Chapter 7:
Kalman Filter

- State-space formulation
- Kalman filtering algorithm
- Examples of Kalman filtering
  - Scalar random variable estimation
  - Position estimation
  - System Identification
  - System tracking.

References

[1] Simon Haykin, *Adaptive Filter Theory*, Prentice Hall, 1996 (3rd Ed.), 2001 (4th Ed.)..

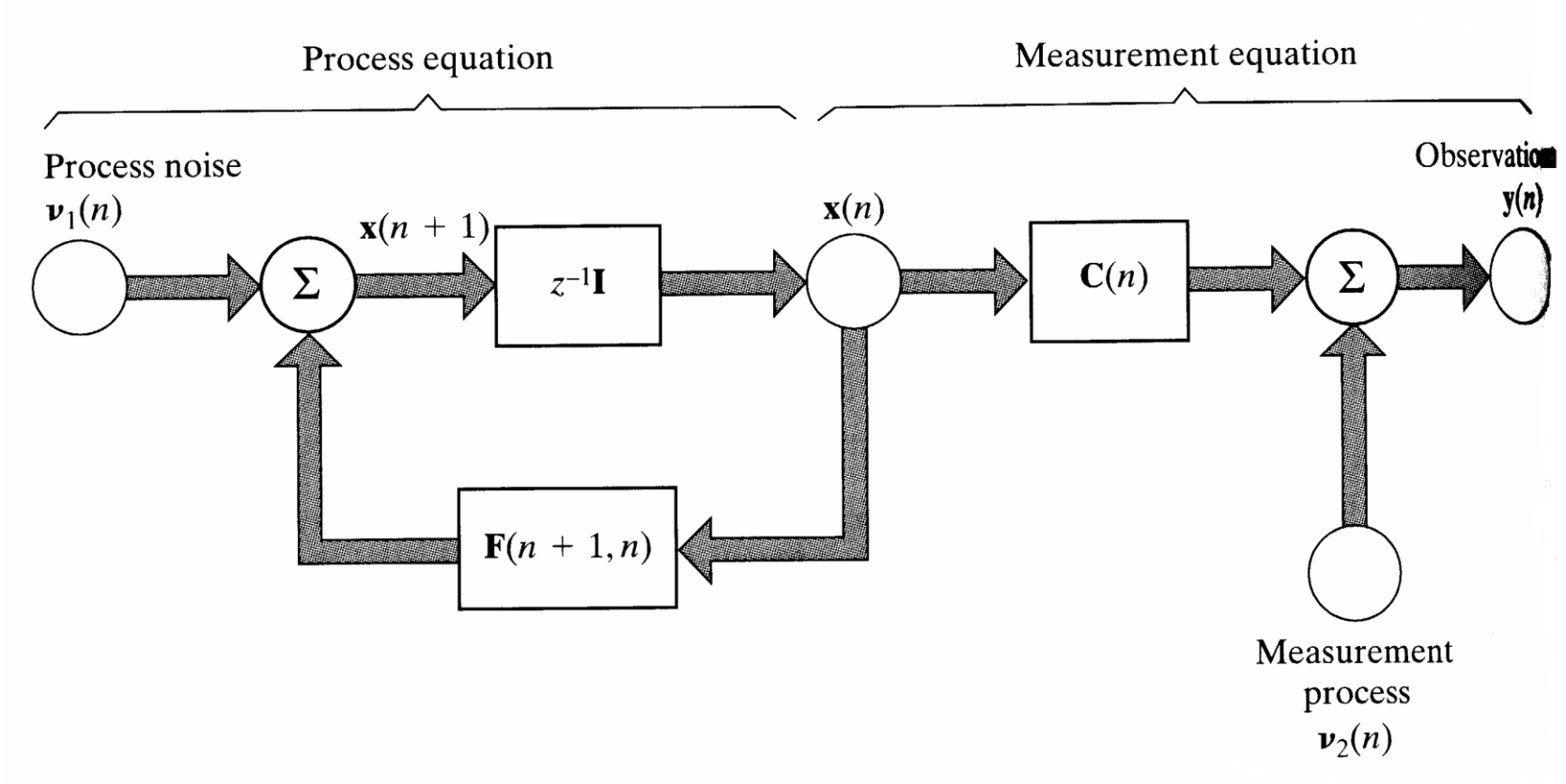
[2] Steven M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*, Prentice Hall, 1993.

[3] Alan V. Oppenheim, Ronald W. Schafer, *Discrete-Time Signal Processing*, Prentice Hall, 1989.

[4] Athanasios Papoulis, *Probability, Random Variables, and Stochastic Processes*, McGraw-Hill, 1991 (3rd Ed.), 2001 (4th Ed.).

7. Kalman Filter: State-Space Formulation (1)

- Linear discrete-time dynamical system: **State vector** $\mathbf{x}(n)$ with dimension M is unknown. To estimate $\mathbf{x}(n)$, using the **observation vector** $\mathbf{y}(n)$ with dimension N.



7. Kalman Filter: State-Space Formulation (2)

- **Process equation:**

$$\mathbf{x}(n+1) = \mathbf{F}(n+1, n)\mathbf{x}(n) + \mathbf{v}_1(n)$$

where $\mathbf{F}(n+1,n)$ is $(M\times M)$-*state transition matrix* relating the state of the system at times $n+1$ and n. $(M\times 1)$-*process noise* vector $\mathbf{v}_1(n)$ is zero-mean white-noise process with correlation matrix:

$$E\left[\mathbf{v}_1(n)\mathbf{v}_1^H(k)\right] = \begin{cases} \mathbf{Q}_1(n), & n = k \\ \mathbf{0}, & n \neq k \end{cases}$$

- **Measurement equation** describing observation vector:

$$\mathbf{y}(n) = \mathbf{C}(n)\mathbf{x}(n) + \mathbf{v}_2(n)$$

where $\mathbf{C}(n)$ is $(N\times M)$-*measurement matrix*. $(N\times 1)$-*measurement noise* vector $\mathbf{v}_2(n)$ is zero-mean white- noise process with correlation matrix:

$$E\left[\mathbf{v}_2(n)\mathbf{v}_2^H(k)\right] = \begin{cases} \mathbf{Q}_2(n), & n = k \\ \mathbf{0}, & n \neq k \end{cases}$$

7. Kalman Filter: State-Space Formulation (3)

- Assumed that the initial state $\mathbf{x}(0)$ is uncorrelated to $\mathbf{v}_1(n)$ and $\mathbf{v}_2(n)$ for $n \geq 0$ and $\mathbf{v}_1(n)$ and $\mathbf{v}_2(n)$ are statistically independent.

