2.160 IDENTIFICATION, ESTIMATION, AND LEARNING

# LECTURE NOTES NO. 2

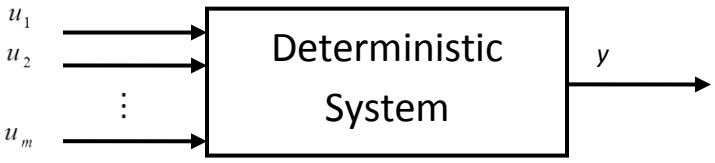
# *PART 1: Regression*

## 2. PARAMETER ESTIMATION FOR DETERMINISTIC SYSTEMS

This chapter's materials are based on Goodwin and Sin's textbook, "Adaptive Filtering, Prediction, and Control", Chapter 3.

### 2.1 LEAST SQUARES ESTIMATION

Consider a deterministic system with a linear input-output model, as shown below.




Linearly parameterized model

Input-output model $y = b_1 u_1 + b_2 u_2 + \ldots + b_m u_m$

Parameters to estimate: $\theta = [b_1 \quad \ldots \quad b_m]^T \in R^m$

Input and output:

$$\begin{cases} \varphi = [u_1 \quad \ldots \quad u_m]^T \in R^m \\ y = \varphi^T \theta \end{cases} \tag{1}$$

The problem is to find parameters $\theta = [b_1 \quad \ldots \quad b_m]^T$ from an observation dataset:

$$\left.\begin{matrix} \varphi(1), y(1) \\ \varphi(2), y(2) \\ \vdots \\ \varphi(N), y(N) \end{matrix}\right\} \rightarrow \theta$$

[Remark] Although this input-output model looks linear and algebraic, we can represent a dynamical system, linear or nonlinear, in this form.

a linear dynamical system, e.g. $y(t) = b_1 u(t-1) + b_2 u(t-2) + \cdots\cdots + b_m u(t-m)$

$$\varphi(t) = [u(t-1), u(t-2), \cdots, u(t-m)]^T \in R^m$$

or

a nonlinear dynamic system, e.g. $y(t) = b_1 u(t-1) + b_2 u(t-2)u(t-1)$

$$\varphi(t) = [u(t-1), u(t-2)u(t-1)]^T$$

Note that the parameters, $b_1$, $b_2$,..... are ***linearly*** involved in the input-output equation. As long as the parameters to estimate are linearly involved in the input-output equation, we can apply the following estimation theory. The vector $\varphi(t)$ is referred to as a ***regressor***.

Using an estimated parameter vector $\hat{\theta}$, we can write a predictor that predicts the output from inputs:

$$\hat{y}(t|\theta) = \varphi(t)^T \hat{\theta} \tag{2}$$

We evaluate the predictor's performance by the mean squared error given by

$$V_N(\theta) = \frac{1}{N}\sum_{t=1}^{N}(\hat{y}(t \mid \theta) - y(t))^2 \tag{3}$$

Problem: Find the parameter vector $\hat{\theta}$ that minimizes the squared error:

$$\hat{\theta} = \arg\min_{\theta} V_N(\theta) \tag{4}$$

Differentiating $V_N(\theta)$ and setting it to zero,

$$\frac{dV_N(\theta)}{d\theta} = 0 \quad \Longrightarrow \quad \frac{2}{N}\sum_{t=1}^{N}(\varphi^T(t)\theta - y(t))\varphi(t) = 0 \tag{5}$$

Note that the above expression is the differentiation of a scalar function $V_N$ with a vector $\theta \in \Re^m$. Rearranging (5) yields

$$\underbrace{\left[\sum_{t=1}^{N}\phi(t)\phi^T(t)\right]}_{\Phi\Phi^T}\theta = \sum_{t=1}^{N}y(t)\phi(t) \tag{6}$$

Note that, since $\varphi^T\theta$ is a scalar, we can move it before or after a vector: $(\varphi^T\theta)\varphi = \varphi(\varphi^T\theta)$.

Concatenating the $N$ regressor vectors, we can define an $m \times N$ matrix

$$\Phi = [\varphi(1) \quad \varphi(2) \quad \ldots \quad \varphi(N)] \tag{7}$$

$$m \le N$$

If vectors $\phi(1), \phi(2), \cdots, \phi(N)$ span the whole *m*-dimensional vector space, $rank\ \Phi = m$; full rank. Then matrix $\Phi\Phi^T \in \mathbb{R}^{m\times m}$ is invertible, since $rank\Phi = rank\Phi\,\Phi^T = m$.

