Mathworks:
MATLAB toolboxes > Fuzzy Logic
(do the tutorials)

LSE (least squares estimator) → used to optimize the linear parameters of a system

$$\vec{\theta} = \{\theta_1, \theta_2, \theta_3, \dots, \theta_n\}^T$$
$$\vec{u}_i = \{x_1, x_2, x_3, \dots, x_p\}^T$$

$$m - \text{training data pairs}$$

$$\{\vec{u}_1, y_1\}, \{\vec{u}_2, y_2\}, \dots, \{\vec{u}_m, y_m\}$$
$$i = 1, 2, 3, \dots, m$$

$$\underline{A}\vec{\theta} = \vec{y}$$
$$\vec{\theta} = \left(\underline{A}^T \underline{A}\right)^{-1} \underline{A}^T \vec{y}$$

This is **offline training** (speed of operation is not a primary concern)
- you use all the training data pairs at once.

Recursive, or **online training**, is when training data pairs are used one after the other, or one at a time.

4.2 Recursive Lease Squares Estimator (LSE)

Suppose m −training data pairs.
k^{th} training data pair
k^{th} training operation

$$0 \leq k \leq m - 1$$
$$(\text{In MATLAB: } 1 \leq k \leq m)$$

Corresponding to the k^{th} training data pair: $1, 2, \dots, k$

$$\begin{array}{cccc} A & \vec{\theta} & & \vec{y} \\ \begin{bmatrix} f_1(\vec{u}_1) & f_2(\vec{u}_1) & \cdots & f_n(\vec{u}_1) \\ f_1(\vec{u}_2) & f_2(\vec{u}_2) & \cdots & f_n(\vec{u}_2) \\ \vdots & \vdots & \ddots & \vdots \\ f_1(\vec{u}_k) & f_2(\vec{u}_k) & \cdots & f_n(\vec{u}_k) \\ f_1(\vec{u}_{k+1}) & f_2(\vec{u}_{k+1}) & \cdots & f_n(\vec{u}_{k+1}) \end{bmatrix} & \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} & = & \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \\ y_{k+1} \end{bmatrix} \\ \vec{a}^T_{k+1} & \theta_k & & \end{array}$$

$$\underline{A}\vec{\theta}_k = \vec{y}$$
$$\vec{\theta}_k = \underbrace{\left(\underline{A}^T \underline{A}\right)^{-1}}_{P_k} \underline{A}^T \vec{y}$$

If $(k+1)^{th}$ training data pair is available:

$$\{\vec{u}_{k+1}, y_{k+1}\}$$

Will do $(k+1)^{th}$ update operation:

$$\begin{bmatrix} \underline{A} \\ \vec{a}^T_{k+1} \end{bmatrix} \vec{\theta}_{k+1} = \begin{bmatrix} \vec{y} \\ y_{k+1} \end{bmatrix}$$

$$\vec{\theta}_{k+1} = \underbrace{\left[\begin{bmatrix} \underline{A} \\ \vec{a}_{k+1}^T \end{bmatrix}^T \begin{bmatrix} \underline{A} \\ \vec{a}_{k+1}^T \end{bmatrix}\right]^{-1}}_{P_{k+1}} \begin{bmatrix} \underline{A} \\ \vec{a}_{k+1}^T \end{bmatrix}^T \begin{bmatrix} \vec{y} \\ y_{k+1} \end{bmatrix}$$

$$\vec{\theta}_{k+1} \sim \vec{\theta}_k + \text{update (modification)}$$

Introduce:

$$\underline{P}_k = \left(\underline{A}^T \underline{A}\right)^{-1}$$

$$\underline{P}_k^{-1} = \underline{A}^T \underline{A}$$

$$\underline{P}_{k+1} = \left[\begin{bmatrix} \underline{A} \\ \vec{a}_{k+1}^T \end{bmatrix}^T \begin{bmatrix} \underline{A} \\ \vec{a}_{k+1}^T \end{bmatrix}\right]^{-1} \tag{1}$$

$$= \left[\begin{bmatrix} \underline{A}^T & \vec{a}_{k+1} \end{bmatrix} \begin{bmatrix} \underline{A} \\ \vec{a}_{k+1}^T \end{bmatrix}\right]^{-1}$$

$$\underline{P}_{k+1} = \left[\underline{A}^T \underline{A} + \vec{a}_{k+1}^T \vec{a}_{k+1}\right]^{-1} \tag{2}$$

$$\underline{P}_{k+1}^{-1} = \underbrace{\underline{A}^T \underline{A}}_{P_k^{-1}} + \vec{a}_{k+1} \vec{a}_{k+1}^T$$

$$\underline{P}_{k+1}^{-1} = \underline{P}_k^{-1} + \vec{a}_{k+1} \vec{a}_{k+1}^T \tag{3}$$

$$\vec{\theta}_k = \left(\underline{A}^T \underline{A}\right)^{-1} \underline{A}^T \vec{y}$$

$$\vec{\theta}_k = \underline{P}_k \underline{A}^T \vec{y} \tag{4}$$

$$\vec{\theta}_{k+1} = \underline{P}_{k+1} \begin{bmatrix} \underline{A}^T & \vec{a}_{k+1} \end{bmatrix} \begin{bmatrix} \vec{y} \\ y_{k+1} \end{bmatrix}$$

$$\vec{\theta}_{k+1} = \begin{bmatrix} \underline{A}^T \vec{y} & \vec{a}_{k+1} y_{k+1} \end{bmatrix} \tag{5}$$