- **Statement of Kalman filterring:** Using entire observed data $\mathbf{y}(1)$, $\mathbf{y}(2)$,..., $\mathbf{y}(n)$, to find for each $n \geq 1$ the minimum mean-squared estimates of the components of the state $\mathbf{x}(i)$. If
  - $i=n$: Filtering problem
  - $i>n$: Predicting problem
  - $n>i \geq 1$: Smoothing problem

7. Kalman Filter: Algorithm (1)

Input vector process

Observations $= \{\mathbf{y}(1), \mathbf{y}(2), \ldots, \mathbf{y}(n)\}$

Known parameters

Transition matrix $= \mathbf{F}(n+1, n)$

Measurement matrix $= \mathbf{C}(n)$

Correlation matrix of process noise $= \mathbf{Q}_1(n)$

Correlation matrix of measurement noise $= \mathbf{Q}_2(n)$

Computation: $n = 1, 2, 3, \ldots$

$$\mathbf{G}(n) = \mathbf{F}(n+1, n)\mathbf{K}(n, n-1)\mathbf{C}^H(n)\left[\mathbf{C}(n)\mathbf{K}(n, n-1)\mathbf{C}^H(n) + \mathbf{Q}_2(n)\right]^{-1}$$

$$\boldsymbol{\alpha}(n) = \mathbf{y}(n) - \mathbf{C}(n)\hat{\mathbf{x}}(n \mid \mathcal{Y}_{n-1})$$

$$\hat{\mathbf{x}}(n+1 \mid \mathcal{Y}_n) = \mathbf{F}(n+1, n)\hat{\mathbf{x}}(n \mid \mathcal{Y}_{n-1}) + \mathbf{G}(n)\boldsymbol{\alpha}(n)$$

$$\mathbf{K}(n) = \mathbf{K}(n, n-1) - \mathbf{F}(n, n+1)\mathbf{G}(n)\mathbf{C}(n)\mathbf{K}(n, n-1)$$

$$\mathbf{K}(n+1, n) = \mathbf{F}(n+1, n)\mathbf{K}(n)\mathbf{F}^H(n+1, n) + \mathbf{Q}_1(n)$$

Initial conditions:

$$\hat{\mathbf{x}}(1 \mid \mathcal{Y}_0) = E[\mathbf{x}(1)]$$

$$\mathbf{K}(1, 0) = E\left[(\mathbf{x}(1) - E[\mathbf{x}(1)])(\mathbf{x}(1) - E[\mathbf{x}(1)])^H\right] = \Pi_0$$

Detail derivation in [1]

7. Kalman Filter: Algorithm (2)

where $\boldsymbol{\alpha}(n)$ is **innovation process** and $\mathbf{G}(n)$ is **Kalman gain**. $\mathbf{K}(n+1,n)$ is the **predicted state-error correlation matrix**:

$$\mathbf{K}(n+1,n) = E\left[\boldsymbol{\varepsilon}(n+1,n)\boldsymbol{\varepsilon}^{H}(n+1,n)\right]$$

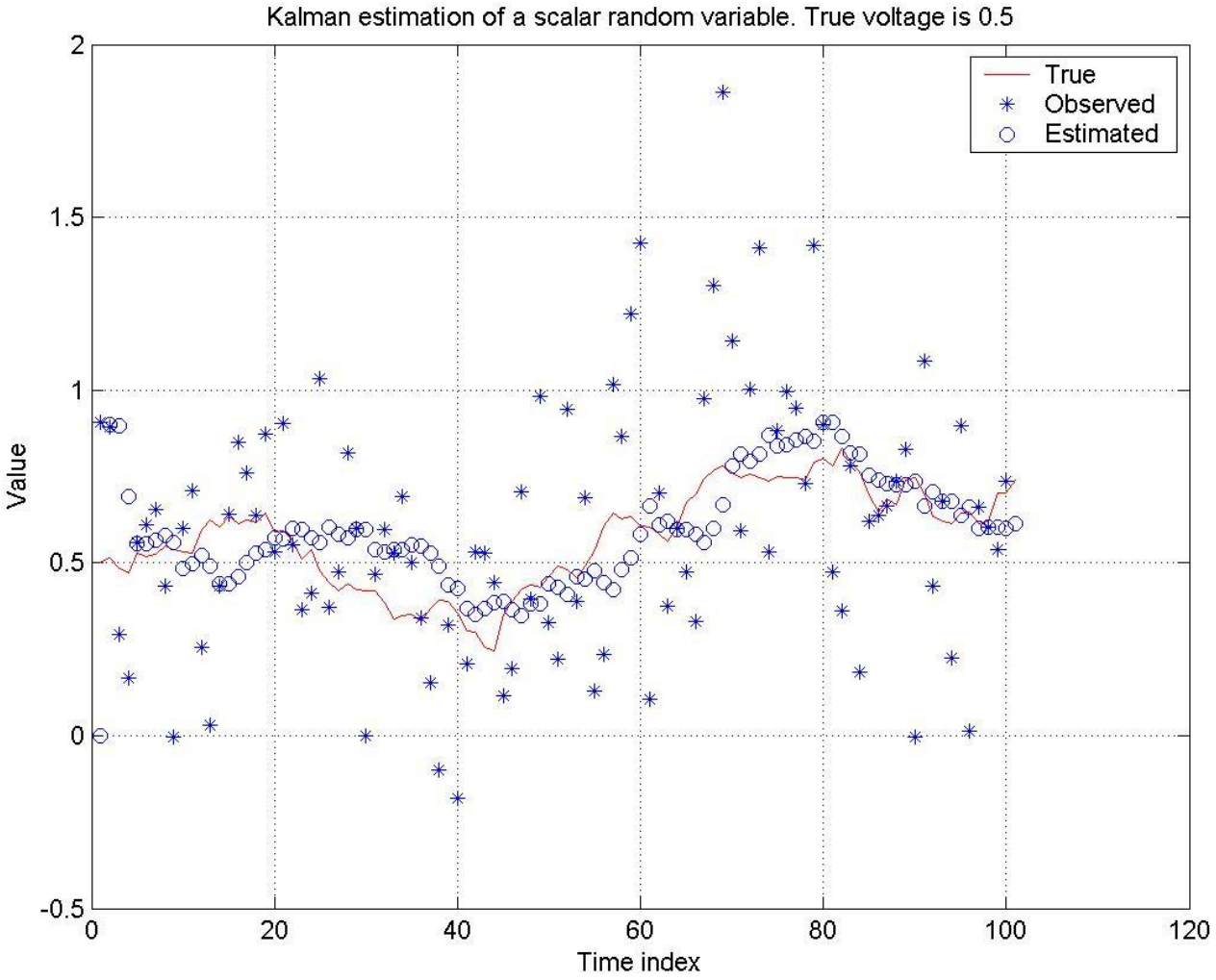
with $\boldsymbol{\varepsilon}(n+1, n)$: the **predicted state-error vector**:

$$\boldsymbol{\varepsilon}(n+1,n) = \mathbf{x}(n+1) - \hat{\mathbf{x}}(n+1)$$

7. Kalman Filter: Example 1 (1)

- Formulate a scalar voltage measurement experiment using state-space equations if the true voltage is 0.5 and the process and measurement noise variances are 0.001 and 0.1 respectively. Investigate the performance of the Kalman filter for this problem.