Under this condition, the optimal parameter vector is given by

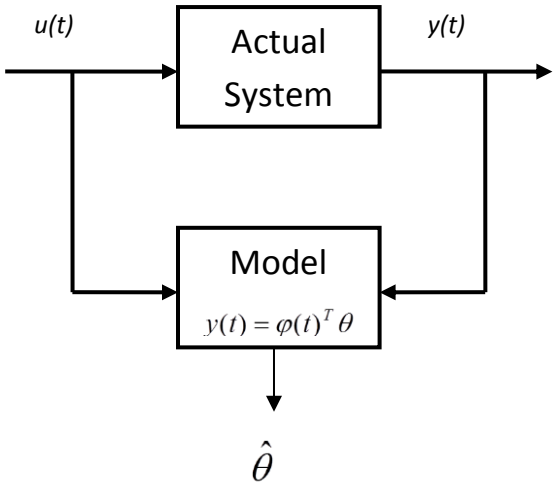
$$\hat{\theta} = PB \tag{8}$$

where $P = \left[ \sum_{t=1}^{N} (\varphi(t)\varphi^T(t)) \right]^{-1} = \left( \Phi \Phi^T \right)^{-1}$ (9)

$$B = \sum_{t=1}^{N} y(t)\varphi(t) \quad (10)$$

## 2.2 THE RECURSIVE LEAST-SQUARES ALGORITHM

While the above algorithm is appropriate for off-line, batch processing where the whole data are available, we often need to estimate parameters in real-time where data are assimilated in sequence.
A recursive algorithm for updating parameter estimation in each sampling period is a powerful tool for such real-time applications.




$\hat{\theta} = PB$ : Batch Processing

⇩

On-Line Estimation
Based on the latest data {*y(t)*, $\varphi(t)$ }
We update the estimate $\hat{\theta}$

Recursive Formula

- Simple enough to complete within a given sampling period
- No need to store the whole observed data

$\hat{\theta}(t) = P_t B_t$

Recall $\hat{\theta} = PB$, where

$$P = \left[ \sum_{t=1}^{N} (\varphi(t)\varphi^T(t)) \right]^{-1} = \left( \Phi \Phi^T \right)^{-1} \qquad P^{-1} = \left[ \sum_{t=1}^{N} (\varphi(t)\varphi^T(t)) \right] = \left( \Phi \Phi^T \right)$$

$$B = \sum_{t=1}^{N} y(t)\varphi(t)$$

A recursive computation algorithm can be obtained in the following three steps:

***Step 1*** Splitting $B_t$ and $P_t$
From (10)

$$B_t = \sum_{i=1}^{t} y(i)\varphi(i) = \sum_{i=1}^{t-1} y(i)\varphi(i) + y(t)\varphi(t)$$

$$B_t = B_{t-1} + y(t)\varphi(t) \tag{11}$$

From (9)

$$P_t^{-1} = \sum_{i=1}^{t} (\varphi(i)\varphi^T(i))$$

$$P_t^{-1} = P_{t-1}{}^{-1} + \varphi(t)\varphi^T(t) \tag{12}$$

***Step 2*** Apply the **Matrix Inversion Lemma** (An Intuitive Method)

Premultiplying $P_t$ and postmultiplying $P_{t-1}$ to (12) yield

$$P_t P_t^{-1} P_{t-1} = P_t P_{t-1}{}^{-1} P_{t-1} + P_t \varphi(t)\varphi^T(t) P_{t-1}$$

$$P_{t-1} = P_t + P_t \varphi(t)\varphi^T(t) P_{t-1} \tag{13}$$

Postmultiplying $\varphi(t)$

$$P_{t-1}\varphi(t) = P_t\varphi(t) + P_t\varphi(t)\varphi^T(t)P_{t-1}\varphi(t) = P_t\varphi(t)\left(1 + \varphi^T(t)P_{t-1}\varphi(t)\right)$$

Dividing both sides by $1+\phi^T(t)P_{t-1}\phi(t) > 0$ yields

$$P_t\varphi(t) = \frac{P_{t-1}\varphi(t)}{\left(1 + \varphi^T(t)P_{t-1}\varphi(t)\right)} \tag{13-a}$$

Postmultiplying $\varphi^T(t)P_{t-1}$

$$P_t\varphi(t)\varphi^T(t)P_{t-1} = \frac{P_{t-1}\varphi(t)\varphi^T(t)P_{t-1}}{\left(1 + \varphi^T(t)P_{t-1}\varphi(t)\right)}$$

(13) $\rightarrow$

$$P_{t-1} - P_t$$

Therefore,

$$P_t = P_{t-1} - \frac{P_{t-1}\varphi(t)\varphi^T(t)P_{t-1}}{\left(1 + \varphi^T(t)P_{t-1}\varphi(t)\right)} \tag{14}$$

Note that no matrix inversion is needed for updating $P_t$!
This is a special case of the Matrix Inversion Lemma.

