reasons include: (i) The promises of AI and ML far exceeded the results, and therefore academia and industry disinvested in these ideas. (ii) The lack of consistent results is partially attributed to theoretical misunderstandings, many of which have since been resolved. (iii) We now have the computational resources to handle very large datasets and parallelized computation. Consequently, ML and AI have re-emerged in the 2010's as "Data Science". Now academia and industry are re-investing heavily in these technologies. In this chapter, you will see that ML is essentially an application of optimization, statistics, and control theory. With this view, you will become ML experts who understand the fundamental principles.

To this end, Chapter 2 examines the following questions:

- How do we formulate a parametric model?
- How do we identify a parametric model from data?
- Why do these algorithms work? Why do they sometimes not work?

1.1 White-box vs. Black-box Models

Inferring models from observations and studying their properties is the essential basis of science. These models (a.k.a. hypotheses, laws of nature, theories) can be more or less formal in nature.

However, they all attempt to link observations together through some pattern. Machine learning involves the inference of a system model from input-output observations, as shown in Fig. 1. The construction of this system often falls within three different paradigms: white-box models, black-box models, or grey-box models.

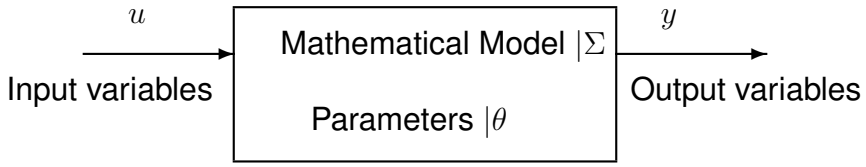

Figure 1: Machine Learning involves inferring Σ and/or θ, from measurements of u, y.

- **[White-box Models]:** Sometimes we can formulate the mathematical model equations from first principles (e.g. Newton's laws, Navier-Stokes, Maxwell's equations, etc.). This white-box model paradigm is often preferable, since it simplifies the model identification process and endows the model with physically relevant meaning. Moreover, these models are capable of accurately predicting behavior outside of the training data.
- **[Black-box Models]:** In some applications, the underlying dynamic process is too complex to model from first principles. Examples include human behavior, biological systems, social systems, or large-scale systems with many intertwined dynamic phenomena. The black-box modeling paradigm starts from measurements of the system's inputs and outputs and determines a mathematical relation between them without attempting to model the internals of the system. Note that these models are generally unreliable extrapolating behavior outside of the training data range.
- **[Grey-box Models]:** The aforementioned categories are end-points on a spectrum. In many practical applications, we seek a combination of white-box and black-box models. This is the most interesting and practical category. Grey-box models have some components that we can model from first principles, and other components that must be fit empirically with data.

In this chapter, we are interested in tools applicable to white-box models, black-box models, and all the shades of grey in between. Interested readers may refer to the textbook by Ljung [1] for a thorough exposition of system identification methods for black-box models.

1.2 Online vs. Offline Algorithms

Throughout this chapter, we introduce offline and online algorithms for machine learning. Offline (a.k.a. batch or non-recursive) algorithms operate on a fixed collection of data, asynchronous from

the true physical system. These algorithms are often simpler to formulate. However, they require large collections of data and computational power, and are not amenable to implementation in an embedded computational system. Sometimes one wishes to "learn" the model in a continuous manner, where it utilizes data as it arrives and then discards it. These techniques are known as online (a.k.a. recursive) algorithms. These algorithms are typically more difficult to formulate, but are amenable to implementation in embedded systems in synchrony with the dynamic system.

1.3 Chapter Organization

The remainder of this chapter is organized as follows:

1. Parametric Modeling
2. Gradient Algorithm
3. Least-Squares (LSQ) Algorithm
4. Nonlinear Least-Squares (NL-LSQ) Algorithm
5. Offline Techniques
6. Sensitivity Analysis

2 Regression

We begin our exposition of regression with the following example.

Example 2.1 (Linear Regression)**.** Suppose you have collected measured data pairs (x_i, y_i), for $i = 1, \cdots, N$ where $N > 6$, as shown in Fig. 2. You seek to fit a fifth-order polynomial to this data, i.e.

$$y = c_0 + c_1 x + c_2 x^2 + c_3 x^3 + c_4 x^4 + c_5 x^5 \tag{1}$$

A crucial point we emphasize is that (1) is linear-in-the-parameters c_i. Indeed, the output y is nonlinear with respect to x. However, our focus is on fitting values for the c_i's, and y is linear with respect to each and every c_i.

The goal is to determine parameters $c_i,\ i = 0, \cdots, 5$ that "best" fit the data in some sense. To this end, you may compute the residual r for each data pair:

$$\begin{aligned} c_0 + c_1 x_1 + c_2 x_1^2 + c_3 x_1^3 + c_4 x_1^4 + c_5 x_1^5 - y_1 &= r_1, \\ c_0 + c_1 x_2 + c_2 x_2^2 + c_3 x_2^3 + c_4 x_2^4 + c_5 x_2^5 - y_2 &= r_2, \\ \vdots &= \vdots \\ c_0 + c_1 x_N + c_2 x_N^2 + c_3 x_N^3 + c_4 x_N^4 + c_5 x_N^5 - y_N &= r_N, \end{aligned} \tag{2}$$

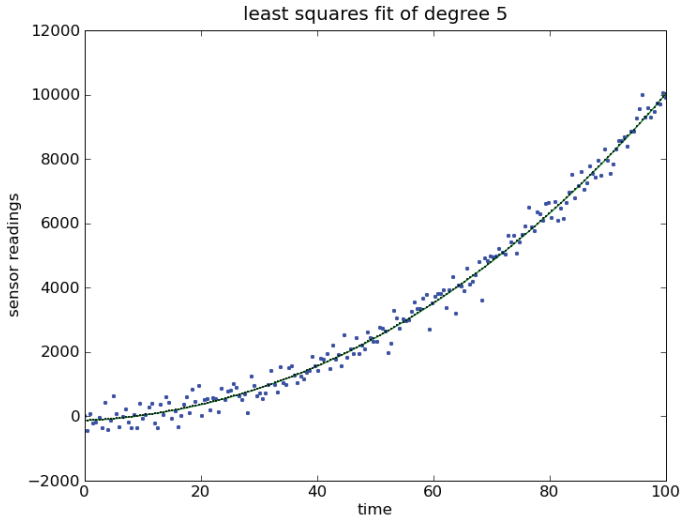

Figure 2: You seek to fit a fifth-order polynomial to the measured data above.

which can be arranged into matrix-vector form $Ac - y = r$, where

$$A = \begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 & x_1^4 & x_1^5 \\ 1 & x_2 & x_2^2 & x_2^3 & x_2^4 & x_2^5 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_N & x_N^2 & x_N^3 & x_N^4 & x_N^5 \end{bmatrix}, \qquad c = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{bmatrix}, \qquad y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}, \qquad r = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_N \end{bmatrix}. \quad (3)$$

Now we compute an optimal fit for c in the following sense. We seek the value of c which minimizes the squared residual

$$\min_c \quad \frac{1}{2}\|r\|_2^2 = \frac{1}{2}r^T r = \frac{1}{2}(Ac-y)^T(Ac-y) = \frac{1}{2}c^T A^T Ac - y^T Ac + \frac{1}{2}y^T y. \quad (4)$$

Note that (4) is quadratic in variable c and therefore a convex function of c. This produces a **quadratic program**. In this case the problem is unconstrained. That is, our immediate objective is to solve an unconstrained QP to fit the parameters. To this end, we set the gradient with respect to c to zero and directly solve for the minimizer.

$$\begin{aligned} \frac{\partial}{\partial c}\frac{1}{2}\|r\|_2^2 = A^T Ac - A^T y = 0, \\ A^T Ac = A^T y, \\ \boxed{c = (A^T A)^{-1} A^T y} \end{aligned} \quad (5)$$

This provides a direct formula for fitting the polynomial coefficients $c_i,\ i = 0, \cdots, 5$ using the measured data.

Exercise 1. *Consider fitting the coefficients c_1, c_2, c_3 of the following sum of radial basis functions to data pairs (x_i, y_i), $i = 1, \cdots, N$.*

$$y = c_1 e^{-(x-0.25)^2} + c_2 e^{-(x-0.5)^2} + c_3 e^{-(x-0.75)^2} \tag{6}$$

Formulate the corresponding QP problem. Derive the explicit formula for fitting the coefficients.

Exercise 2. *Repeat the same exercise for the following Fourier Series:*

$$y = c_1 \sin(\omega x) + c_2 \cos(\omega x) + c_3 \sin(2\omega x) + c_4 \cos(2\omega x) \tag{7}$$

Exercise 3 (Tikhonov or L_2 regularization, a.k.a. Ridge Regression)**.** *Consider the fifth-order polynomial regression model in (1). Suppose we seek the value of c which minimizes the squared residual plus a so-called Tikhonov regularization term:*

$$\min_c \quad \frac{1}{2}\|r\|_2^2 + \frac{1}{2}\|\Gamma c\|_2^2. \tag{8}$$

for some matrix $\Gamma \in \mathbb{R}^{6\times 6}$. Derive the QP. Solve for the minimizer of this unconstrained QP. Provide a formula for the optimal coefficients c.

3 Maximum Likelihood Estimation

4 Parametric Modeling

Algorithms and optimization receive all the glory in model identification. However, skillful parametric modeling gets grossly under-appreciated. It is a necessary initial step for any model identification study. A cleverly formulated parametric model can render the algorithms nearly trivial. This section discusses various canonical formulations for parametric models.

In Chapter 1, we discussed the derivation of models from first principles. This typically renders ODEs that can be formulated into state space form as a series of first-order ODEs

$$\dot{x}(t) = f(x(t), u(t); \theta) \tag{9}$$

where $x(t) \in \mathbb{R}^{n\times 1}$, $u(t) \in \mathbb{R}^{p\times 1}$, t represents continuous time, and $\theta \in \mathbb{R}^{p\times 1}$ are a set of constant parameters to be identified. In practice, data is sampled in discrete-time instances. Consequently we shall also consider discrete-time models in this chapter, which take the form

$$x(k+1) = f(x(k), u(k); \theta) \tag{10}$$

In this so-called "difference equation model", k is related to t according to $t = k\Delta t$, where Δt is the sampling time.