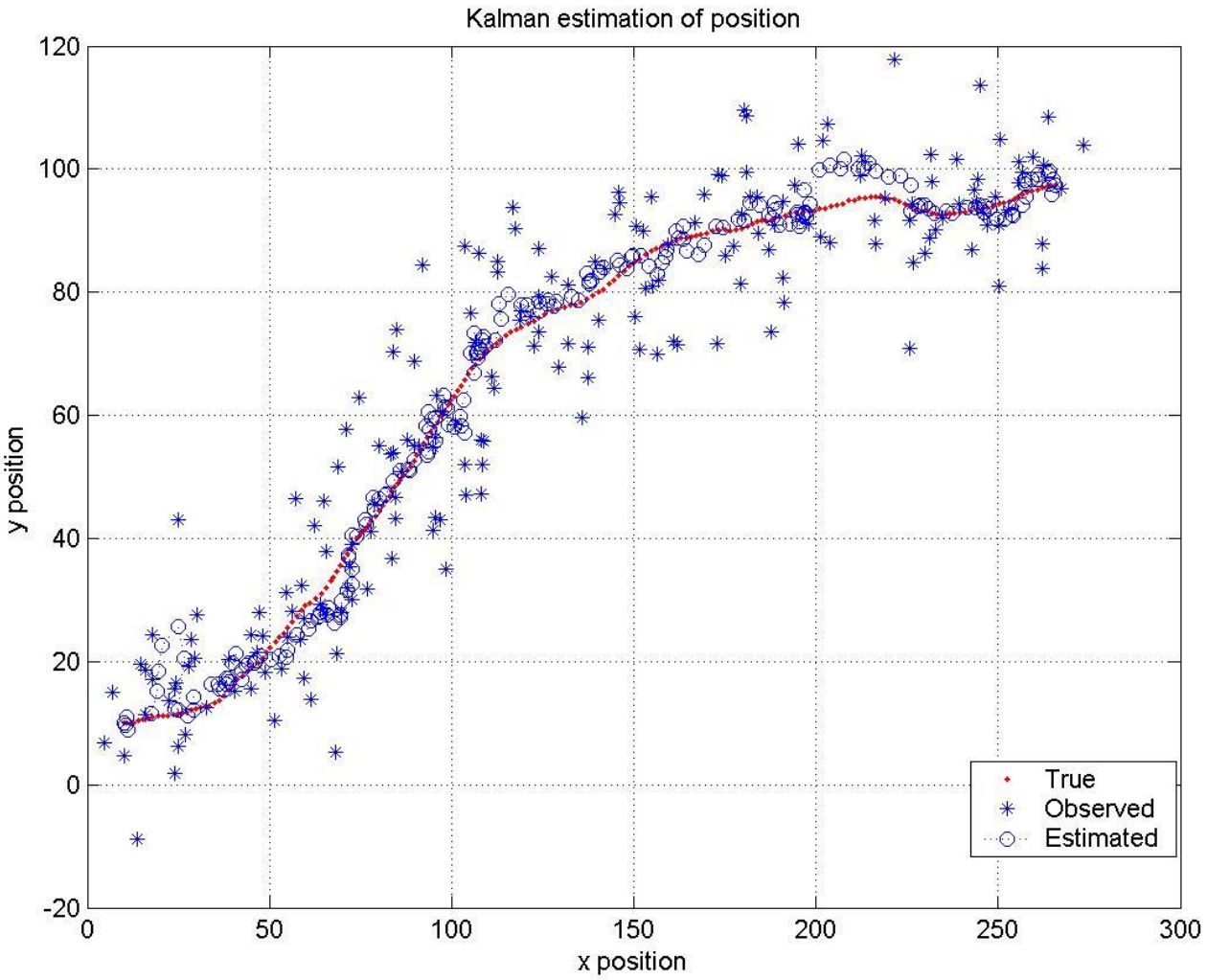
7. Kalman Filter: Example 1 (2)



7. Kalman Filter: Example 2 (1)

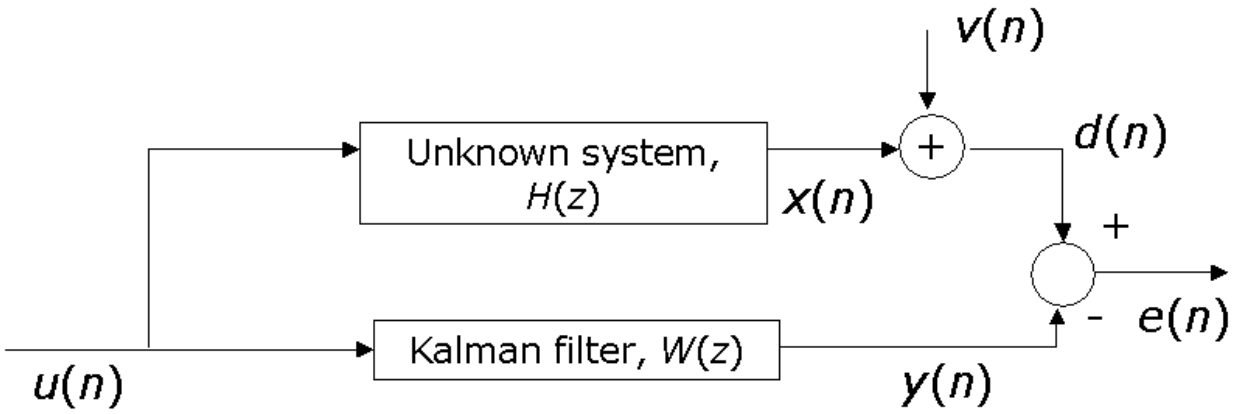
- Formulate the state-space equations for a particle moving with a **constant velocity** and apply the Kalman filter to track its trajectory.