$$\left[A + BCD\right]^{-1} = A^{-1} - A^{-1}B\left[DA^{-1}B + C^{-1}\right]^{-1}DA^{-1} \tag{15}$$

**Exercise 1** Prove (15) and use (15) to obtain (14) from (12)

***Step 3*** Reducing $\hat{\theta}(t) = P_t B_t$ (16)

to the following recursive form:

$$\hat{\theta}(t) = \hat{\theta}(t-1) + \mathrm{K}_t \left( \underbrace{y(t) - \phi^T(t)\hat{\theta}(t-1)}_{\text{Prediction Error}} \right) \tag{17}$$

A type of gain for correcting the error

From (16)

$$\hat{\theta}(t) = P_t B_t \ , \ \hat{\theta}(t-1) = P_{t-1} B_{t-1}$$

Therefore,

$$\begin{aligned}
\hat{\theta}(t) - \hat{\theta}(t-1) &= P_t B_t - P_{t-1} B_{t-1} \\
&= \left( P_{t-1} - \frac{P_{t-1}\phi(t)\phi^T(t)P_{t-1}}{\left(1+\phi^T(t)P_{t-1}\phi(t)\right)} \right) \left( B_{t-1} + y(t)\phi(t) \right) - P_{t-1}B_{t-1} \\
&= P_{t-1}y(t)\phi(t) - \frac{P_{t-1}\phi(t)\phi^T(t)P_{t-1}}{\left(1+\phi^T(t)P_{t-1}\phi(t)\right)} \left( B_{t-1} + y(t)\phi(t) \right) \\
&= \frac{P_{t-1}y(t)\phi(t)\left(1+\phi^T(t)P_{t-1}\phi(t)\right) - P_{t-1}\phi(t)\phi^T(t)P_{t-1}\phi(t)y(t) - P_{t-1}\phi(t)\phi^T(t)P_{t-1}B_{t-1}}{\left(1+\phi^T(t)P_{t-1}\phi(t)\right)} \\
&= \frac{P_{t-1}y(t)\phi(t) - P_{t-1}\phi(t)\phi^T(t)\hat{\theta}(t-1)}{\left(1+\phi^T(t)P_{t-1}\phi(t)\right)} \\
&= \frac{P_{t-1}\phi(t)}{\left(1+\phi^T(t)P_{t-1}\phi(t)\right)} \left[ y(t) - \phi^T(t)\hat{\theta}(t-1) \right]
\end{aligned}$$

Replacing this by $\mathrm{K}_t \in R^{m\times 1}$, we obtain (17)

**The Recursive Least Squares (RLS) Algorithm**

$$\hat{\theta}(t) = \hat{\theta}(t-1) + \frac{P_{t-1}\varphi(t)}{\left(1+\varphi^T(t)P_{t-1}\varphi(t)\right)} \left( y(t) - \varphi^T(t)\hat{\theta}(t-1) \right) \tag{18}$$

$$P_t = P_{t-1} - \frac{P_{t-1}\varphi(t)\varphi^T(t)P_{t-1}}{\left(1+\varphi^T(t)P_{t-1}\varphi(t)\right)} \quad t = 1,2,\ldots \text{ w/initial conditions} \tag{14}$$

with

$\hat{\theta}(0)$ : arbitrary

$P_o$: positive definite matrix

This Recursive Least Squares Algorithm was originally developed by
**Carl Friedrich Gauss** (1777 – 1855)

## 2.3 PHYSICAL MEANING OF MATRIX *P*

The Recursive Least Squares (RLS) algorithm updates parameter vector $\hat{\theta}(t-1)$ based on new data $\varphi^T(t),\ y(t)$ in such a way that the overall squared error may be minimal in each step. This is done by multiplying the prediction error $\varphi^T(t)\hat{\theta}(t-1)-y(t)$ with the gain matrix which contains matrix $P_{t-1}$. To better understand the RLS algorithm, let us examine the physical meaning of matrix $P_{t-1}$.

Recall the definition of matrix *P*: $P_t^{-1}=\sum_{i=1}^{t}\varphi(i)\varphi^T(i)=\Phi\,\Phi^T$ where $\Phi=\left[\phi(1)..\phi(t)\right]\in R^{m\times t}$.