4.1 Linear Parametric Models

Consider the discrete-time analog of an $n-$th order differential equation given by

$$y(k+1) + a_0 y(k) + a_1 y(k-1) + \ldots + a_n y(k-n) = b_0 u(k) + b_1 u(k-1) + \ldots + b_n u(k-n) \quad (11)$$

where a_i, b_i, for $i = 0, 1, \ldots, n$ are scalar parameters. In the stochastic process literature [1], this is known as the famous ARX model (auto-regressive with exogenous input $u()$), and is a common black-box model. If we lump all the a_i, b_i parameters into the vector

$$\theta = [b_0, b_1, \ldots, b_n, a_0, a_1, \ldots, a_n]^T \quad (12)$$

and all the input/output (I/O) signals into the vector

$$\phi(k) = [u(k), u(k-1), \ldots, u(k-n), -y(k), -y(k-1), \ldots, -y(k-n)]^T \quad (13)$$

then we may re-write (11) into matrix-vector form as

$$y(k+1) = \theta^T \phi(k). \quad (14)$$

Equation (14) is linear in θ which, as we show in subsequent sections, is crucial for designing parameter identifiers to estimate θ from the measurements of $y(k+1)$ and $\phi(k)$.

4.2 Nonlinear Parametric Models

It is always beneficial to formulate the model into the form (14) when possible. However, sometimes the model is not linear in the parameters. In these cases, the set of available algorithms decreases dramatically and their properties generally weaken. However, these nonlinear cases do arise in practice. Generally, a nonlinear in the parameters model takes the form of

$$y(k+1) = f(\phi(k); \theta) \quad (15)$$

where $\phi(k)$ is an appropriately defined vector of I/O data, $y(k+1)$ is the output data at time-step $k+1$, θ is the unknown parameter vector, and $f(\cdot, \cdot)$ is a nonlinear function that maps I/O data and parameters to a scalar measured output. Artificial neural networks are an example of a nonlinear-in-the-parameters model. In this case, we must utilize iterative optimization methods to estimate the value of parameter vector θ that best matches the data and model.

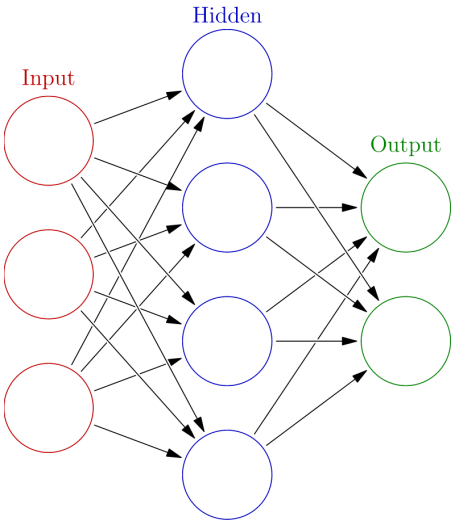

Figure 3: An artificial neural network (ANN) is a weighted and directed graph, where the nodes are called "neurons".

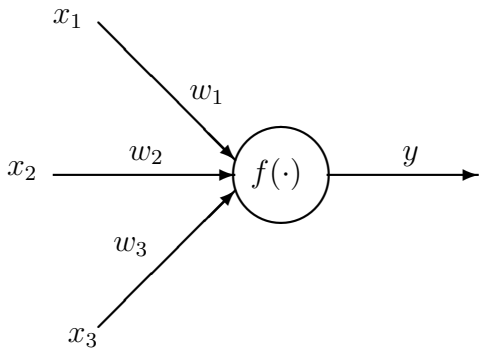

Figure 4: Simple single neuron network with activation function $f(\cdot)$.

Example 4.1 (Artificial Neural Networks (ANN))**.** Artificial neural networks (ANNs) were first created by Warren McCulloch and Walter Pitts in 1943 as a computational model for the human brain. Recently, ANNs have re-emerged within artificial intelligence due to (i) the availability of enhanced computational power, and (ii) resolutions to theoretical stability issues.

A neural network is composed of a weighted directed graph, as shown in Fig. 3. Each node of the graph (a.k.a. neuron) is "activated" by inputs. The node will perform some simple computations on these inputs, using a so-called "activation function", and then pass its result along the network. More specifically, each node computes a weighted sum of its inputs and then processes this sum with its activation function, i.e. $f\left(\sum_i w_i x_i\right)$ or written in vector notation as $f(w^T x)$.

Consider the trivially simple single neuron network shown in Fig. 4. Denote the input weights by w_i. There are several canonical activation functions.

Linear-in-the-parameters Model: Consider the *affine* activation function

$$f(w^T x) = a \cdot w^T x + b, \tag{16}$$

where $a, b \in \mathbb{R}$. This renders the parametric input-output model: $y = a \cdot w^T x + b$. Now, suppose we define the parameter vector to be:

$$\theta = [a, b, w_1, w_2, \cdots, w_n]^T. \tag{17}$$

where $\theta \in \mathbb{R}^{n+2}$. This is intuitive. We just collect each scalar parameter into a vector θ. However, this yields a parametric input-output model which is nonlinear in θ. To see this, let θ_i be the ith

component of θ. Then we have

$$y = \theta_1 \cdot \sum_{j=1}^{n-1} \theta_{j+2} x_j + \theta_2 \tag{18}$$

where $\theta_1 \cdot \theta_{j+2}$ is a nonlinear term.

An alternative choice (and better w.r.t. learning convergence) would be to define the parameter vector to be:

$$\vartheta = [a \cdot w_1, a \cdot w_2, \cdots, a \cdot w_n, b]^T. \tag{19}$$

where $\vartheta \in \mathbb{R}^{n+1}$. Let ϑ_i be the ith component of ϑ. Then we have

$$y = \sum_{i=1}^{n} \vartheta_i x_i + \vartheta_{n+1} \tag{20}$$

This parametric model is linear w.r.t. ϑ. Moreover, it has less parameters $n+1$ versus $n+2$ for θ.

Nonlinear-in-the-parameters Model: Two other common examples are the *sigmoid* and *tanh* activation functions, given respectively by

$$f_1(w^T x) = \frac{1}{1+\exp(w^T x)}, \qquad f_2(w^T x) = \tanh(w^T x). \tag{21}$$

These two activation functions produce parametric input-output models given by:

$$y = \frac{1}{1+\exp(w^T x)}, \qquad y = \tanh(w^T x), \tag{22}$$

which are both nonlinear in parameter vector w.

Example 4.2 (Flywheel Energy Storage)**.** Consider the flywheel energy storage example from CH1. Recall that the equation of motion is governed by Euler's rotation equation:

$$I\dot{\omega}(t) = -b\omega(t) + T(t), \tag{23}$$

where $\dot{(\cdot)}$ denotes the derivative with respect to time. Note that $\omega(t)$ is the state variable, $T(t)$ is a controllable input torque, and I, b are physical parameters representing the rotational inertia and friction coefficient, respectively. Suppose we can measure $\omega(t), T(t)$. Our ultimate goal is to learn the unknown parameters. To this end, we must first derive a parametric model. In this example, we will derive both linear and nonlinear-in-the-parameters models.

Suppose we approximate the time-derivative using forward Euler's method: $\frac{d}{dt}\omega(t) \approx \frac{\omega(t+\Delta t)-\omega(t)}{\Delta t}$. Furthermore, we define discrete-time index k such that $t = k \cdot \Delta t$. Then we can re-write (23) as

discrete time difference equation

$$I \cdot \frac{\omega(k+1)-\omega(k)}{\Delta t} = -b\omega(k) + T(k), \tag{24}$$

$$\omega(k+1) = \left[1 - \frac{b\Delta t}{I}\right]\omega(k) + \frac{\Delta t}{I}T(k) \tag{25}$$

Linear-in-the-parameters Model: The flywheel dynamics are now in the form of $\omega(k+1) = \theta^T\phi$ in (14), where the parameter vector θ and regressor signal ϕ are:

$$\theta = \left[1 - \frac{b\Delta t}{I},\ \frac{\Delta t}{I}\right]^T, \qquad \phi = [\omega(k),\ T(k)]^T. \tag{26}$$

Interesting, if the sampling time Δt is known (a very mild assumption), then we can back-calculate I, b from $\theta_1 = 1 - \frac{b\Delta t}{I}$, $\theta_2 = \frac{\Delta t}{I}$ by the following formulae: $I = \frac{\Delta t}{\theta_2}$, $b = \frac{1-\theta_1}{\theta_2}$. Note, it is NOT always possible to back-calculate the physical parameters from the linear parameter vector θ. A necessary (but not sufficient) condition for back-calculating physical parameters from θ is that the number of physical parameters equals the dimension of θ.
Nonlinear-in-the-parameters Model: Alternatively, we might initially define the parameter vector to correspond to the physical parameters as follows: $\theta = [I, b]^T$. In this case, we have a nonlinear-in-the-parameters model given by:

$$\omega(k+1) = f(\omega(k), T(k); \theta) \tag{27}$$

Loosely speaking, it is MUCH more difficult to identify parameters for the nonlinear-in-the-parameters model above. Technically speaking, there are no general algorithms that are provably convergent. Hence, linear-in-the-parameters models are always preferred.

4.3 Identifiability

A logical question to ask is: "*Is it always possible to determine the model parameters by observing a sufficient amount of data? If not, then is there a test to determine when it is possible to determine the model parameters?*" In model identification this *existence* question is called *identifiability*. We now present several notions of identifiability:

- **[Identifiable]:** A model is said to be identifiable if it is possible to uniquely learn the parameters by observing a sufficient amount of data.
- **[Partially Identifiable]:** A model is said to be partially identifiable if it is possible to uniquely learn a sub-set of the parameters by observing a sufficient amount of data.

Note that identifiability and partial identifiability *depend on your data*. When people say "I have excellent data quality," they are referring (perhaps ignorantly) to this concept. Given these con-

cepts, a simple test to determine identifiability would be extremely convenient. We shall discuss two tests in this chapter. The first test given in Section 5.3 is specific to models that are linear-in-the-parameters, and is known as the *persistence of excitation* condition. The second test given in Section 9 is applicable to nonlinear-in-the-parameters models, and is called *sensitivity analysis*.

5 Gradient Algorithm

5.1 Scalar Example

Consider the scalar input-output system given by the algebraic equation

$$y(t) = \theta u(t) \tag{28}$$

where $y(t)$ is a scalar output, $u(t)$ is a scalar input, and θ is an unknown scalar parameter. Assuming that $u(t), y(t)$ are measured, we seek an estimate of θ at each time t. We denote the estimated version of the parameter vector as $\hat{\theta}$. If the measurements of y, u were noise free, one could simply calculate $\hat{\theta}(t)$ as

$$\hat{\theta} = \frac{y(t)}{u(t)} \tag{29}$$

whenever $u(t) \neq 0$. The division (29), however, may not be desirable because $u(t)$ may assume values arbitrarily close to zero. Furthermore, the effect of noise on the measurement of u, y may lead to an erroneous estimate of θ. The noise and divide-by-zero effects in (29) may be reduced by using various other nonrecursive or offline methods, especially when θ is a constant for all t.