7. Kalman Filter: Example 2 (2)



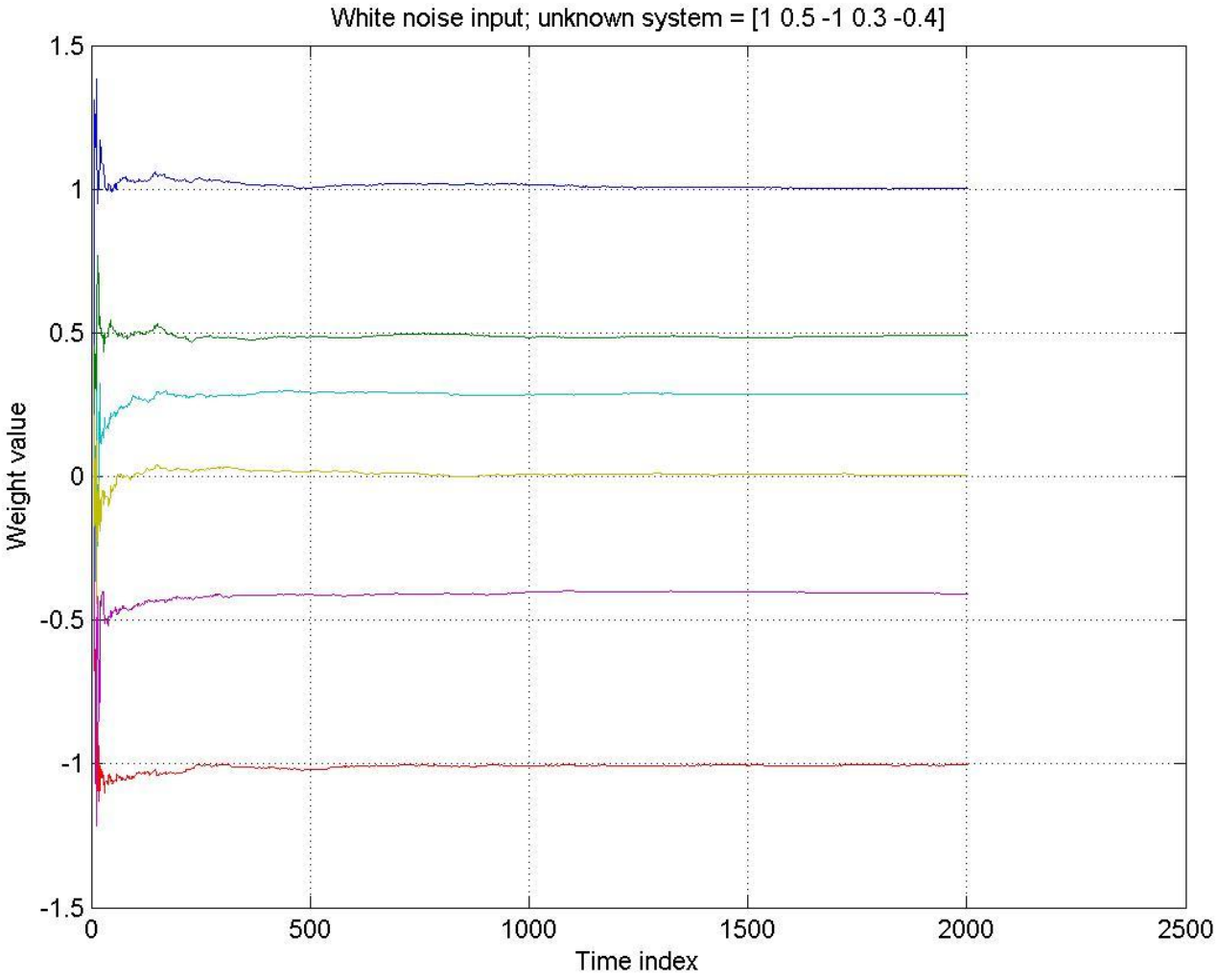
7. Kalman Filter: Example 3 (1)

- Consider the **system identification problem** shown below:

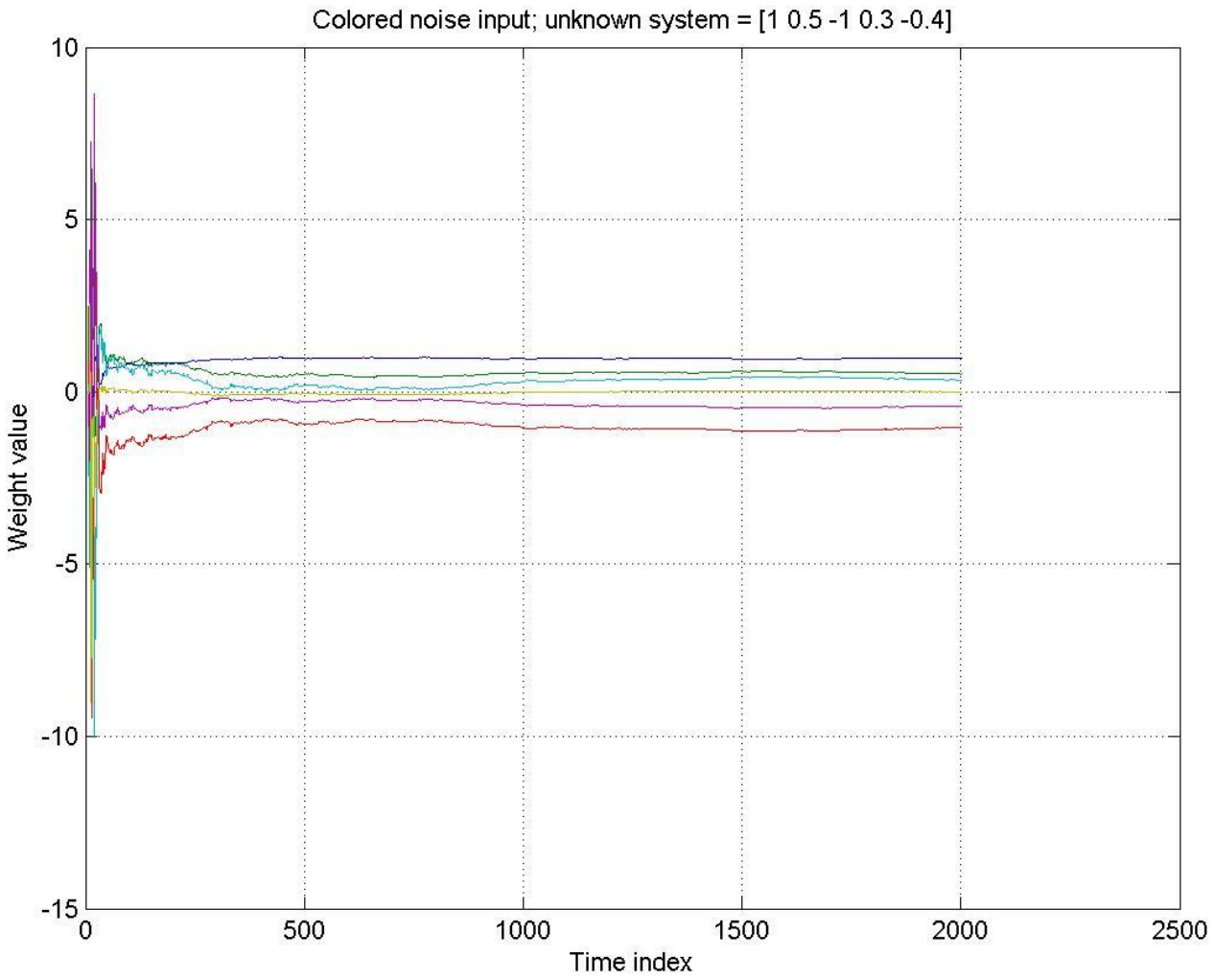


$v(n)$ is a zero-mean, white noise sequences with variance 0.1. If $H(z) = 1+0.5z^{-1} - z^{-2}+0.3\ z^{-3} - 0.4\ z^{-4}$, evaluate the performance of the Kalman filtering algorithm when $u(n)$ is (a) a white noise sequence, and (b) a colored noise sequence.