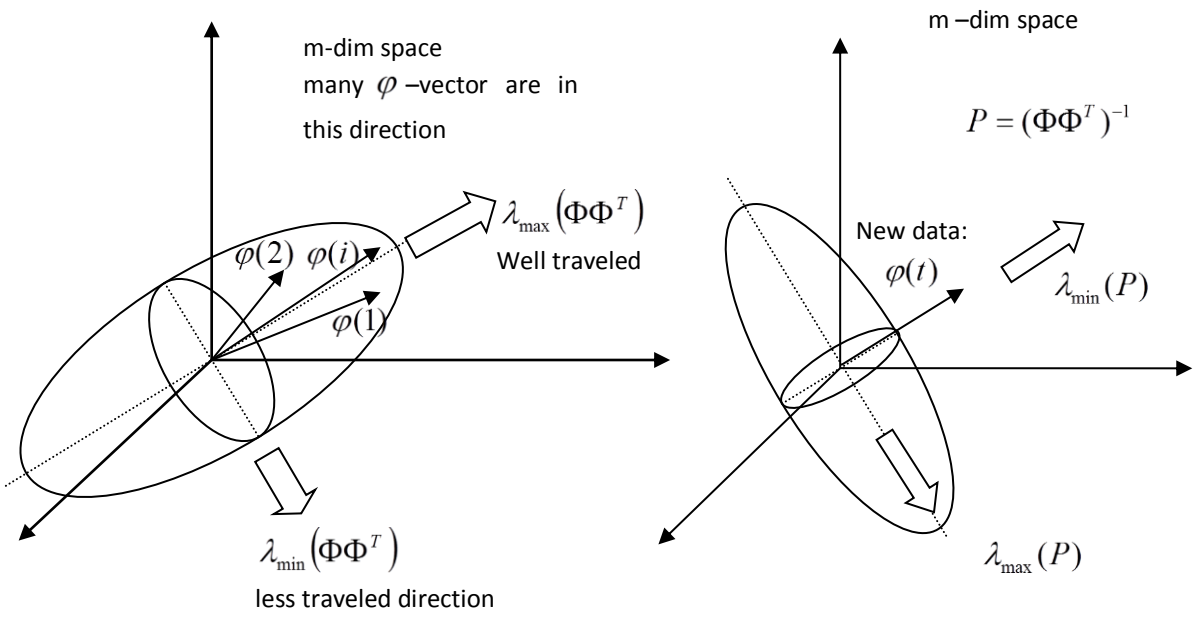
From (14) and (15) we can find that the parameter update gain $K_t$ in (17) can be written as:

$$K_t=P_t\,\varphi(t) \tag{19}$$

Therefore, the parameter update is scaled with the matrix $\boldsymbol{P}$.

**Exercise 2** Prove (19).

Note that matrix $\Phi\,\Phi^T$ as well as matrix P vary depending on how the set of regressor vectors $\{\varphi(i)\}$ span the *m*-dimensional space. See the left figure below.




Geometric Interpretation of matrix $P^{-1}$.

Since $\Phi\Phi^T \in R^{mxm}$ is a symmetric matrix of real numbers, it has all ***real*** eigenvalues. The eigen vectors associated with the individual eigenvalues are also real. Therefore, the matrix $\Phi\Phi^T$ can be reduced to a diagonal matrix using a coordinate transformation, i.e. using the eigen vectors as the bases.

$$\Phi\Phi^T \Rightarrow D = \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & & \vdots \\ \vdots & & \ddots & \\ 0 & \cdots & & \lambda_m \end{pmatrix} \in R^{mxm}$$

$$\lambda_{\max} = \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m = \lambda_{\min}$$

$$P = (\Phi\Phi^T)^{-1} \Rightarrow D^{-1} = \begin{pmatrix} 1/\lambda_1 & 0 & \cdots & 0 \\ 0 & 1/\lambda_2 & & \vdots \\ \vdots & & \ddots & \\ 0 & \cdots & & 1/\lambda_m \end{pmatrix} \in R^{mxm} \qquad (20)$$

The direction of $\lambda_{\max}(\Phi\Phi^T)$ = The direction of $\lambda_{\min}(P)$.

If $\lambda_{\min}(\Phi\Phi^T) = 0$, then $\det(\Phi\Phi^T) = 0$, and the ellipsoid collapses. This implies that there is no input data $\varphi(i)$ in the direction of $\lambda_{\min}$, i.e. the input data set does not contain any information in that direction. In consequence, the *m*-dimensional parameter vector $\theta$ cannot be fully determined with this dataset.

In the direction of $\lambda_{\max}(\Phi\Phi^T)$, there are plenty of input data: $\varphi(i)\cdots$. This direction has been well explored, well excited. Although new data are obtained, the correction to the parameter vector $\hat{\theta}(t-1)$ is small, if the new input data $\varphi(t)$ is in the same direction as that of $\lambda_{\max}(\Phi\Phi^T)$, i.e. the direction of $\lambda_{\min}(P)$. See the second figure above.