In our case, we are interested in a recursive or on-line method to generate $\hat{\theta}(t)$. We are looking for a differential equation, which (i) depends on signals that are measured, (ii) whose solution is $\hat{\theta}(t)$, (iii) has equilibrium point $\hat{\theta}^{eq} = \theta$, and (iv) this equilibrium point is asymptotically stable. The procedure for developing such a differential equation is given below.

Using $\hat{\theta}(t)$ as the estimate of θ at time t, we generate the estimated or predicted value $\hat{y}(t)$ of the output $y(t)$ as

$$\hat{y}(t) = \hat{\theta}(t)u(t) \tag{30}$$

The prediction or estimation error $\epsilon(t)$ is given by

$$\epsilon(t) = y(t) - \hat{y}(t) = y(t) - \hat{\theta}(t)u(t) \tag{31}$$

Now define the parameter estimation error as $\tilde{\theta}(t) = \theta - \hat{\theta}(t)$, where the error $\epsilon(t)$ can be re-written as

$$\epsilon(t) = \theta u(t) - \hat{y}(t) = \tilde{\theta}(t)u(t) \tag{32}$$

The differential equation for generating $\hat{\theta}(t)$ is developed by minimizing various cost criteria of ϵ

with respect to $\hat{\theta}$. For this example, we use the simple cost criterion

$$J(\hat{\theta}) = \frac{1}{2}\epsilon^2 = \frac{1}{2}\left(y - \hat{\theta}u\right)^2 \tag{33}$$

It is easy to see that, for each time t, the cost function $J(\hat{\theta})$ is convex over all $\mathbb{R}$; therefore, any local minimum of J is also global and satisfies $\frac{d}{d\hat{\theta}}J(\hat{\theta}) = 0$. Consequently, we may utilize a gradient-scheme (a.k.a. steepest descent) to generate an update law for $\hat{\theta}(t)$, as follows

$$\dot{\hat{\theta}}(t) = -\gamma\frac{d}{d\hat{\theta}}J(\hat{\theta}) = \gamma(y - \hat{\theta}u)u = \gamma\epsilon u, \qquad \hat{\theta}(0) = \hat{\theta}_0 \tag{34}$$

where $\gamma > 0$ is a scaling constant. In the literature, the differential equation (34) is referred to as the *update law* or *identification algorithm* for updating $\hat{\theta}(t)$ to estimate θ online.

Stability analysis helps us show that the solution of (34) indeed converges to the true value of θ, under appropriate conditions. Stability is analyzed by re-writing (34) in terms of the parameter error $\tilde{\theta} = \theta - \hat{\theta}$, i.e.

$$\dot{\tilde{\theta}} = \dot{\theta} - \dot{\hat{\theta}} = \dot{\theta} - \gamma\epsilon u \tag{35}$$

Because θ is constant, i.e. $\dot{\theta} = 0$, and $\epsilon = \tilde{\theta}u$ from (32), we have

$$\dot{\tilde{\theta}}(t) = -\gamma u^2(t)\tilde{\theta}(t) \tag{36}$$

Note that the coefficient $-\gamma u^2(t)$ is always non-positive, which from Chapter 1 we learned proves that the parameter error dynamics are marginally stable or stable in the sense of Lyapunov. If we add the sufficient condition that $u(t) \neq 0$ uniformly in t, then we obtain the stronger condition that the estimation error is asymptotically stable. In other words, $\hat{\theta}(t) \to \theta$ as $t \to \infty$.

5.2 Vector Case

Next we present the vector case of the gradient algorithm. Consider a linear parametric model of the form

$$z(t) = \theta^T\phi(t) \tag{37}$$

where $z(t)$ is a scalar measurement, θ is a vector of unknown parameters, and $\phi(t)$ is a vector of I/O data. Note this is the canonical linear in the parameters model from Section 4.1. The corresponding prediction or estimation error $\epsilon(t)$ is given by

$$\epsilon(t) = z(t) - \hat{z}(t) = z(t) - \hat{\theta}^T(t)\phi(t) \tag{38}$$

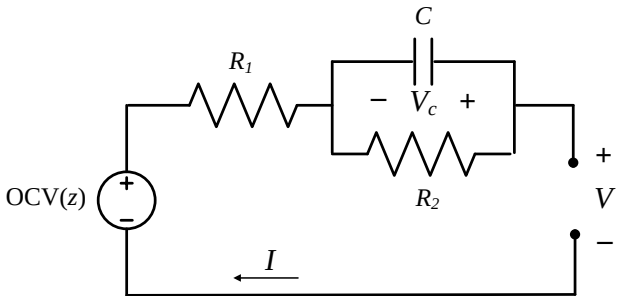

Figure 5: OCV-R-RC Equivalent circuit model of a battery. Includes an open circuit voltage in series with a resistance, in series with a resistor-capacitor pair.

Now define the parameter estimation error as $\tilde{\theta}(t) = \theta - \hat{\theta}(t)$, where the error $\epsilon(t)$ can be re-written as

$$\epsilon(t) = \theta^T \phi(t) - \hat{z}(t) = \tilde{\theta}^T(t)\phi(t) \tag{39}$$

The differential equation for generating $\hat{\theta}(t)$ is developed by minimizing various cost criteria of ϵ with respect to $\hat{\theta}$. For this example, we use the simple cost criterion

$$J(\hat{\theta}) = \frac{1}{2}\epsilon^2 = \frac{1}{2}\left(z - \hat{\theta}^T\phi\right)^2 \tag{40}$$

It is easy to see that, for each time t, the cost function $J(\hat{\theta})$ is convex over all $\mathbb{R}$; therefore, any local minimum of J is also global and satisfies $\frac{d}{d\hat{\theta}}J(\hat{\theta}) = 0$. Consequently, we may utilize a gradient-scheme (aka steepest descent) to generate an update law for $\hat{\theta}(t)$, as follows

$$\dot{\hat{\theta}}(t) = -\Gamma\frac{d}{d\hat{\theta}}J(\hat{\theta}) = -\Gamma\frac{dJ}{d\epsilon}\frac{d\epsilon}{d\hat{\theta}} = \Gamma\phi(t)\epsilon(t), \qquad \hat{\theta}(0) = \hat{\theta}_0 \tag{41}$$

where $\Gamma = \text{diag}(\gamma_1, \gamma_2, \cdots, \gamma_n) \succ 0$ is a diagonal, positive-definite matrix.

Example 5.1 (Battery Voltage)**.** In HW1, you studied an equivalent circuit model shown in Fig. 5, where the battery voltage output function is,

$$V(t) = OCV(z(t)) + V_c(t) + R_1 I(t) \tag{42}$$

The state variables are state-of-charge $z(t)$ and capacitor voltage $V_c(t)$. The input is current $I(t)$. In this equation, R is a parameter and function $OCV(z)$ can be parameterized with a cubic polynomial: $OCV(z) = p_0 + p_1 z + p_2 z^2 + p_3 z^3$. Substituting,

$$V(t) = p_0 + p_1 z(t) + p_2 z^2(t) + p_3 z^3(t) + V_c(t) + R_1 I(t) \tag{43}$$

Our goal is to use the gradient algorithm to recursively learn parameters p_0, p_1, p_2, p_3, R_1 in the model (43), given streaming data $V(t), z(t), V_c(t), I(t)$.

First, we must define a parametric model. A linear-in-the-parameters model $z(t) = \theta^T \phi(t)$ can be defined as follows:

$$z(t) = V(t) - V_c(t), \qquad \theta = [p_0, p_1, p_2, p_3, R_1]^T, \qquad \phi(t) = \left[1, z(t), z^2(t), z^3(t), I(t)\right]^T \tag{44}$$

Second, we apply the gradient algorithm given by

$$\dot{\hat{\theta}}(t) = \Gamma \phi(t) \epsilon(t), \qquad \hat{\theta}(0) = \hat{\theta}_0 \tag{45}$$

where $\Gamma = \text{diag}(\gamma_1, \gamma_2, \gamma_3, \gamma_4, \gamma_5) \succ 0$ is a diagonal, positive-definite matrix. Think of Γ as user-selected parameter that dictates the "learning rate". Also, $\epsilon = z(t) - \hat{\theta}^T(t)\phi(t)$ is the estimation error and $\hat{\theta}_0$ is an initial guess for uncertain parameters θ.

Third, we implement the gradient algorithm with parameters $\Gamma = \texttt{diag}(10^{-1}, 10^1, 10^1, 10^1, 10^{-1})$, $\hat{\theta}_0 = [3.5, 1, 0, 0, 0]^T$. The results are provided in Fig. 6, where the exit estimates are $\hat{\theta}_{t_{final}} = [3.504, 1.0185, -0.044, -0.037, 0.050]^T$. Recall the true parameter values are $\theta = [3.4707, 1.6112, -2.6287, 1.7175, 0.05]^T$. Observe the predicted voltage $\hat{V}(t) = \hat{p}_0(t) + \hat{p}_1(t)z(t) + \hat{p}_2(t)z^2(t) + \hat{p}_3(t)z^3(t) + V_c(t) + \hat{R}_1(t)I(t)$ converges to the true voltage $V(t)$. However, the parameter estimates $\hat{\theta}(t)$ do not all converge to the true values θ. This may or may not be an issue, depending on your particular objective. Is it critical to estimate the TRUE physical parameters, or simply obtain accurate predictions of battery voltage? In this example, the model parameters are only *partially identifiable* from this model and data. We discuss identifiability next.

5.3 Identifiability Test

Similar to the scalar case, we require an extra technical condition to ensure parameter vector $\hat{\theta}(t)$ converges asymptotically to the true parameter θ, given sufficient input-output data. As before, we can write the parameter estimation error dynamics as

$$\begin{aligned} \dot{\tilde{\theta}}(t) &= \dot{\theta} - \dot{\hat{\theta}} = -\Gamma\phi\epsilon = -\Gamma\phi(\theta^T\phi - \hat{\theta}^T\phi) = -\Gamma\phi(\phi^T\theta - \phi^T\hat{\theta}), \\ &= -\Gamma\phi\phi^T\tilde{\theta}, \\ &= -\Gamma A(t)\tilde{\theta}(t) \end{aligned} \tag{46}$$

where Γ is a diagonal positive-definite matrix and $A(t) = \phi(t)\phi^T(t)$. Recalling our notion of asymptotic stability for an autonomous linear time-invariant (LTI) systems of the form $\dot{x}(t) = A_{LTI}x(t)$ from Chapter 1, we require that A_{LTI} have eigenvalues that all have strictly negative real parts. In the present situation, we have a linear time-varying system, which requires a slightly more technical condition, known as *Persistent Excitation*. Intuitively, it encapsulates the fact that the data must be sufficiently rich to excite the dynamics such that the true parameters can be inferred.