7. Kalman Filter: Example 3 (2)

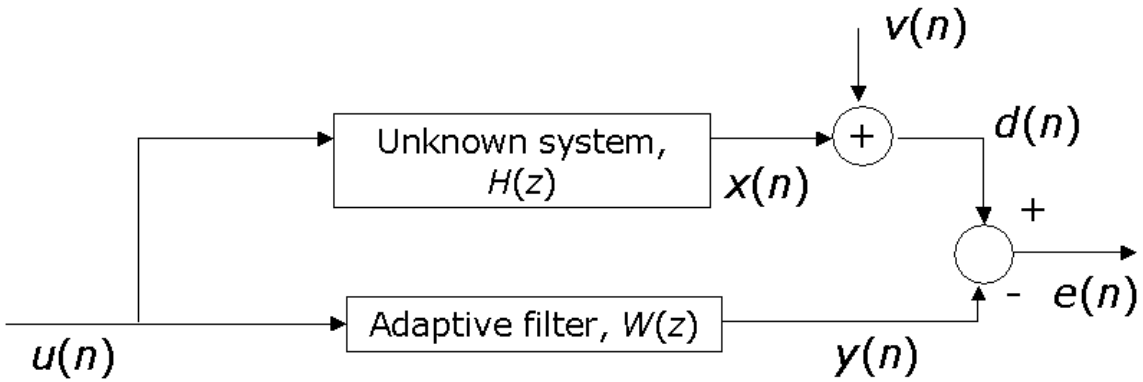


7. Kalman Filter: Example 3 (3)



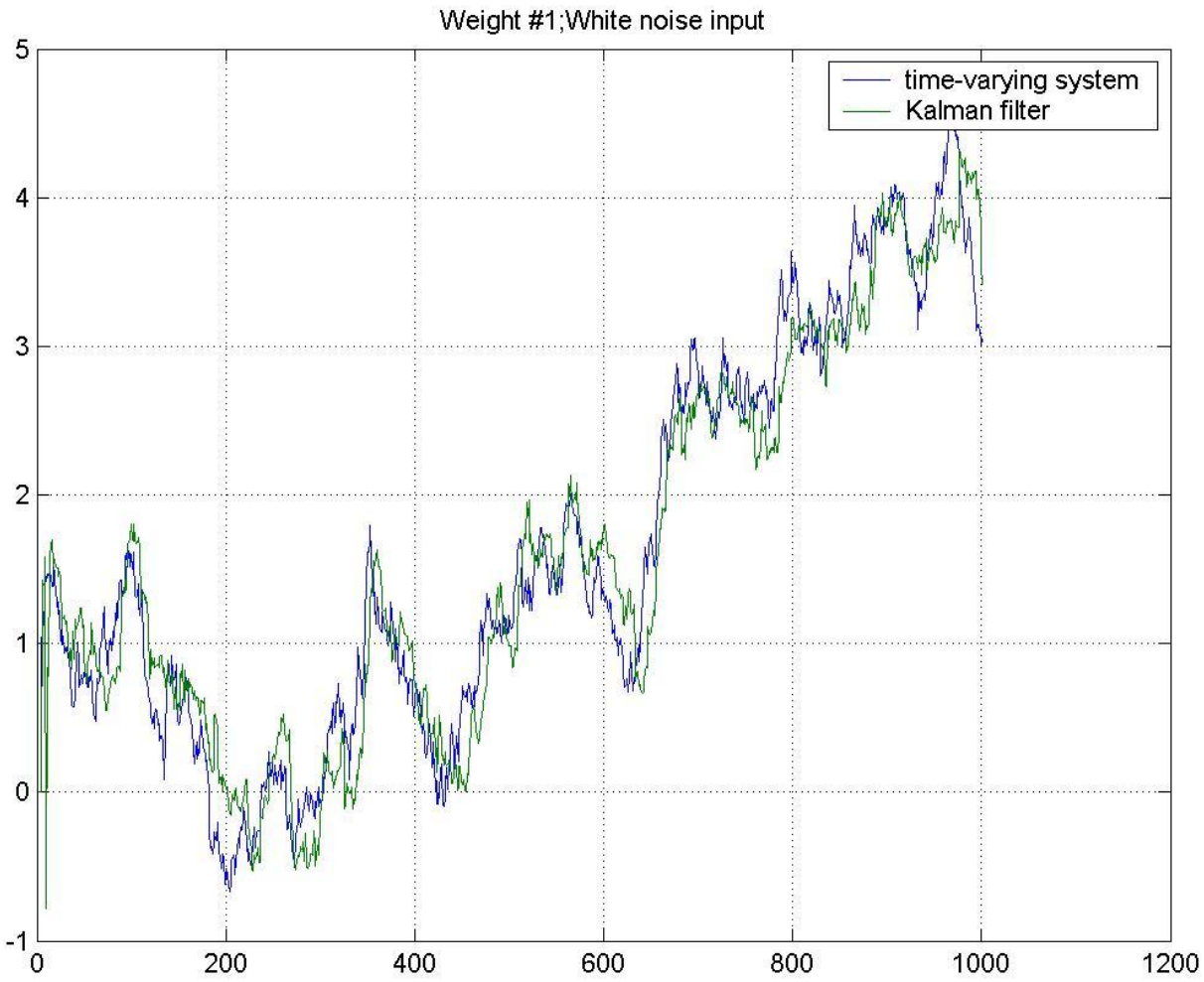
7. Kalman Filter: Example 4 (1)

- Consider the **system identification problem** shown below:

