2. Parameter Estimation for Deterministic Systems

2.1 Least Squares Estimation

\[ y = b_1 u_1 + b_2 u_2 + \ldots + b_m u_m \]

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

Observations:
\[
\begin{align*}
\phi &= [u_1 \ldots u_m]^T \in R^m \\
y &= \phi^T \theta
\end{align*}
\]

The problem is to find the parameters \( \theta = [b_1 \ldots b_m]^T \) from observation data:
\[
\begin{bmatrix}
\phi(1), y(1) \\
\phi(2), y(2) \\
\vdots \\
\phi(N), y(N)
\end{bmatrix} \rightarrow \theta
\]

The system may be
- a linear dynamic system, e.g. \( y(t) = b_1 u(1-t) + b_2 u(2-t) + \ldots + b_m u(t-m) \)
  \[ \phi(t) = [u(t-1), u(t-2), \ldots, u(t-m)]^T \in R^m \]
- or a nonlinear dynamic system, e.g. \( y(t) = b_1 u(t-1) + b_2 u(t-2)u(t-1) \)
  \[ \phi(t) = [u(t-1), u(t-2)u(t-1)]^T \]

Note that the parameters, \( b_1, b_2 \), are **linearly** involved in the input-output equation.

Using an estimated parameter vector \( \hat{\theta} \), we can write a predictor that predicts the output from inputs:
\[
\hat{y}(t|\theta) = \phi(t)^T \hat{\theta}
\]
We evaluate the predictor’s performance by the squared error given by

\[ V_N(\theta) = \frac{1}{N} \sum_{t=1}^{N} (\hat{y}(t | \theta) - y(t))^2 \] (3)

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

\[ \hat{\theta} = \arg \min_{\theta} V_N(\theta) \] (4)

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

\[ \frac{dV_N(\theta)}{d\theta} = 0 \quad \Rightarrow \quad \frac{2}{N} \sum_{t=1}^{N} (\phi^T(t)\theta - y(t)) \phi(t) = 0 \] (5)

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

Consider \( m \times N \) matrix

\[ \Phi = [\phi(1) \; \phi(2) \; \ldots \; \phi(N)] \] (7)

If vectors \( \phi(1) \; \phi(2) \; \ldots \; \phi(N) \) span the whole \( m \)-dimensional vector space

\[ \text{rank} \; \Phi = m \; ; \; \text{full rank} \]

If there are \( m \) linearly independent (column) vectors in this matrix \( \Phi \),

\[ \text{rank} \Phi = m \; ; \; \text{full rank} \]

\[ \text{rank} \Phi = \text{rank} \Phi \Phi^T = m \; ; \; \text{full rank, hence invertible} \]

Under this condition, the optimal parameter vector is given by

\[ \hat{\theta} = PB \] (8)

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

\[ B = \sum_{t=1}^{N} y(t) \phi(t) \] (10)

### 2.2 The Recursive Least-Squares Algorithm

While the above algorithm is for batch processing of whole data, we often need to estimate parameters in real-time where data are coming from a dynamical system.
A recursive algorithm for computing the parameters is a powerful tool in such applications.

\[ \hat{\theta}(t) = P_t B_t \]

\[ \hat{\theta} = PB \]

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

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

Three steps for obtaining a recursive computation algorithm

a) 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 (11)

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

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

b) The Matrix Inversion Lemma (An Intuitive Method)

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

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

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

Postmultiplying \( \varphi(t) \)

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

\[ P_t \varphi(t) = \frac{P_{t-1} \varphi(t)}{1 + \varphi^T(t) P_{t-1} \varphi(t)} \]

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

\[ P_t \varphi(t)\varphi^T(t) P_{t-1} = \frac{P_{t-1} \varphi(t)\varphi^T(t) P_{t-1} \varphi(t)}{1 + \varphi^T(t) P_{t-1} \varphi(t)} \]

\[ P_{t-1} = P_t \]

Therefore,

\[ P_t = P_{t-1} = \frac{P_{t-1} \varphi(t)\varphi^T(t) P_{t-1} \varphi(t)}{1 + \varphi^T(t) P_{t-1} \varphi(t)} \tag{14} \]

Note that no matrix inversion is needed for updating \( P_t \! \)!

This is a special case of the Matrix Inversion Lemma.

\[ [A + BCD]^{-1} = A^{-1} - A^{-1} B \left[ D A^{-1} B + C^{-1} \right]^{-1} D A^{-1} \tag{15} \]
Exercise 1 Prove (15) and use (15) to obtain (14) from (12)

c) Reducing \( \hat{\theta}(t) = P_t B_t \) to the following recursive form:

\[
\hat{\theta}(t) = \hat{\theta}(t-1) + K_t \frac{y(t) - \varphi^T(t) \hat{\theta}(t-1)}{\text{Prediction Error}}
\]  \hspace{1cm} (17)

A type of gain for correcting the error

From (16) \( \hat{\theta}(t) = P_t B_t \) \( \hat{\theta}(t-1) = P_{t-1} B_{t-1} \)

\[
\hat{\theta}(t) - \hat{\theta}(t-1) = P_t B_t - P_{t-1} B_{t-1}
\]

\[
\hat{\theta}(t) - \hat{\theta}(t-1) = P_{t-1} y(t) \varphi(t) (1 + \varphi^T(t) P_{t-1} \varphi(t)) - P_{t-1} B_{t-1}
\]

\[
\hat{\theta}(t) - \hat{\theta}(t-1) = \frac{P_{t-1} y(t) \varphi(t) (1 + \varphi^T(t) P_{t-1} \varphi(t))}{1 + \varphi^T(t) P_{t-1} \varphi(t)}
\]

\[
\hat{\theta}(t) - \hat{\theta}(t-1) = \frac{P_{t-1} y(t) \varphi(t)}{1 + \varphi^T(t) P_{t-1} \varphi(t)}
\]

Replacing this by \( K_t \in R^{m \times l} \), we obtain (17)

The Recursive Least Squares (RLS) Algorithm

\[
\hat{\theta}(t) = \hat{\theta}(t-1) + \frac{P_{t-1} \varphi(t)}{1 + \varphi^T(t) P_{t-1} \varphi(t)} (y(t) - \varphi^T(t) \hat{\theta}(t-1))
\]  \hspace{1cm} (18)

\[
P_t = P_{t-1} - \frac{P_{t-1} \varphi(t) \varphi^T(t) P_{t-1}}{1 + \varphi^T(t) P_{t-1} \varphi(t)} \quad t=1,2,\ldots \text{ w/ initial conditions}
\]  \hspace{1cm} (14)

with

\[
\hat{\theta}(0): \text{ arbitrary}
\]

\( P_0: \) positive definite matrix

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