The above observations are summarized as follows:

1) Matrix ***P*** determins the gain of the prediction error feedback

$$\hat{\theta}(t) = \hat{\theta}(t-1) + \mathrm{K}_t e(t) \qquad (17)$$

where $K_t$ is the gain matrix proportional to $P_t$ : $\mathrm{K}_t = \dfrac{P_{t-1}\phi(t)}{\left(1+\phi^T(t)P_{t-1}\phi(t)\right)} = P_t\phi(t)$

(21)

2) If a new-data point $\varphi(t)$ is aligned with the direction of $\lambda_{\max}\left(\Phi\Phi^T\right)$ or $\lambda_{\min}\left(P_t\right)$,

then $\mathrm{K}_t = P_t\phi(t)$ is small. Therefore, the correction is small.

3) Matrix $\boldsymbol{P}_t$ represents how much we know about the *m*-dimensional data space. The more we already know, the less the error correction gain $\boldsymbol{K}_t$ becomes. Correction $\Delta\theta$ gets smaller and smaller as *t* tends infinity.

## 2.4 INITIAL CONDITIONS AND PROPERTIES OF RLS

In general, computation of a recursive formula entails initial conditions. There are two initial conditions for RLS: $\hat{\theta}(0)$ and $P_o$. There do not have to be close to their correct values, since they are recursively modified. But $P_o$ must be good enough to be able to use the RLS algorithm. As discussed below, we can set an arbitrary value for $\hat{\theta}(0)$, but

$P_o$ must be a positive definite matrix such as the identity matrix $I$.

Given a positive definite $P_o$ and an arbitrary $\hat{\theta}(0)$, the following theorem shows that the recursive algorithm is usable and computable. Furthermore, the following theorem explains how the recursive algorithm brings the parameter estimate $\hat{\theta}(t)$ from a given initial value to the one that minimizes the squared error[1]. Depending on the given initial values of $\hat{\theta}(0)$ and $P_o$ the (best)estimation process thereafter will be different. The following theorem shows how the initial conditions influence the recursive estimation process. The point is that

- We treat initial values of $\hat{\theta}(0)$ and $P_o$ as the best estimate derived from some preliminary data. As long as $P_o$ is positive definite, it is always possible to decompose it to $P_0^{-1} = \sum \varphi(\bullet)\varphi(\bullet)^T$. Here $\varphi(\bullet)$ are fictitious regressor vectors, but they exist for an arbitrary positive definite $P_o$. Assuming batch processing for this preliminary optimal estimate, we can write a given initial parameter vector as $\hat{\theta}(0) = P_0 B_0$, where $B_0 = \sum y(\bullet)\varphi(\bullet)$. We treat that $\hat{\theta}(0)$ and $P_o$ were determined from the fictitious dataset $\{\varphi(\bullet), y(\bullet)\}$.
- Then the resultant optimal estimate that RLS provides is the one that minimizes the following cost function.

Theorem

The Recursive Least Squares (RLS) algorithm minimizes the following cost function:

$$J_t(\theta) = \frac{1}{2}\sum_{i=1}^{t}\left(y(i) - \varphi^T(i)\theta\right)^2 + \frac{1}{2}\left(\theta - \hat{\theta}(0)\right)^T P_0^{-1}\left(\theta - \hat{\theta}(0)\right) \qquad (22)$$

where $P_o$ is an arbitrary positive definite matrix ($m$ by $m$) and $\hat{\theta}(0) \in R^m$ is arbitrary.

Proof Differentiating $J_t(\theta)$

$$\frac{dJ_t(\theta)}{d\theta} = 0 \quad \Longrightarrow \quad -\sum_{i=1}^{t}\left(y(i) - \varphi^T(i)\theta\right)\varphi(i) + P_0^{-1}\left(\theta - \hat{\theta}(0)\right) = 0 \qquad (23)$$

Collecting terms

$$\underbrace{\left[\sum_{i=1}^{t}\varphi(i)\varphi^T(i) + P_0^{-1}\right]}\theta = \sum_{i=1}^{t} y(i)\varphi(i) + P_0^{-1}\hat{\theta}(0)$$

[1] This does not mean the parameter estimate converges to the true parameter value. It depends on the "richness" of the regressor vectors. More complete convergence analysis will be addressed later in the system identification chapters.

$$P_t^{-1}$$

The parameter vector minimizing (22) is then given by

$$\hat{\theta}(t) = P_t\left[\sum_{i=1}^{t} y(i)\varphi(i) + P_0^{-1}\hat{\theta}(0)\right]$$

$$= P_t\left[y(t)\varphi(t) + \underbrace{\sum_{i=1}^{t-1} y(i)\varphi(i) + P_0^{-1}\hat{\theta}(0)}\right]$$

$$\Downarrow$$

$$\sum_{i=1}^{t-1} y(i)\varphi(i) + P_0^{-1}\hat{\theta}(0) = \sum_{i=1}^{t-1} y(i)\varphi(i) + B_0 \rightarrow B_{t-1} = P_{t-1}^{-1}\cdot\hat{\theta}(t-1) \tag{24}$$