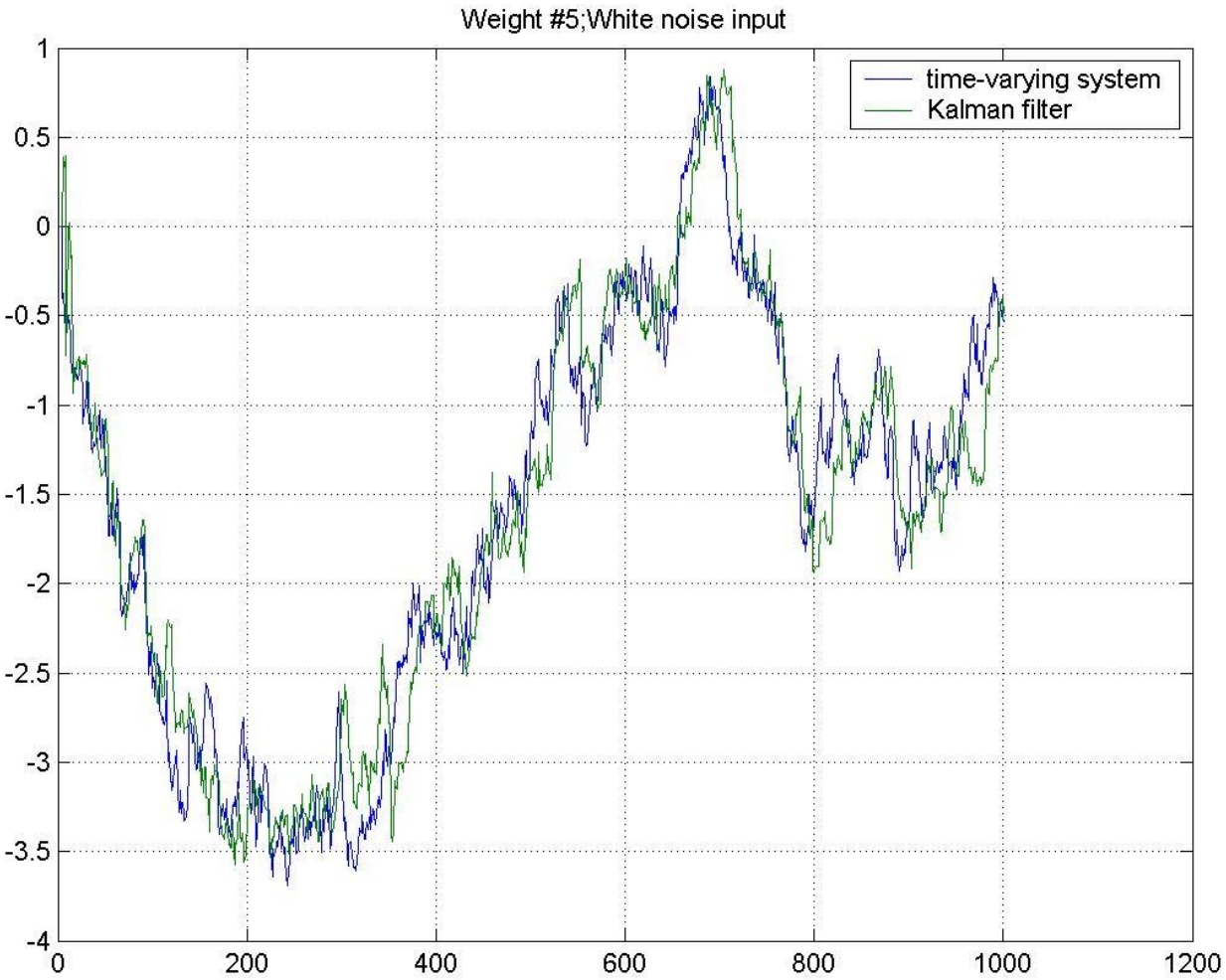


$v(n)$ is a zero-mean, white noise sequences with variance 1. The unknown system has a time-varying impulse response represented by $h(n)=a\ h(n-1)+b(n)$, where a is the model parameter, $b(n)$ is a white noise process with variance of 0.01, and $h(0)$=[1 0.5 -1 0.3 -0.4]. Evaluate the tracking performance of the Kalman filtering algorithm when $u(n)$ is (a) a white noise sequence, and (b) a colored noise sequence.

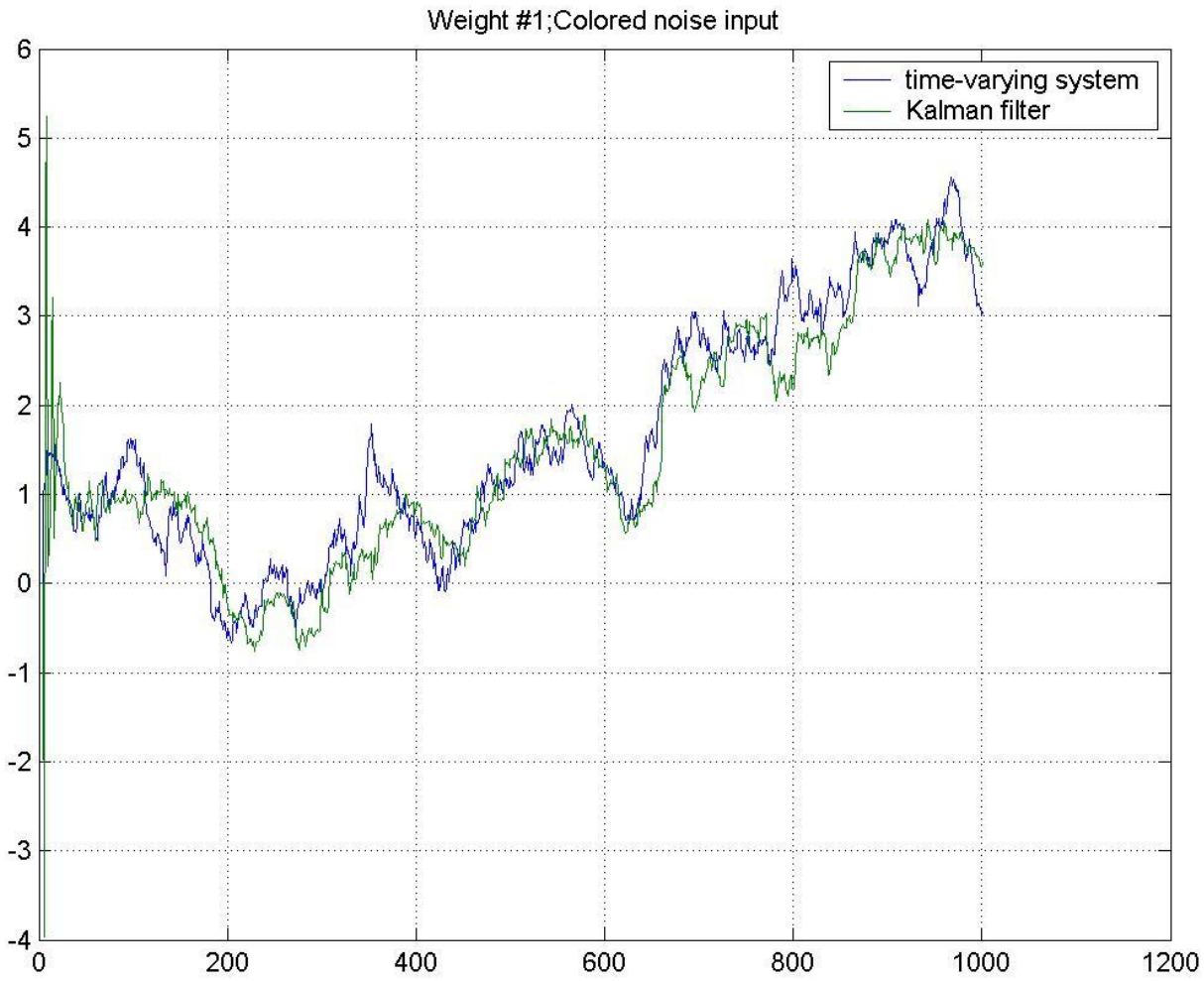
7. Kalman Filter: Example 4 (2)