Definition 5.1. *(Persistent Excitation (PE)) The (bounded) regressor signal $\phi(t)$ is called PE with*

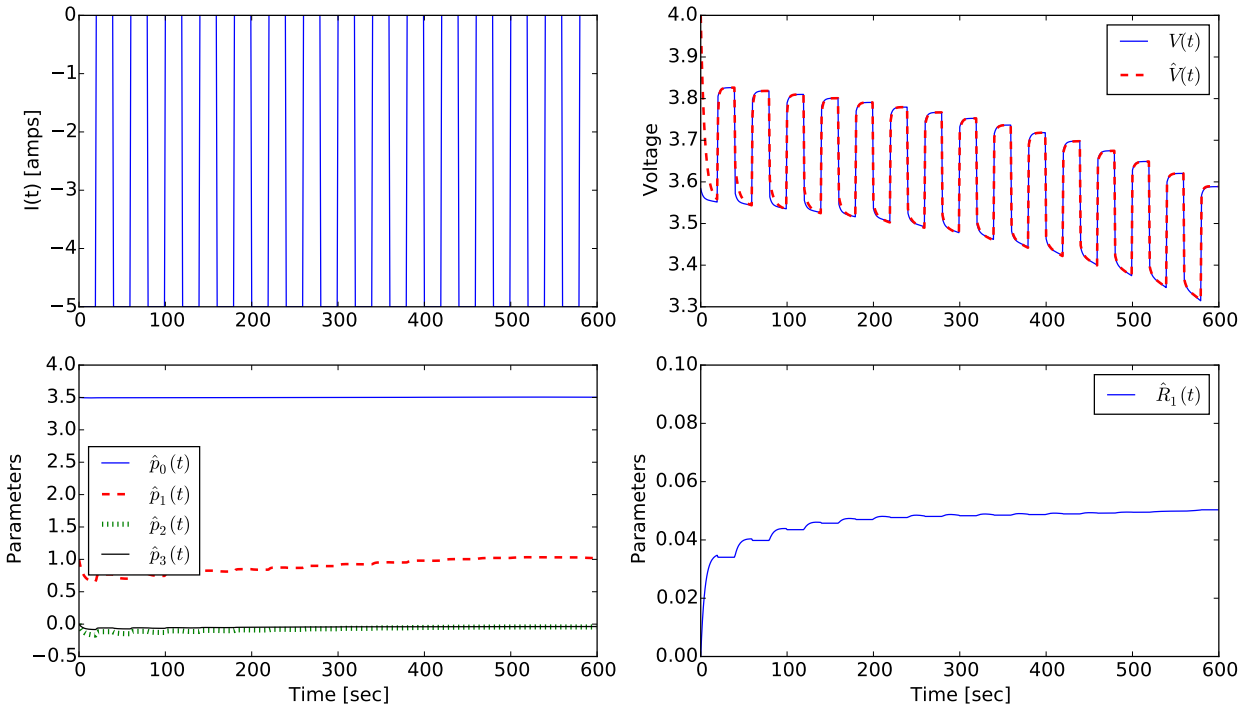

Figure 6: Parameter estimates of HW1 voltage model, using the gradient algorithm.

a level of excitation α if there exists some constant T such that

$$\frac{1}{T}\int_t^{t+T} \phi(\tau)\phi^T(\tau)d\tau \geq \alpha I, \qquad \forall t \geq 0 \tag{47}$$

If the regressor signal $\phi(t)$ is PE, then $\hat{\theta}(t) \to \theta$ as $t \to \infty$. Note that a sufficient condition to satisfy the PE definition (47), which can be checked numerically, is

$$\lambda_{\min}\left\{\int_t^{t+T} \phi(\tau)\phi^T(\tau)d\tau\right\} > 0 \tag{48}$$

where $\lambda_{\min}(\cdot)$ signifies the minimum eigenvalue of $(\cdot)$.

Remark 5.1. Note that the PE condition means $A(t)$ in the dynamic system (46) must be "positive" in some sense. Specifically, the integrated value of $A(t)$ over some time interval of size T must be positive definite. The "more" positive-definite the left-hand-side of (47) is, the faster the estimate $\hat{\theta}(t)$ converges to the true value θ. This convergence speed can be characterized by the level of excitation, α. We summarize this important result in the following theorem.

Theorem 5.1 (Gradient Algorithm Parameter Convergence, adopted from Corollary 4.3.1 of [2])**.** *Consider parametric model $z(t) = \hat{\theta}^T(t)\phi(t)$ and update law $\dot{\hat{\theta}}(t) = \Gamma\phi(t)\epsilon(t)$ with Γ diagonal and positive definite, as defined in (37)-(41). If $\phi(t)$ is PE according to Definition 5.1 and $\phi(t), \dot{\phi}(t)$ are*

bounded (i.e. stay away from infinity), then $\hat{\theta}(t) \to \theta$ exponentially fast.

Proof. The proof is long and pedantic for an applied course on Energy Systems and Control. We refer interested readers to Section 4.8 of [2]. □

The practical utility of this theorem is that checking if $\phi(t)$ is PE is the key step to determining if the gradient algorithm converges exponentially fast to true parameter vector θ.

Remark 5.2. For non-identifiable models and data, the PE level is - in practice - never exactly zero. In fact, it is usually some very small number close to zero, e.g. 10^{-6}. This leads to the natural question: Does PE practically enable us to determine identifiability or non-identifiability? As an absolute numeric value, the PE level is not very useful. However, the PE level metric allows us to compare different data sets or different models to use for model identification.

Remark 5.3. As energy systems engineers, we can often design our input-output data. The consequence of this is that data can be selected such that ϕ is PE. Checking this PE property a priori provides a certificate that the data will indeed render estimates that converge to the true values. More interestingly, we can seek to optimize $\phi(t)$ such that the left-hand-side of (48) is maximum, thereby producing the richest possible data for model identification.

Example 5.2 (Battery Voltage)**.** Consider again the equivalent circuit model shown from Ex. 5.1. The parametric model is given by $z = \theta^T \phi$, where

$$z(t) = V(t) - V_c(t), \qquad \theta = [p_0, p_1, p_2, p_3, R_1]^T, \qquad \phi(t) = \left[1, z(t), z^2(t), z^3(t), I(t)\right]^T \tag{49}$$

Using the -5A pulsed current input in Fig. 6, the PE level can be calculated via (48) to be: 10^{-6}. For comparison, let us consider a *linear* OCV function, which gives the parametric model $z = \vartheta^T \varphi$:

$$z(t) = V(t) - V_c(t), \qquad \vartheta = [p_0, p_1, R_1]^T, \qquad \varphi(t) = [1, z(t), I(t)]^T \tag{50}$$

This parametric model has only 3 parameters, whereas the previous had 5 parameters. Computing the PE level via (48) yields: 0.013. Consequently, the smaller parameterization has strong identifiability properties, despite using a linear OCV function instead of the true cubic function.

We might also consider different data. Suppose we apply a constant -1A current input to the battery model. The PE level for both parametric models yields a trivially small result: 10^{-16}. This implies the pulsed current is better for model identification than the constant current. The results are summarized in Table 1.

6 Recursive Least Squares Algorithm

Least-squares is an old method dating back to Gauss in the eighteenth century where he used it to determine the orbit of planets. The basic idea behind the least-squares is fitting a mathematical

Table 1: PE Level for Equivalent Circuit Battery Model

Data	-5A pulse	-1A constant
θ	10^{-6}	10^{-16}
ϑ	0.013	10^{-16}

model to a sequence of observed data by minimizing the sum of the squares of the difference between the observed and computed data. In doing so, any noise or inaccuracies in the observed data are expected to have less effect on the accuracy of the mathematical model. Next we derive a recursive version of the famous least squares algorithm, known as recursive least squares (RLS).

The method is simple to apply and analyze for linear-in-the-parameters models of the form

$$z(t) = \theta^T \phi(t) \tag{51}$$

As before, consider the estimate $\hat{z}$ of z and the estimation error are generated as

$$\hat{z}(t) = \hat{\theta}^T(t)\phi(t), \qquad \epsilon(t) = z(t) - \hat{\theta}^T(t)\phi(t) \tag{52}$$

In this case, we are interested in minimizing a cost function which integrates the squared error over time, as follows

$$J(\hat{\theta}) = \frac{1}{2}\int_0^t \epsilon(\tau)^2 d\tau = \frac{1}{2}\int_0^t \left[z(\tau) - \hat{\theta}^T(t)\phi(\tau)\right]^2 d\tau \tag{53}$$

The cost $J(\hat{\theta})$ penalizes all the past errors from $\tau = 0$ to t that are due to $\hat{\theta}(t) \neq \theta$. Note the conceptual difference between this cost function, and the cost function (40) used for the gradient algorithm. Specifically, the gradient algorithm minimizes an *instantaneous* cost, whereas RLS minimizes an *integrated* cost. This is a crucial point, conceptually. Since $J(\hat{\theta})$ is a convex function over $\mathbb{R}$ at each time t, its minimizer $\hat{\theta}$ satisfies

$$0 = \frac{d}{d\hat{\theta}} J(\hat{\theta}) = \frac{dJ}{d\epsilon}\frac{d\epsilon}{d\hat{\theta}} = -\int_0^t \epsilon(\tau)\phi^T(\tau)d\tau = \int_0^t \left[\hat{\theta}^T(t)\phi(\tau) - z(\tau)\right]\phi^T(\tau)d\tau, \tag{54}$$

$$0 = \hat{\theta}^T(t)\int_0^t \phi(\tau)\phi^T(\tau)d\tau - \int_0^t z(\tau)\phi^T(\tau)d\tau \tag{55}$$

for any given time t, where $dJ(\hat{\theta})/d\hat{\theta} \in \mathbb{R}^{1\times n}$ is defined as a row vector. Solving for $\hat{\theta}(t)$ results in

$$\hat{\theta}(t) = P(t)\int_0^t z(\tau)\phi(\tau)d\tau \tag{56}$$

where

$$P(t) = \left[\int_0^t \phi(\tau)\phi^T(\tau)d\tau\right]^{-1} \tag{57}$$

provided the inverse exists. This is the celebrated offline (a.k.a. batch or non-recursive) least squares algorithm.