Recall $P_t^{-1} = \varphi(t)\varphi^T(t) + P_{t-1}^{-1}$

$$\begin{aligned}\hat{\theta}(t) &= P_t\left[P_t^{-1}\hat{\theta}(t-1) + y(t)\varphi(t) - \varphi(t)\varphi^T(t)\hat{\theta}(t-1)\right]\\ &= \hat{\theta}(t-1) + P_t\varphi(t)\left[y(t) - \varphi^T(t)\hat{\theta}(t-1)\right]\end{aligned} \tag{25}$$

Postmultiplying $\varphi(t)$ to both sides of (14)

$$\begin{aligned}P_t\varphi(t) &= P_{t-1}\varphi(t) - \frac{P_{t-1}\varphi(t)\varphi^T(t)P_{t-1}\varphi(t)}{\left(1+\varphi^T(t)P_{t-1}\varphi(t)\right)}\\ &= \frac{P_{t-1}\varphi(t)}{\left(1+\varphi^T(t)P_{t-1}\varphi(t)\right)}\end{aligned} \tag{26}$$

using (26) in (25) yields (18)

$$\hat{\theta}(t) = \hat{\theta}(t-1) + \frac{P_{t-1}\varphi(t)}{\left(1+\varphi^T(t)P_{t-1}\varphi(t)\right)}\left(y(t) - \varphi^T(t)\hat{\theta}(t-1)\right) \tag{18}$$

Q.E.D.

**Discussion on the Theorem of RLS**

$$\hat{\theta}(t) = \arg\min_{\theta}\left[\underbrace{\frac{1}{2}\sum_{i=1}^{t}\left(y(i) - \varphi^T(i)\theta\right)^2}_{\text{Squared estimation error}\quad A} + \underbrace{\frac{1}{2}\left(\theta - \hat{\theta}(0)\right)^T P_0^{-1}\left(\theta - \hat{\theta}(0)\right)}_{\text{Weighted squared distance from } \hat{\theta}(0)\quad B}\right] \tag{27}$$

1) As $t$ gets larger, more data are obtained and term A gets overwhelmingly larger than term B. As a result, the influence of initial conditions fades out.
2) In an early stage, i.e. small time index *t*, $\theta$ is pulled towards $\hat{\theta}(0)$, particularly when the eigenvalues of matrix $P_0^{-1}$ are large.
3) In contrast, if the eigenvalues of $P_0^{-1}$ are small, $\theta$ tends to change more quickly in response to the prediction error, $y(t) - \varphi^T(t)\theta$.

4) The inverse initial matrix $P_0^{-1}$ represents the level of ***confidence*** for the initial parameter value $\hat{\theta}(0)$.

Note: Strictly speaking, the ***P*** matrix used in the above argument is different from the original definition, $P_t^{-1} = \sum_{i=1}^{t} \varphi(i)\varphi^T(i)$. It has been extended to

$$P_t^{-1} = \sum_{i=1}^{t} \varphi(i)\varphi^T(i) + P_o^{-1} \tag{28}$$

Other important properties of RLS include:

- Convergence of $\theta(t)$. It can be shown that

$$\lim_{t\to\infty} \left\| \hat{\theta}(t) - \hat{\theta}(t-1) \right\| = 0 \tag{29}$$

  See Goodwin and Sin's book, Ch.3, for proof.

- The change to the ***P*** matrix: $\Delta P = P_t - P_{t-1}$ is negative semi-definite, i.e.

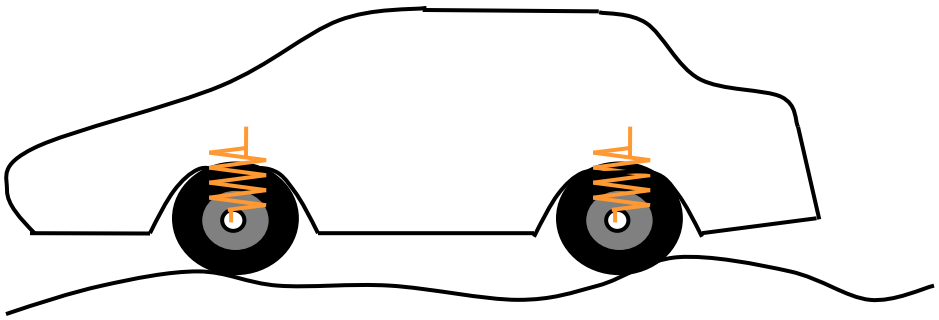
$$\Delta P = -\frac{P_{t-1}\varphi(t)\varphi^T(t)P_{t-1}}{\left(1+\varphi^T(t)P_{t-1}\varphi(t)\right)} \le 0 \tag{30}$$

  for an arbitrary $\varphi(t) \in R^m$ and positive definite $P_{t-1}$.