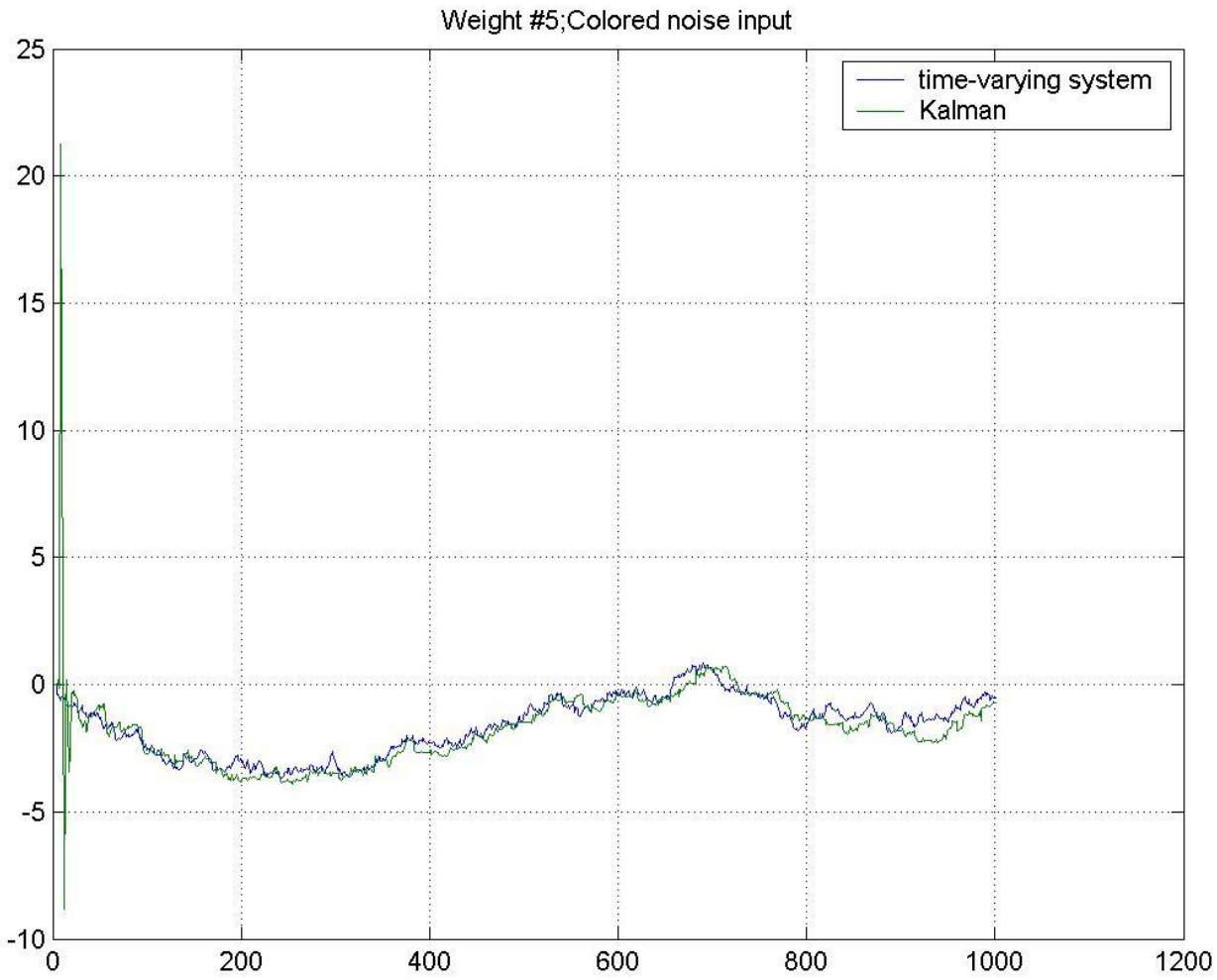
7. Kalman Filter: Example 4 (3)



7. Kalman Filter: Example 4 (4)



7. Kalman Filter: Example 4 (5)



7. Extended Kalman Filter (1)

- **Nonlinear state-space models:**

  Process equation:

$$\mathbf{x}(n+1) = \mathbf{F}(n, \mathbf{x}(n)) + \mathbf{v}_1(n)$$

  ($M\times1$)-*process noise v*ector $\mathbf{v}_1(n)$ is zero-mean white-noise process with correlation matrix:

$$E\left[\mathbf{v}_1(n)\mathbf{v}_1^H(k)\right] = \begin{cases} \mathbf{Q}_1(n), & n = k \\ \mathbf{0}, & n \neq k \end{cases}$$

  Measurement equation:

$$\mathbf{y}(n) = \mathbf{C}(n, \mathbf{x}(n)) + \mathbf{v}_2(n)$$

  ($N\times1$)-*measurement noise v*ector $\mathbf{v}_2(n)$ is zero-mean white- noise process with correlation matrix:

$$E\left[\mathbf{v}_2(n)\mathbf{v}_2^H(k)\right] = \begin{cases} \mathbf{Q}_2(n), & n = k \\ \mathbf{0}, & n \neq k \end{cases}$$

7. Extended Kalman Filter (2) [1]

- **Using first-order Taylor approximation, with:**

$$\mathbf{F}(n+1,n) = \left.\frac{\partial \mathbf{F}(n,\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}(n|\mathcal{Y}_n)} \tag{7.113}$$

$$\mathbf{C}(n) = \left.\frac{\partial \mathbf{C}(n,\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}(n|\mathcal{Y}_{n-1})} \tag{7.114}$$

Therefore:

$$\mathbf{F}(n,\mathbf{x}(n)) \simeq \mathbf{F}(n, \hat{\mathbf{x}}(n|\mathcal{Y}_n)) + \mathbf{F}(n+1,n)\,[\mathbf{x}(n) - \hat{\mathbf{x}}(n|\mathcal{Y}_n)]$$

$$\mathbf{C}(n, \mathbf{x}(n)) \simeq \mathbf{C}(n, \hat{\mathbf{x}}(n|\mathcal{Y}_{n-1})) + \mathbf{C}(n)[\mathbf{x}(n) - \hat{\mathbf{x}}(n|\mathcal{Y}_{n-1})]$$