We can derive a recursive method using the identity

$$\frac{d}{dt}PP^{-1} = \dot{P}P^{-1} + P\frac{d}{dt}P^{-1} = 0 \tag{58}$$

to show that $P(t)$ satisfies the differential equation

$$\dot{P}(t) = -P(t)\phi(t)\phi^T(t)P(t), \qquad P(0) = P_0 \tag{59}$$

where $P_0 = \text{diag}(p_{01}, p_{02}, \cdots, p_{0n}) \succ 0$ is diagonal and positive definite. The implication of (59) is the calculation of the inverse in (57) can be avoided by generating P as the solution to the matrix differential equation (59). Similarly, differentiating $\hat{\theta}(t)$ w.r.t. t in (56) and using (59) we can derive the relation

$$\dot{\hat{\theta}}(t) = P(t)\phi(t)\epsilon(t) \tag{60}$$

To summarize, this produces the continuous-time recursive or online least-squares algorithm,

$$\dot{\hat{\theta}}(t) = P(t)\phi(t)\epsilon(t), \qquad \hat{\theta}(0) = \hat{\theta}_0, \tag{61}$$

$$\dot{P}(t) = -P(t)\phi(t)\phi^T(t)P(t), \qquad P(0) = P_0 \tag{62}$$

Exercise 4. *Derive (59) by substituting (57) into (58) and solving for $\dot{P}$. Next, derive (60) by differentiating $\hat{\theta}(t)$ w.r.t. t in (56) and using (59).*

For recursive least squares (RLS), one can also prove convergence to the true parameter values. We summarize this result with the following theorem.

Theorem 6.1 (RLS Parameter Convergence, adopted from Corollary 4.3.4 of [2])**.** *Consider parametric model $z(t) = \hat{\theta}^T(t)\phi(t)$ and RLS update law $\dot{\hat{\theta}}(t) = P(t)\phi(t)\epsilon(t)$, $\dot{P}(t) = -P(t)\phi(t)\phi^T(t)P(t)$ with $P(0) = P_0$ diagonal and positive definite, as defined in (37) and (61)-(62). If $\phi(t)$ is PE according to Definition 5.1 and $\phi(t)$ is bounded (i.e. stay away from infinity), then $\hat{\theta}(t) \to \theta$ as $t \to \infty$.*

Proof. The proof is long and pedantic for an applied course on Energy Systems and Control. We refer interested readers to Section 4.8 of [2]. □

The practical utility of this theorem is that checking if $\phi(t)$ is PE is the key step to determining if RLS converges asymptotically fast to true parameter vector θ.

Example 6.1 (Battery Voltage)**.** Consider once again the equivalent circuit model shown from Ex. 5.1. The parametric model is given by $z = \theta^T \phi$, where

$$z(t) = V(t) - V_c(t), \qquad \theta = [p_0, p_1, p_2, p_3, R_1]^T, \qquad \phi(t) = \left[1, z(t), z^2(t), z^3(t), I(t)\right]^T \tag{63}$$

We apply the RLS algorithm given by

$$\dot{\hat{\theta}}(t) = P(t)\phi(t)\epsilon(t), \qquad \hat{\theta}(0) = \hat{\theta}_0, \tag{64}$$

$$\dot{P}(t) = -P(t)\phi(t)\phi^T(t)P(t), \qquad P(0) = P_0 \tag{65}$$

where $P_0 = \text{diag}(p_{01}, p_{02}, p_{03}, p_{04}, p_{05}) \succ 0$ is a diagonal, positive-definite matrix. Think of P_0 as the initial "learning rate", which gets automatically scaled by 65. Also, $\epsilon = z(t) - \hat{\theta}^T(t)\phi(t)$ is the estimation error and $\hat{\theta}_0$ is an initial guess for uncertain parameters θ.

Next, we implement the RLS algorithm with parameters $P_0 = \texttt{diag}(10^{-1}, 10^1, 10^1, 10^1, 10^{-1})$, $\hat{\theta}_0 = [3.5, 1, 0, 0, 0]^T$. The results are provided in Fig. 7, where the exit estimates are $\hat{\theta}_{t_{final}} = [3.525, 0.898, -0.370, -0.246, 0.048]^T$. Recall the true parameter values are $\theta = [3.4707, 1.6112, -2.6287, 1.7175, 0.05]^T$. Also, note that some parameters estimated by RLS are slightly different than the exit estimates from the gradient law in Ex 5.1. Observe the predicted voltage $\hat{V}(t) = \hat{p}_0(t) + \hat{p}_1(t)z(t) + \hat{p}_2(t)z^2(t) + \hat{p}_3(t)z^3(t) + V_c(t) + \hat{R}_1(t)I(t)$ converges to the true voltage $V(t)$, but at a slower rate than the gradient algorithm. This effect is intuitive, since RLS penalizes the *integrated* square error as opposed to the *instantaneous* square error. In other words, the evolution of the parameter updates $\hat{\theta}(t)$ is less sensitive to prediction errors $\epsilon(t)$ with RLS than with the gradient algorithm.

Algorithm Embellishments and Variations

The version of least-squares introduced above is known as "pure" least squares, to differentiate it from the various embellished versions. For a comprehensive coverage of these various adaptive laws, please consult [2]. These alternate versions impose constraints on the estimated parameters (projection), weight recent data more heavily than older data (forgetting factors), and reset the covariance (covariance resetting). A summary of various adaptive laws is given in the Appendix, for interested readers.

Exercise 5. *A popular embellishment of the least squares cost function in (53) is to add a penalty on the parameter magnitude. These penalties ameliorate ill-conditioned problems, and are generally called "regularization" in mathematics and statistics. For example, consider the integrated square error with weighted L_2 penalty:*

$$J(\hat{\theta}) = \frac{1}{2}\int_0^t \epsilon^2(\tau)d\tau + \frac{1}{2}\hat{\theta}^T\Gamma\hat{\theta} = \frac{1}{2}\int_0^t \left[z(\tau) - \hat{\theta}^T(t)\phi(\tau)\right]^2 d\tau + \frac{1}{2}\hat{\theta}^T(t)\Gamma\hat{\theta}(t) \tag{66}$$

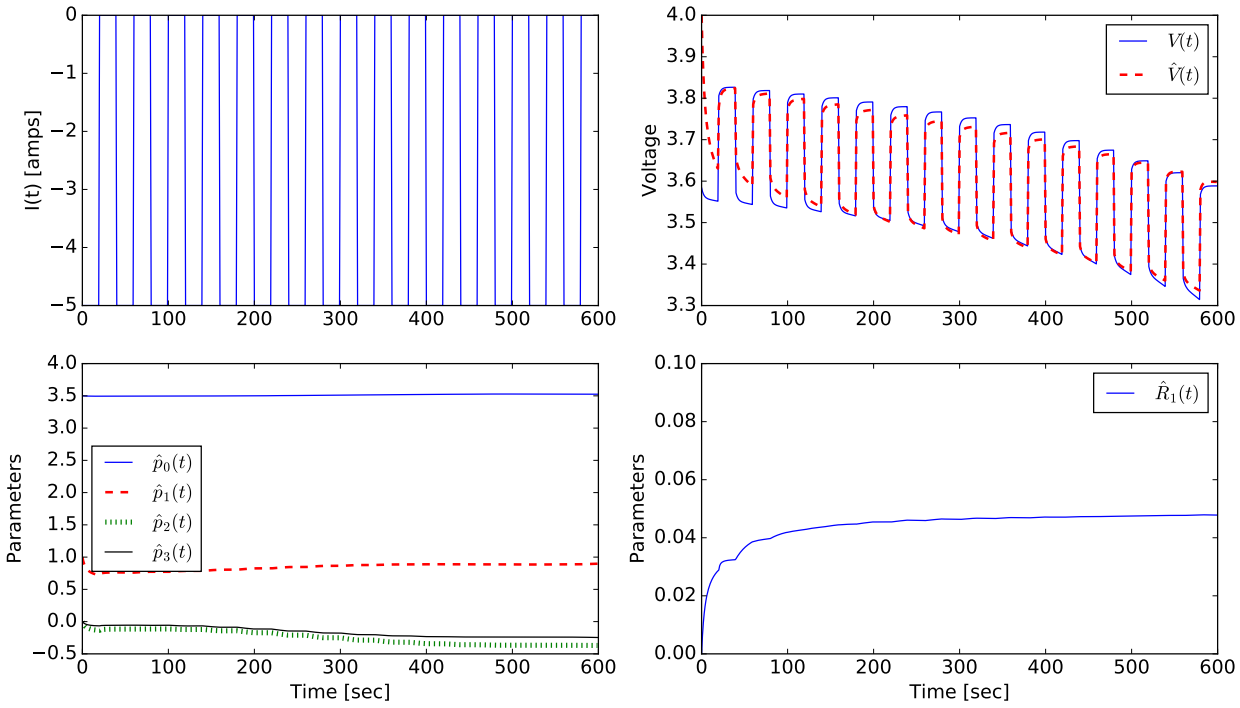

Figure 7: Parameter estimates of HW1 voltage model, using recursive least squares.

where Γ is a diagonal positive definite weighting matrix. This cost function with L_2 penalty is known as Tikhonov regularization or "ridge regression" in the statistics literature. For this exercise, derive the updates laws for $\hat{\theta}(t)$ and $P(t)$ for the cost function in (66). Your answer will be analogous to (61)-(62). HINT: The answer is exactly (61)-(62), except the initial condition for $P(t)$ is $P(0) = \Gamma^{-1}$.