From Eq. (4):

$$\underline{P}_k^{-1} \vec{\theta}_k = \underbrace{\underline{P}_k^{-1} \underline{P}_k}_{I} \underline{A}^T \vec{y}$$

$$\underline{A}^T \vec{y} = \underline{P}_k^{-1} \vec{\theta}_k$$

Eq. (5) becomes:

$$\vec{\theta}_{k+1} = \underline{P}_{k+1} \left[\underline{P}_k^{-1} \vec{\theta}_k \quad \vec{a}_{k+1}^T y_{k+1}\right]$$

From Eq. (3):

$$\underline{P}_k^{-1} = \underline{P}_{k+1}^{-1} - \vec{a}_{k+1}^T \vec{a}_{k+1}$$

Eq. (5) becomes:

$$\vec{\theta}_{k+1} = \underline{P}_{k+1} \left[\left(\underline{P}_{k+1}^{-1} - \vec{a}_{k+1} \vec{a}_{k+1}^T\right) \vec{\theta}_k \quad \vec{a}_{k+1} y_{k+1}\right]$$

$$= [\mathrm{I} - \underline{P}_{k+1}\vec{a}_{k+1}\vec{a}_{k+1}^{T}]\vec{\theta}_k + \underline{P}_{k+1}\vec{a}_{k+1}y_{k+1}$$
$$= \vec{\theta}_k - \underline{P}_{k+1}\vec{a}_{k+1}\vec{a}_{k+1}^{T}\vec{\theta}_k + \underline{P}_{k+1}\vec{a}_{k+1}y_{k+1}$$
$$\vec{\theta}_{k+1} = \vec{\theta}_k + \underline{P}_{k+1}\vec{a}_{k+1}(y_{k+1} - \vec{a}_{k+1}^{T}\vec{\theta}_k) \longleftarrow 6$$

From Eq. (3):

$$\underline{P}_{k+1}^{-1} = \underline{P}_k^{-1} + \vec{a}_{k+1}\vec{a}_{k+1}^{T}$$
$$\underline{P}_{k+1} = [\underbrace{\underline{P}_k^{-1}}_{A} + \underbrace{\vec{a}_{k+1}}_{B}\underbrace{\vec{a}_{k+1}^{T}}_{C}]^{-1}$$

Formula:

$$(A + BC)^{-1}$$
$$= A^{-1} - A^{-1}B(I + CA^{-1}B)^{-1}CA^{-1}$$

$$A = \underline{P}_k^{-1}$$
$$B = \vec{a}_{k+1}$$
$$C = \vec{a}_{k+1}^{T}$$

$$\underline{P}_{k+1} = \underline{P}_k - \underline{P}_k\,\vec{a}_{k+1}\left(I + \vec{a}_{k+1}^{T}\underline{P}_k\,\vec{a}_{k+1}\right)^{-1}\vec{a}_{k+1}^{T}\underline{P}_k$$

$$\underline{P}_{k+1} = \underline{P}_k - \frac{\underline{P}_k\,\vec{a}_{k+1}\vec{a}_{k+1}^{T}\underline{P}_k}{I + \vec{a}_{k+1}^{T}\underline{P}_k\,\vec{a}_{k+1}} \longleftarrow 7$$

Use Eq. (6) and Eq. (7) to do recursive LSE and update $\vec{\theta}_{k+1}$

Initialization:

$$\underline{P}_0 = \alpha I$$

Where α is a larger number (1000, 10000, etc.).

From this, you can generate: → To be used in project

$$(\vec{\theta}_0 \ldots \vec{\theta}_1 \ldots \vec{\theta}_2 \ldots)$$

4.3 Gradient Algorithms

Non-linear parameter optimization method.

For linear parameters, LSE is the general method – not many methods are required.

Compared to linear parameter optimization, there are many optimization methods for non-linear systems, but this one is the basic one (most general).

$$\vec{\theta} = \left[\vec{\theta}_1, \vec{\theta}_2, \dots, \vec{\theta}_n\right]^T$$

Objective function (error function):

$$E(\vec{\theta})$$

We want to minimize this function.

But, θ can have many values, and it's possible that different numbers produce the same value:

Consider:

$2x_1^2 + x_2$p

When $x_1 = 1, x_2 = 1, E(\theta) = 3$
When $x_1 = 2$, $x_2 = -5$, $E(\theta) = 3$

These points would be on the same 'error height'