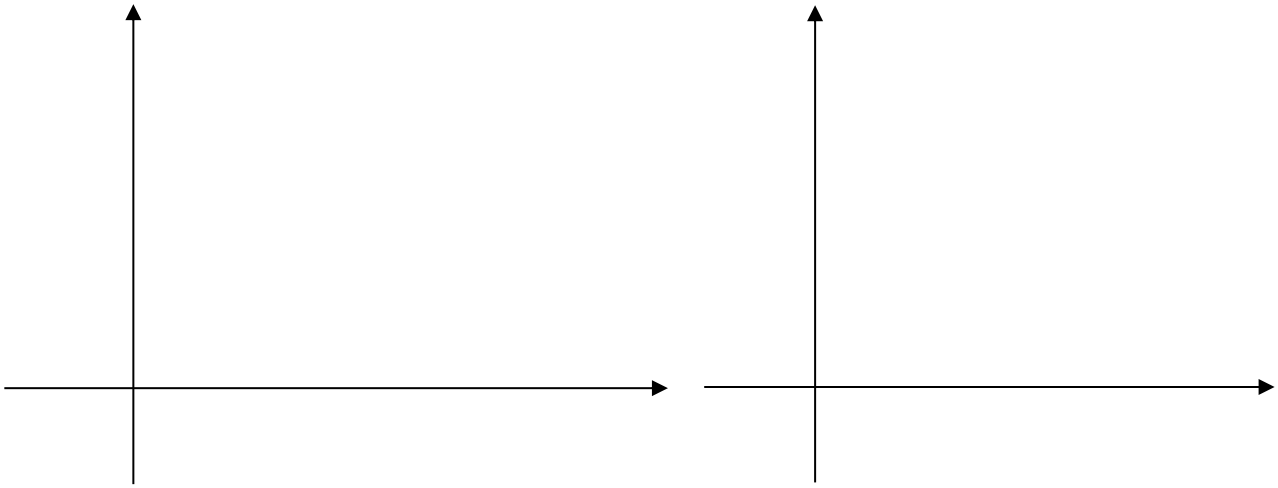
***Exercise 3*** Prove this property, (30).

## 2.5 Estimation of Time-varying Parameters

So far we have discussed the estimation of parameters that are assumed to be constant: $\theta =$ constant. In many important applications, including indirect adaptive control, parameters tend to vary over time. Consider a smart traction control system for an automobile. It monitors the road conditions, e.g. roughness, wetness, rain, snow, or ice, and adapts the traction control parameters accordingly. Clearly these parameter values change over time.



In estimating such varying parameters, old data that were observed long time ago are useless. More recent past data are more important, which must be weighted heavily than those older data.

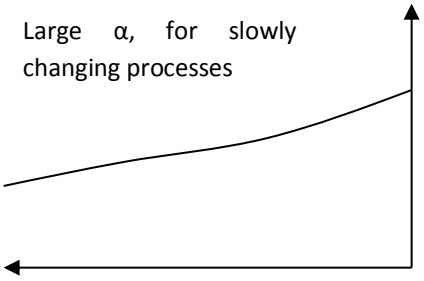


**Least Squares with Exponential Data Weighting**

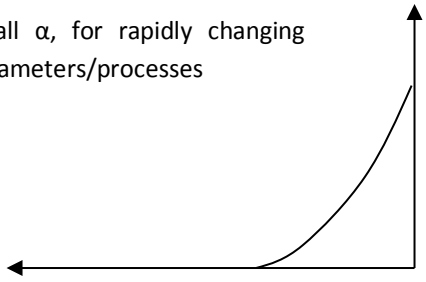
Forgetting factor: $\boldsymbol{\alpha}$

$$0 < \alpha \leq 1 \tag{31}$$

Large α, for slowly changing processes



Small α, for rapidly changing parameters/processes



Weighted Squared Error

$$J_t(\theta) = \sum_{i=1}^{t} \alpha^{t-i} e^2(i) \tag{32}$$

(33)

$$\hat{\theta}(t) = \arg\min_{\theta} J_t(\theta)$$

$\hat{\theta}(t)$ is given by the following recursive algorithm.

$$\hat{\theta}(t) = \hat{\theta}(t-1) + \frac{P_{t-1}\varphi(t)}{\left(\alpha + \varphi^T(t)P_{t-1}\varphi(t)\right)}\left(y(t) - \varphi^T(t)\hat{\theta}(t-1)\right) \tag{34}$$

$$P_t = \frac{1}{\alpha}\left[P_{t-1} - \frac{P_{t-1}\varphi(t)\varphi^T(t)P_{t-1}}{\left(\alpha + \varphi^T(t)P_{t-1}\varphi(t)\right)}\right] \tag{35}$$