7 Nonlinear Least Squares Algorithm

In practice, we often encounter nonlinear-in-the-parameters models, such as (15). Examples of nonlinear-in-the-parameters models include activation energy E in the Arrhenius law $k = Ae^{-E/RT}$, shape parameter ε in the radial basis function $\phi(r) = e^{-(\varepsilon r)^2}$, and frequency ω in the harmonic function $\sin(\omega t) + \cos(\omega t)$. In fact, most neural network models (including the recently popular class of "deep learning networks") are characterized by nonlinear-in-the-parameter models. Nonlinear least squares is by far the most popular algorithm for handling these models. The key steps of this algorithm are:

1. Re-express the nonlinear model in terms of the parameter estimation error $\tilde{\theta} = \theta - \hat{\theta}$.
2. Perform a Taylor series expansion around $\tilde{\theta} = 0$.
3. Define a linearized parametric model based upon a first-order Taylor series approximation

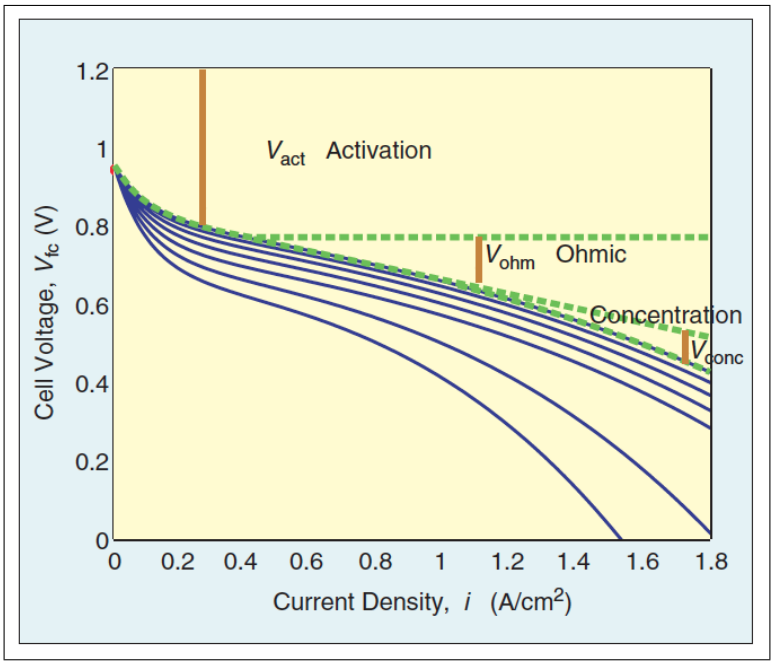

Figure 8: Fuel cell polarization curve from [4].

4. Apply least squares, using a nonlinear estimation error.

We demonstrate this algorithm by example.

Example 7.1 (Fuel Cell Polarization Curve)**.** Figure 8 presents the output voltage of a fuel cell, as a function of the current density [A/cm^2]. This is known as the fuel cell polarization curve. The fuel cell voltage is calculated using a combination of physical and empirical relationships, and is given by [3]

$$V_{fc} = E - V_{act} - V_{ohm} - V_{conc} \tag{67}$$

where E is the open circuit voltage, and $V_{act}, V_{ohm}, V_{conc}$ are activation, ohmic, and concentration over voltages, which represent losses due to various physical or chemical factors. For pedagogical purposes, we assume V_{conc} is negligible. The remaining loss terms are given by

$$V_{act} = v_a \left(1 - e^{-c \cdot i}\right), \tag{68}$$

$$V_{ohm} = Ri \tag{69}$$

where E, v_a, c, R are unknown parameters to be identified. The problematic parameter is c, since this parameter is nonlinear in the expression. **Step 1:** We redefine the parameters as $\theta_1 = E, \theta_2 = v_a, \theta_3 = c, \theta_4 = R$ and re-write the polarization curve equation in terms of the parameter error as

$$V_{fc} = (\tilde{\theta}_1 + \hat{\theta}_1) - (\tilde{\theta}_2 + \hat{\theta}_2)\left[1 - e^{-(\tilde{\theta}_3 + \hat{\theta}_3)i}\right] - (\tilde{\theta}_4 + \hat{\theta}_4)i \tag{70}$$

Step 2: Next we perform a Taylor series expansion around $\tilde{\theta} = 0$,

$$\begin{aligned} V_{fc} &= \hat{\theta}_1 - \hat{\theta}_2 \left[1 - e^{-\hat{\theta}_3 i}\right] - \hat{\theta}_4 i \\ &\quad + \tilde{\theta}_1 - \left[1 - e^{-\hat{\theta}_3 i}\right] \cdot \tilde{\theta}_2 - \hat{\theta}_2 i e^{-\hat{\theta}_3 i} \cdot \tilde{\theta}_3 - i \cdot \tilde{\theta}_4 + (\text{higher order terms}) \end{aligned} \tag{71}$$

Step 3: Truncating and re-arranging terms, we arrive at the linearized parametric model

$$e_{nl} = \tilde{\theta}^T \phi \tag{72}$$

where

$$e_{nl} = V_{fc} - \hat{\theta}_1 + \hat{\theta}_2 \left[1 - e^{-\hat{\theta}_3 i}\right] + \hat{\theta}_4 i, \tag{73}$$

$$\phi = \begin{bmatrix} 1 \\ e^{-\hat{\theta}_3 i} - 1 \\ -\hat{\theta}_2 i e^{-\hat{\theta}_3 i} \\ -i \end{bmatrix} \tag{74}$$

and both e_{nl} and ϕ are functions of the current parameter estimate $\hat{\theta}$. **Step 4:** is to apply the least squares parameter update law as

$$\dot{\hat{\theta}} = P e_{nl} \phi, \qquad \hat{\theta}(0) = \hat{\theta}_0 \tag{75}$$

$$\dot{P} = -P \phi \phi^T P, \qquad P(0) = P_0 = P_0^T > 0 \tag{76}$$

This algorithm has *local* stability properties. That is, when the parameter estimate $\hat{\theta}$ is near its true value, then the linear approximation is reasonably accurate and $\hat{\theta}$ converges asymptotically to the true value θ under an appropriate persistence of excitation condition for the regressor ϕ.

Exercise 6 (Photovoltaic Array Learning)**.** *Consider a photovoltaic (PV) array. Figure 9 [top] presents the characteristic nonlinear relationship between the array's voltage and current. Multiplying the current by voltage yields the concave relationship between array current and output power in Fig. 9 [bottom]. Maximum power point tracking (MPPT) algorithms seek to maintain the PV at maximum output power, in the face of varying ambient temperatures and incident solar irradiation (see [5] and references therein).*

In this exercise, you seek to learn the characteristic nonlinear relationship in Figure 9, using current $I(t)$ and voltage $V(t)$ data. Since these curves shift with varying temperature and irradiation levels, you seek to learn this relationship in real-time. Derive a nonlinear least squares algorithm to identify $\theta = [\theta_1, \theta_2, \theta_3, \theta_4, \theta_5]^T$ using the following nonlinear-in-the-parameters model:

$$\theta_1 - \theta_2 \left[e^{(\theta_3 V(t) + \theta_4 I(t))} - 1\right] - \theta_5 \left[\theta_3 V(t) + \theta_4 I(t)\right] = 0 \tag{77}$$

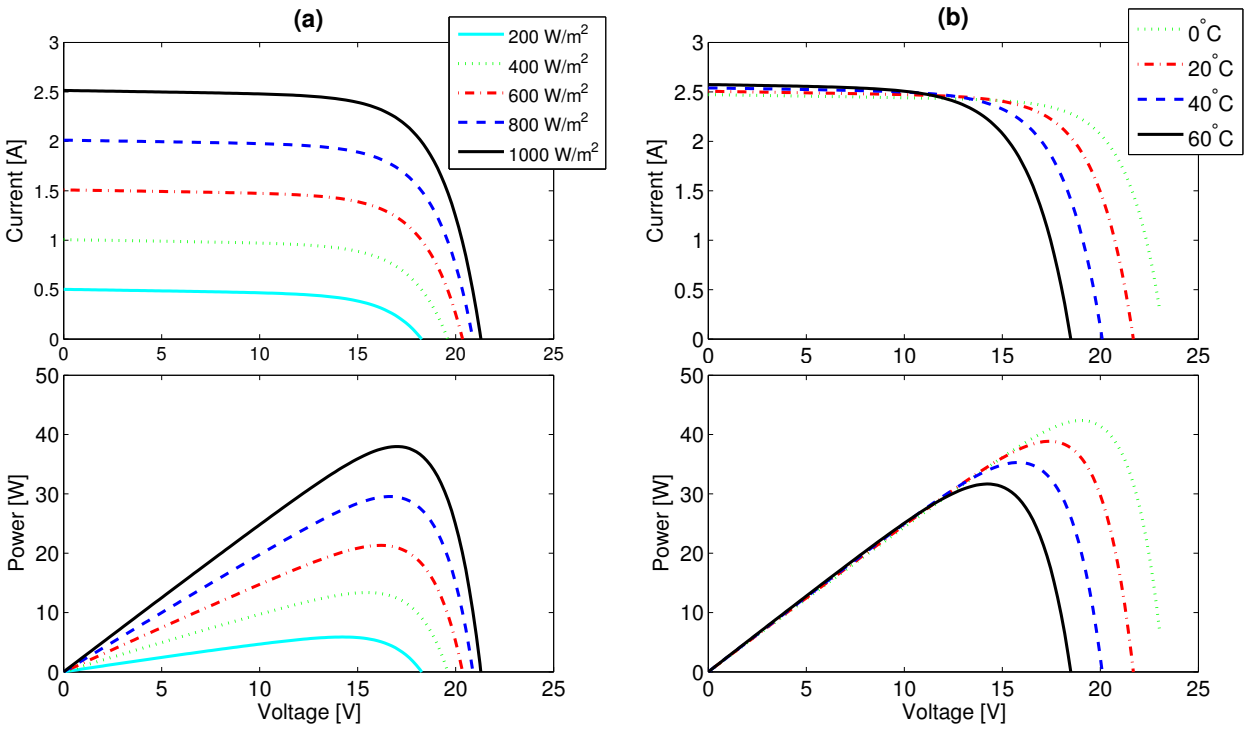

Figure 9: Characteristic I-V and P-V curves for (a) varying irradiation levels and T = 25°C, (b) varying temperature levels and S = 1000W/m^2.

Derive expressions for e_{nl} and ϕ. Note, this model form is not arbitrary nor eye-balled from data. (You are too skilled to accept such amateur approaches). It comes from an equivalent circuit model detailed in Chapter 5 of Masters [6].

8 Offline Techniques

Offline model identification techniques consider scenarios where data is collected from the energy system as a batch set. This data is then used for model identification, separate from the dynamic evolution of the energy system itself. As a result, we do not recursively estimate the parameter values as data arrives. Instead, we fit the model parameters in a single computational procedure. In this section, we define this problem category and a general formulation. However, we do not provide any specific algorithms and refer the reader to the optimization literature.

Consider the nonlinear-in-the-parameters model

$$y = f(u;\theta) \tag{78}$$

where $y \in \mathbb{R}^{n_y}$ is the model output, $\theta \in \mathbb{R}^{n_\theta}$ is the true parameter vector, and $u \in \mathbb{R}^{n_u}$ is a

measured input vector that may or may not be present in the model. We can then formulate an estimated output $\hat{y}$ from estimated model parameters $\hat{\theta}$ as follows

$$\hat{y} = f(u; \hat{\theta}). \tag{79}$$

With this notation, we can formulate the generic offline model identification problem as an optimization task

$$\min_{\hat{\theta}} \quad \|y - \hat{y}\| = \|y - f(u; \hat{\theta})\| \tag{80}$$

$$\text{subject to} \quad \hat{\theta} \in \mathcal{S} \tag{81}$$

where $\|\cdot\|$ in (80) is an appropriate norm (e.g. sum of squares, maximum magnitude, integrated over time) and $\mathcal{S}$ in (81) is a set of feasible values for the unknown parameters $\hat{\theta}$.