***Exercise***: Obtain (34) and (35) from (32) and (33).

A drawback of the forgetting factor approach

When the system under consideration enters "steady state", the matrix $P_{t-1}\varphi(t)\varphi^T(t)P_{t-1}$ tends to the null matrix. This implies

$$P_t \approx \frac{1}{\alpha}P_{t-1} \tag{36}$$

As α<1, 1/ α makes $P_t$ larger than $P_{t-1}$. Therefore { $P_t$ } begins to increase exponentially.
The "Blow-Up" problem

## Remedy:

Covariance Re-setting Approach

- The forgetting factor approach has the "Blow-Up" problem
- In the ordinary RLS the P matrix gets small after some iteration (typically 10-20 iterations). Then the gain dramatically reduces, and $\widehat{\theta}$ is no longer updated.

  The Covariance Re-Setting method is to solve these shortcomings by occasionally re-setting the P matrix to:

  $$P_{t^*} = kI \qquad 0 < k < \infty \tag{37}$$

  This re-vitalizes the algorithm.

### 2.6 ORTHOGONAL PROJECTION

The RLS algorithm provides an iterative procedure to converge to its final parameter value. This may take more than *m*-steps, dimension of $\theta$. The Orthogonal Projection algorithm provides the least squares solution exactly in *m* recursive steps

Assume $\Phi_m = [\varphi(1) \quad \varphi(2) \quad \dots \quad \varphi(m)]$ (38)

Spanning the whole *m*-dim space
Set $P_0 = I$ (the *m* x *m* identity matrix) and $\hat{\theta}(0)$ arbitrary
Compute

$$\hat{\theta}(t) = \hat{\theta}(t-1) + \frac{P_{t-1}\varphi(t)}{\varphi^T(t)P_{t-1}\varphi(t)}\left(y(t) - \varphi^T(t)\hat{\theta}(t-1)\right) \tag{39}$$

where matrix $P_{t-1}$ is updated with the following recursive formula.

$$P_t = P_{t-1} - \frac{P_{t-1}\varphi(t)\varphi^T(t)P_{t-1}}{\varphi^T(t)P_{t-1}\varphi(t)}$$

Note that +1 involved in the denominator of RLS is eliminated. This causes a numerical problem when $\varphi^T(t)P_{t-1}\varphi(t)$ is small, the gain is large.

This orthogonal projection algorithm is more efficient, but is very sensitive to noisy data. Ill-conditioned when $\varphi^T(t)P_{t-1}\varphi(t) \approx 0$. RLS is more robust.

## 2.7 MULTI-OUTPUT, WEIGHTED LEAST SQUARES ESTIMATION


$l$-output $\vec{y}(t) = \begin{pmatrix} y_1 \\ \vdots \\ y_l \end{pmatrix} \in R^l$

For each output $\hat{y}_i(t) = \varphi_i^T(t)\theta$

$$\hat{\vec{y}}(t) = \begin{pmatrix} \varphi_1^T \\ \vdots \\ \varphi_\ell^T \end{pmatrix}\theta = \Psi^T(t)\theta \qquad \Psi \in R^{m\times l},\ \theta \in R^{m\times 1} \tag{40}$$

Error $\vec{e}(t) = \begin{pmatrix} e_1 \\ \vdots \\ e_\ell \end{pmatrix} = \vec{y}(t) - \Psi^T(t)\theta$ (41)

Consider that each squared error is weighted differently, or
Weighted Multi-Output Squared Error:

$$J_t(\theta) = \sum_{i=1}^{t} \vec{e}^T(i)W\vec{e}(i) = \sum_{i=1}^{t} (\vec{y}(i) - \Psi^T\theta)^T W(\vec{y}(i) - \Psi^T\theta) \tag{42}$$

$$\hat{\theta}(t) = \arg\min_{\theta} J_t(\theta) \qquad \Longrightarrow \hat{\theta}(t) = \mathrm{P}_t\mathrm{B}_t$$

$$\mathrm{P}_t = \left[\sum_{i=1}^{t} \Psi(i)W\Psi(i)^T\right]^{-1} \in R^{m\times m}$$

$$\mathrm{B}_t = \sum_{i=1}^{t} \Psi^T(i)W\vec{y}(i) \tag{43}$$

The recursive algorithm

$$\hat{\theta}(t) = \hat{\theta}(t-1) + \mathrm{P}_t\Psi(t)W\left(\vec{y}(t) - \Psi^T(t)\hat{\theta}(t-1)\right) \tag{44}$$

$$\mathrm{P}_t = \mathrm{P}_{t-1} - \mathrm{P}_{t-1}\Psi(t)\left[W^{-1} + \Psi^T(t)\mathrm{P}_{t-1}\Psi(t)\right]^{-1}\Psi^T(t)\mathrm{P}_{t-1}$$