Depending on the mathematical structure and size of the cost function and constraints (80)-(81), various optimization techniques can be pursued. We provide two categories of techniques and associated keywords that interested readers can search for further information

- **[Gradient-based Methods]** include the classical optimization techniques, such as gradient-descent, quasi-Newton methods, and convex programming. These methods generally contain attractive convergence and optimality properties. However, they require the cost/constraints to be differentiable in the variable $\hat{\theta}$.
- **[Non-Gradient-based Methods]** are pursued when it is intractable to compute (analytically or numerically) the gradient. Several common methods include Genetic Algorithms, Particle Swarm Optimization, and Stochastic Optimization. These algorithms typically require less assumptions about the model structure, yet sacrifice computational efficiency and convergence/optimality properties.

9 Sensitivity Analysis

Sensitivity analysis addresses the identifiability question. That is, this analysis is a structured way to determine which parameters are identifiable. Moreover, it quantifies the level of identifiability, allowing one to rank which parameters have the greatest influence on the output of interest. This analysis is specifically oriented toward nonlinear-in-the-parameters models, where the output does NOT have a linear dependence on the parameters. Despite our previous presentation of gradient, least squares, and nonlinear least squares algorithms, sensitivity analysis is a necessary first step in nonlinear parameterized models of energy systems.

Consider the nonlinear-in-the-parameters model

$$y = f(u; \theta) \tag{82}$$

where $y \in \mathbb{R}^{n_y}$ is the model output, $\theta \in \mathbb{R}^{n_\theta}$ is the parameter vector, and $u \in \mathbb{R}^{n_u}$ is a measured input vector that may or may not be present in the model. The sensitivity derivative of the model outputs to the parameter vector is defined as

$$S = \frac{\partial f}{\partial \theta}(u;\theta) \in \mathbb{R}^{n_y \times n_\theta}, \tag{83}$$

$$S = [S_1, S_2, \cdots, S_{n_\theta}] = \left[\frac{\partial f}{\partial \theta_1}(u;\theta), \frac{\partial f}{\partial \theta_2}(u;\theta), \cdots, \frac{\partial f}{\partial \theta_{n_\theta}}(u;\theta)\right] \tag{84}$$

Each column of the sensitivity derivative, which we denote S_i represents one parameter's sensitivity in all outputs, and is viewed as the sensitivity direction for the corresponding parameter. Since the model is nonlinear in the parameters, global identifiability cannot be proven. That is, the parameter's sensitivity in an output may depend on the particular parameter values. Moreover, it may also depend on the input vector u. In addition, only a subset of the parameters may be identifiable. This occurs when linear dependence exists between columns of the sensitivity derivative. Intuitively, this means an output produces nearly identical reactions when two different parameters are perturbed. As a result, it's essentially impossible to differentiate between these parameters

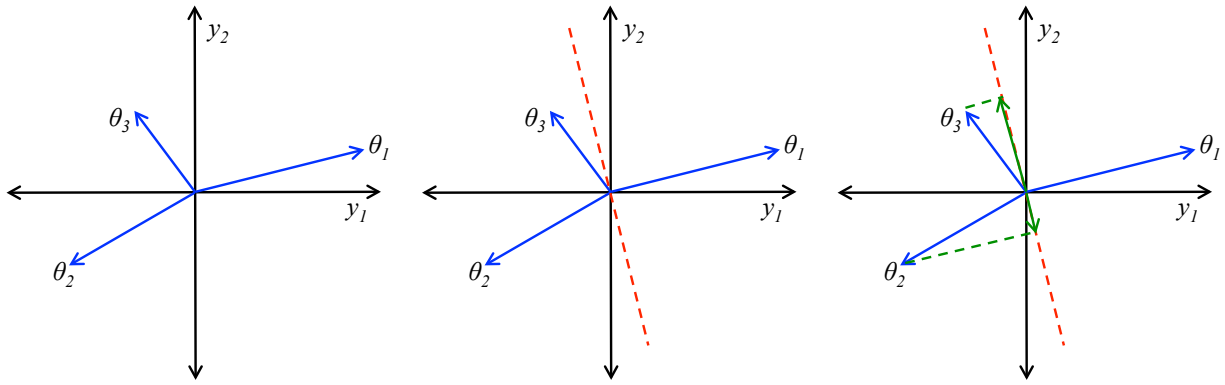

Figure 10: Graphical description of Linear Dependence in Parameter Sensitivity

by observing the output. These complications motivate the need for an efficient and structured method, which performs the following tasks:

1. **Rank parameter sensitivity:** This allows us to focus on identifying the most influential parameters.

2. **Identify linear dependence between parameter sensitivities:** This allows us to understand which combinations of parameters are not identifiable, and focus only on the identifiable subset.

For the purposes of presenting this methodology, consider $n_\theta = 4$ parameters and an arbitrary number of output data n_y. Then the sensitivity derivative is composed of $S = [S_1, S_2, S_3, S_4] \in$

$\mathbb{R}^{n_y \times 4}$. A particular decomposition of matrix $S^T S$ reveals both sensitivity ranking and linear dependence. Let $S^T S = D^T C D$ where

$$D = \begin{bmatrix} \|S_1\| & 0 & 0 & 0 \\ 0 & \|S_2\| & 0 & 0 \\ 0 & 0 & \|S_3\| & 0 \\ 0 & 0 & 0 & \|S_4\| \end{bmatrix}, \qquad C = \begin{bmatrix} 1 & \frac{\langle S_1, S_2 \rangle}{\|S_1\| \|S_2\|} & \frac{\langle S_1, S_3 \rangle}{\|S_1\| \|S_3\|} & \frac{\langle S_1, S_4 \rangle}{\|S_1\| \|S_4\|} \\ \frac{\langle S_2, S_1 \rangle}{\|S_2\| \|S_1\|} & 1 & \frac{\langle S_2, S_3 \rangle}{\|S_2\| \|S_3\|} & \frac{\langle S_2, S_4 \rangle}{\|S_2\| \|S_4\|} \\ \frac{\langle S_3, S_1 \rangle}{\|S_3\| \|S_1\|} & \frac{\langle S_3, S_2 \rangle}{\|S_3\| \|S_2\|} & 1 & \frac{\langle S_3, S_4 \rangle}{\|S_3\| \|S_4\|} \\ \frac{\langle S_4, S_1 \rangle}{\|S_4\| \|S_1\|} & \frac{\langle S_4, S_2 \rangle}{\|S_4\| \|S_2\|} & \frac{\langle S_4, S_3 \rangle}{\|S_4\| \|S_3\|} & 1 \end{bmatrix}, \quad (85)$$

where $\| \cdot \|$ denotes the Euclidian norm ($\|x\| = \sqrt{x^T x}$) and $\langle \cdot, \cdot \rangle$ is the inner product ($\langle x, y \rangle = x^T y$). This decomposition can be generated by performing Gram-Schmidt orthonormalization of the sensitivity column vectors. This process coincides with the economical QR decomposition with permutation (see `qr` command in Matlab). Now we discuss how to interpret this decomposition.

1. **Rank parameter sensitivity:** Diagonal matrix D provides a direct quantification of parameter sensitivity. Sorting the elements $\|S_i\|$ in descending order provides the desired sensitivity ranking. Note that one should normalize the parameters and outputs across a suitable range to ensure a fair comparison.

2. **Linear Dependence:** By the Cauchy Schwarz inequality $-1 \leq \frac{\langle S_i, S_j \rangle}{\|S_i\| \|S_j\|} \leq 1$. This has the interpretation that values of $\frac{\langle S_i, S_j \rangle}{\|S_i\| \|S_j\|}$ near -1 or 1 imply strong linear dependence between parameters θ_i and θ_j, whereas values near zero imply orthogonality. In other words, when the off-diagonal element of matrix C is near -1 or 1, then the corresponding pair of parameters exhibit strong linear dependence, and are difficult to identify separately. When the off-diagonal element is near zero, then the corresponding pair of parameters exhibit strong linear independence, and can be easily identified separately. This concept is depicted graphically in Fig. 10. The sensitivity vectors corresponding to θ_1 and θ_2 have an inner product near -1, implying strong linear dependence. In contrast, the sensitivity vectors corresponding to θ_1 and θ_3 have an inner product near 0, implying strong linear independence. Thus, θ_1 and θ_2 are difficult to identify separately, whereas θ_1 and θ_3 are easier to identify separately.

Example 9.1 (State-of-Health Parameter Estimation in Batteries)**.** This example considers the identification of uncertain parameters in a nonlinear battery model based upon electrochemistry [7]. The nonlinear in the parameters model has the form:

$$V(t) = OCV + \sinh^{-1}\left[\frac{\theta_2}{\sqrt{\theta_1}} I(t)\right] - \sinh^{-1}\left[\frac{\theta_3}{\sqrt{\theta_1}} I(t)\right] + \theta_4 I(t) \quad (86)$$

where $V(t)$ is battery voltage, OCV is a known open circuit potential, $I(t)$ is battery current, and $\theta_1, \theta_2, \theta_3, \theta_4$ are unknown electrochemical model parameters. Interestingly, identification of these unknown parameters corresponds directly to assessing the battery's age - specifically capacity fade and power fade. For more details consult [7].

An example of matrices D and C is provided in (87). This example analyzes parameter sensitivity for a typical electric drive vehicle charge/discharge cycle.

$$D = \begin{bmatrix} 2.827 & 0 & 0 & 0 \\ 0 & 4.501 & 0 & 0 \\ 0 & 0 & 3.958 & 0 \\ 0 & 0 & 0 & 15.103 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & -0.3000 & 0.2908 & 0.2956 \\ -0.3000 & 1 & -0.9801 & -0.9805 \\ 0.2908 & -0.9801 & 1 & 0.9322 \\ 0.2956 & -0.9805 & 0.9322 & 1 \end{bmatrix}. \tag{87}$$

First, it is evident from D that the parameter sensitivity ranking is as follows: $\theta_4, \theta_2, \theta_3, \theta_1$. Accurate identification of these parameters should be prioritized accordingly. In addition, strong linear dependence exists between $\theta_2, \theta_3, \theta_4$. This means perturbations in any of these three parameters produces almost identical changes in our measured data. Consequently, attempting to individually separate these variables is futile. An infinite number of linear combinations of these parameters may satisfy our data. In this case, we should only identify the most sensitive parameter among these linearly dependent subset. As a result, we conclude the model is partially identifiable and focus on identifying the subset θ_1, θ_4 only.

In this particular application, it turns out θ_1 represents - exactly - charge capacity in the battery and θ_4 represents - exactly - internal resistance. Monitoring changes in these parameters yields directly quantifications of charge and power fade.

Exercise 7 (Photovoltaic Array Learning - continued)**.** *blah blah...*

10 Notes

Model identification is a rich and deep area, in both theory and applications. In this chapter, we attempt to provide students with an introduction, several popular tools, and motivation for further study. The quintessential reference for system identification is Lennart Ljung's textbook [1]. Model identification is also commonly used in the context of control, in what is generally known as "adaptive control." Excellent textbooks in this area include Ioannou & Sun [2] and Krstic, Kanellakopoulos, & Kokotovic [8]. Finally, Lund & Foss give a particularly excellent article on sensitivity analysis in nonlinear parameterized models [9].

References

[1] L. Ljung, System identification. Wiley Online Library, 1999.

[2] P. Ioannou and J. Sun, Robust Adaptive Control. Prentice-Hall, 1996.

[3] J. Pukrushpan, H. Peng, and A. G. Stefanopoulou, "Control-oriented modeling and analysis for automotive fuel cell systems," Journal of dynamic systems, measurement, and control, vol. 126, pp. 14–25, 2004.

[4] J. Pukrushpan, A. Stefanopoulou, and H. Peng, "Control of fuel cell breathing," Control Systems, IEEE, vol. 24, no. 2, pp. 30–46, Apr 2004.

[5] S. J. Moura and Y. A. Chang, "Lyapunov-based Switched Extremum Seeking for Photovoltaic Power Maximization," Control Engineering Practice, vol. 21, no. 7, pp. 971 – 980, 2013.

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

[7] S. J. Moura, N. Chaturvedi, and M. Krstic, "Adaptive PDE Observer for Battery SOC/SOH Estimation via an Electrochemical Model," ASME Journal of Dynamic Systems, Measurement, and Control, vol. 136, no. 1, pp. 011 015–011 026, Oct 2014.

[8] M. Krstic, P. V. Kokotovic, and I. Kanellakopoulos, Nonlinear and adaptive control design. John Wiley & Sons New York, 1995.

[9] B. F. Lund and B. A. Foss, "Parameter ranking by orthogonalization - Applied to nonlinear mechanistic models," Automatica, vol. 44, no. 1, pp. 278 – 281, 2008.

11 Appendix: Summary of Adaptive Laws

In this section, we summarize various model identification algorithms which are particularly useful in practice. These algorithms fall into two categories: Gradient Algorithms (Table 2) and Least Squares Algorithms (Table 3).

Adaptive laws with Normalization: Throughout these tables, we consider adaptive laws *with normalization*. These laws are identical to their non-normalized variants, except the estimation model is altered to $\epsilon = (z - \hat{z})/m^2$, where m^2 is known as the *normalization signal*. This endows the algorithms with additional boundedness properties, and generally provides better numerical conditioning.

Least-Squares with Forgetting Factor: Sometimes it is desirable to discount past data, relative to more recent data. This is particularly useful for model parameters that might slowly change over time. The least squares with forgetting factor is designed under this paradigm, where β is known as the forgetting factor. When $\beta = 0$, then one recovers the pure least squares method. As β increases towards ∞, then one approaches an instantaneous cost function.

Adaptive Laws with Projection: The parameter estimates $\hat{\theta}$ are generated by solving recursive optimization problems, with appropriately formulated cost functions. In practice, one may wish to constrain parameter estimates within a physically meaningful set. For example, heat transfer coefficients cannot be negative, meaningful thermal resistances have upper and lower limits. Motivated by this situation, we can add a convex constraint set to the optimization formulation. Mathematically, we enforce the constraint $\hat{\theta} \in \mathcal{S} = \{\hat{\theta} \in \mathbb{R}^n \mid g(\hat{\theta}) \leq 0\}$, where set $\mathcal{S}$ represents our convex constraint set that is defined by a function $g(\cdot)$ that maps parameter estimate vectors to scalars. In this algorithm, we additionally require that $g(\cdot)$ is a so-called "smooth" function. This means that $g(\cdot)$ and all its derivatives are continuous functions. This is a mathematical technicality, but can be easily achieved in practice. A common selection for the convex constraint set $\mathcal{S}$ is an ellipsoid, which is indeed parameterized by a smooth function $g(\cdot)$. In the algorithm listed in Table 3, int$(\mathcal{S})$ signifies the interior of set $\mathcal{S}$, and $\delta(\mathcal{S})$ signifies the boundary of set $\mathcal{S}$.

Table 2: Gradient Algorithms

Parametric Model	$z = \theta^T \phi$
Estimation model	$\hat{z} = \hat{\theta}^T \phi$
Normalized Estimation model	$\epsilon = \frac{z - \hat{z}}{m^2}$

Based on Instantaneous Cost

Cost	$J(\hat{\theta}) = \frac{1}{2}\epsilon^2$
Adaptive Law	$\dot{\hat{\theta}} = \Gamma \epsilon \phi$
Design variables	$m^2 = 1 + \alpha \phi^T \phi,\ \alpha > 0,\ \Gamma = \Gamma^T \succ 0$

Based on Integral Cost

Cost	$J(\hat{\theta}) = \frac{1}{2}\int_0^t e^{-\beta(t-\tau)} \epsilon^2(t,\tau) m^2(\tau) d\tau$
Adaptive Law	$\dot{\hat{\theta}} = -\Gamma(R\hat{\theta} + Q)$ $\dot{R} = -\beta R + \frac{\phi \phi^T}{m^2}, \quad R(0) = 0$ $\dot{Q} = -\beta Q + \frac{z\phi}{m^2}, \quad Q(0) = 0$
Design variables	$m^2 = 1 + \alpha \phi^T \phi,\ \alpha > 0,\ \beta > 0,\ \Gamma = \Gamma^T \succ 0$

Gradient Law with Projection

Cost	$J(\hat{\theta}) = \frac{1}{2}\epsilon^2$
subject to	$\hat{\theta} \in \mathcal{S} = \{\hat{\theta} \in \mathbb{R}^n \mid g(\hat{\theta}) \leq 0, g : \mathbb{R}^n \to \mathbb{R}$ is smooth fcn$\}$
Adaptive Law	$\dot{\hat{\theta}} = \text{Proj}(\Gamma \epsilon \phi) = \begin{cases} \Gamma \epsilon \phi & \text{if } \hat{\theta} \in \text{int}(\mathcal{S}), \text{ OR} \\ & \text{if } \hat{\theta} \in \delta(\mathcal{S}) \text{ and } (\Gamma \epsilon \phi)^T \nabla g \leq 0 \\ \Gamma \epsilon \phi - \Gamma \frac{\nabla g \nabla g^T}{\nabla g^T \Gamma \nabla g} \Gamma \epsilon \phi & \text{otherwise} \end{cases}$
Design variables	$m^2 = 1 + \alpha \phi^T \phi,\ \alpha > 0,\ \Gamma = \Gamma^T \succ 0,\ g(\cdot)$ smooth

Table 3: Least Squares Algorithms

Parametric Model	$z = \theta^T \phi$
Estimation model	$\hat{z} = \hat{\theta}^T \phi$
Normalized Estimation model	$\epsilon = \frac{z - \hat{z}}{m^2}$

Pure Least Squares with Normalization

Cost	$J(\hat{\theta}) = \frac{1}{2} \int_0^t \frac{[z(\tau) - \hat{\theta}^T(t)\phi(\tau)]^2}{m^2(\tau)} d\tau$
Adaptive Law	$\dot{\hat{\theta}} = P \epsilon \phi$ $\dot{P} = -P \frac{\phi \phi^T}{m^2} P, \quad P(0) = P_0$
Design variables	$m^2 = 1 + \alpha \phi^T \phi, \ \alpha > 0, \ P_0 = P_0^T \succ 0$

Least Squares with Forgetting Factor

Cost	$J(\hat{\theta}) = \frac{1}{2} \int_0^t e^{-\beta(t-\tau)} \frac{[z(\tau) - \hat{\theta}^T(t)\phi(\tau)]^2}{m^2(\tau)} d\tau + \frac{1}{2} e^{-\beta t} \left[\hat{\theta}(t) - \hat{\theta}_0\right]^T Q_0 \left[\hat{\theta}(t) - \hat{\theta}_0\right]$
Adaptive Law	$\dot{\hat{\theta}} = P \epsilon \phi$ $\dot{P} = \beta P - P \frac{\phi \phi^T}{m^2} P, \quad P(0) = P_0 = Q_0^{-1}$
Design variables	$m^2 = 1 + \alpha \phi^T \phi, \ \alpha > 0, \ \beta > 0, \ P_0 = P_0^T \succ 0$

Least Squares with Forgetting Factor and Projection

Cost	$J(\hat{\theta}) = \frac{1}{2} \int_0^t e^{-\beta(t-\tau)} \frac{[z(\tau) - \hat{\theta}^T(t)\phi(\tau)]^2}{m^2(\tau)} d\tau + \frac{1}{2} e^{-\beta t} \left[\hat{\theta}(t) - \hat{\theta}_0\right]^T Q_0 \left[\hat{\theta}(t) - \hat{\theta}_0\right]$
subject to	$\hat{\theta} \in \mathcal{S} = \{\hat{\theta} \in \mathbb{R}^n \mid g(\hat{\theta}) \leq 0, g : \mathbb{R}^n \to \mathbb{R} \text{ is smooth fcn}\}$
Adaptive Law	$\dot{\hat{\theta}} = \text{Proj}(P\epsilon\phi) = \begin{cases} P\epsilon\phi & \text{if } \hat{\theta} \in \text{int}(\mathcal{S}), \text{ OR} \\ & \text{if } \hat{\theta} \in \delta(\mathcal{S}) \text{ and } (P\epsilon\phi)^T \nabla g \leq 0 \\ P\epsilon\phi - P \frac{\nabla g \nabla g^T}{\nabla g^T P \nabla g} P \epsilon \phi & \text{otherwise} \end{cases}$ $\dot{P} = \begin{cases} \beta P - P \frac{\phi \phi^T}{m^2} P & \text{if } \hat{\theta} \in \text{int}(\mathcal{S}), \text{ OR} \\ & \text{if } \hat{\theta} \in \delta(\mathcal{S}) \text{ and } (P\epsilon\phi)^T \nabla g \leq 0 \\ 0 & \text{otherwise} \end{cases}$
Design variables	$m^2 = 1 + \alpha \phi^T \phi, \ \alpha > 0, \ \beta > 0, \ P_0 = P_0^T \succ 0, \ g(\cdot)$ smooth