7. Extended Kalman Filter (3) [1]

Input vector process

Observations = $\{\mathbf{y}(1), \mathbf{y}(2), \ldots, \mathbf{y}(n)\}$

Known parameters

Nonlinear state transition matrix = $\mathbf{F}(n, \mathbf{x}(n))$

Nonlinear measurement matrix = $\mathbf{C}(n, \mathbf{x}(n))$

Correlation matrix of process noise vector = $\mathbf{Q}_1(n)$

Correlation matrix of measurement noise vector = $\mathbf{Q}_2(n)$

Computation: $n = 1, 2, 3, \ldots$

$$\mathbf{G}_f(n) = \mathbf{K}(n, n-1)\mathbf{C}^H(n)[\mathbf{C}(n)\mathbf{K}(n, n-1)\mathbf{C}^H(n) + \mathbf{Q}_2(n)]^{-1}$$

$$\boldsymbol{\alpha}(n) = \mathbf{y}(n) - \mathbf{C}(n, \hat{\mathbf{x}}(n|\mathcal{Y}_{n-1}))$$

$$\hat{\mathbf{x}}(n|\mathcal{Y}_n) = \hat{\mathbf{x}}(n|\mathcal{Y}_{n-1}) + \mathbf{G}_f(n)\boldsymbol{\alpha}(n)$$

$$\hat{\mathbf{x}}(n+1|\mathcal{Y}_n) = \mathbf{F}(n, \hat{\mathbf{x}}(n|\mathcal{Y}_n))$$

$$\mathbf{K}(n) = [\mathbf{I} - \mathbf{G}_f(n)\mathbf{C}(n)]\mathbf{K}(n, n-1)$$

$$\mathbf{K}(n+1, n) = \mathbf{F}(n+1, n)\mathbf{K}(n)\mathbf{F}^H(n+1, n) + \mathbf{Q}_1(n)$$

Note: The linearized matrices $\mathbf{F}(n+1, n)$ and $\mathbf{C}(n)$ are computed from their nonlinear counterparts $\mathbf{F}(n, \mathbf{x}(n))$ and $\mathbf{C}(n, \mathbf{x}(n))$ using Eqs. (7.113) and (7.114), respectively.

Initial conditions

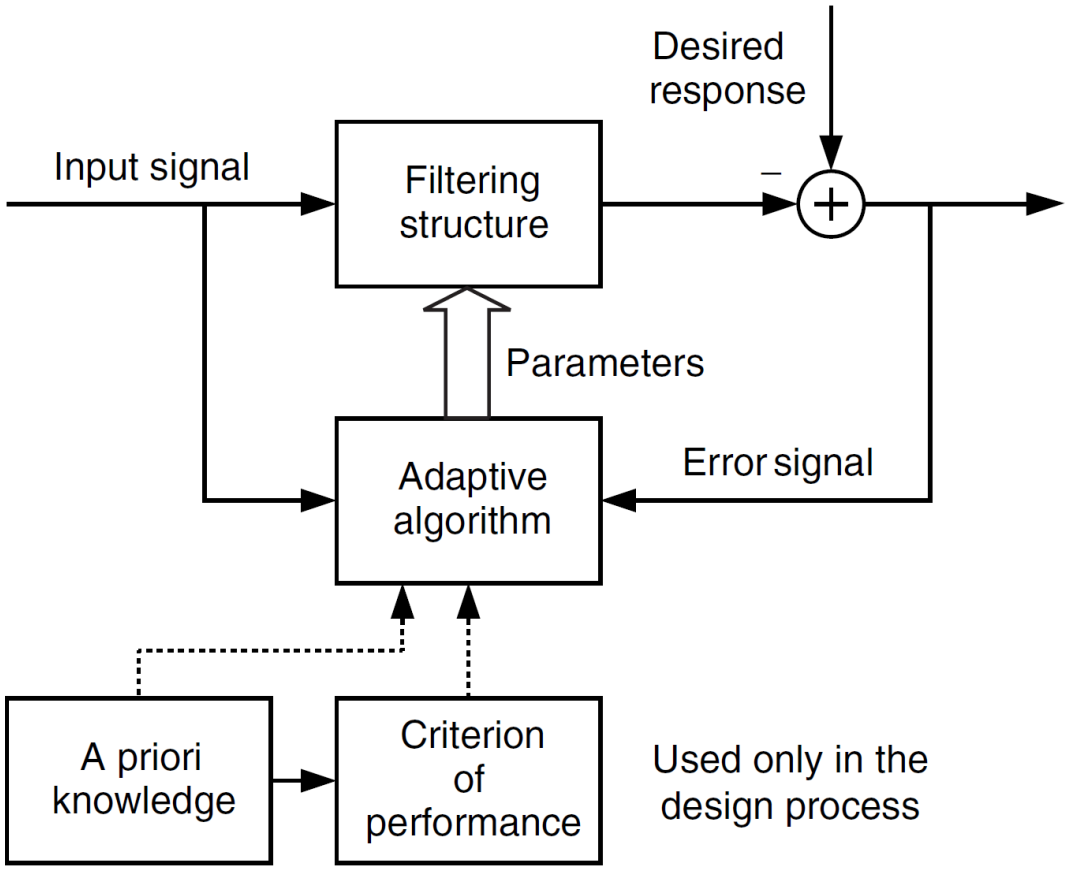
$$\hat{\mathbf{x}}(1|\mathcal{Y}_0) = E[\mathbf{x}(1)]$$

$$\mathbf{K}(1,0) = E[(\mathbf{x}(1) - E[\mathbf{x}(1)])(\mathbf{x}(1) - E[\mathbf{x}(1)])^H] = \boldsymbol{\Pi}_0$$

Chapter 8:
Adaptive Filter

- Method of Steepest Descent.
- Least-Mean-Square Algorithm.
- Examples of Adaptive Filtering:
  - Adaptive Equalization and
  - Adaptive Beamforming.

8. Linear Adaptive Filter: Principles



General principle of adaptive filter

8. LAF: Method of Steepest Descent

- The method of steepest descent is the iterative solution of the Wiener-Hopf equations.

- Wiener-Hopf equations: $\mathbf{R}\mathbf{w}_{opt} = \mathbf{p}$

$$\mathbf{R} = E\left[\mathbf{u}(n)\mathbf{u}^H(n)\right] \in C^{M\times M}, \quad \mathbf{R} = \mathbf{R}^H$$
$$\mathbf{p} = E\left[\mathbf{u}(n)d^*(n)\right] \in C^M$$

Cost function:

$$J(\mathbf{w}) = E\left[e(n)e^*(n)\right] = \sigma_d^2 - \mathbf{w}^H\mathbf{p} - \mathbf{p}^H\mathbf{w} + \mathbf{w}^H\mathbf{R}\mathbf{w}$$

MMSE:

$$J_{\min} = J(\mathbf{w}_{opt}) = \sigma_d^2 - \mathbf{p}^H\mathbf{w}_{opt} = \sigma_d^2 - \mathbf{p}^H\mathbf{R}^{-1}\mathbf{p}$$

8. LAF: Steepest Descent Algorithm (1)

Summary:

1. Begin with an initial guess of the weight vector: **w**(0)
2. Compute the direction of steepest descent pointing from the initial location on the error surface
3. Compute the next guess of the weight vector w by making a change in accordance with the direction of steepest descent
4. Go back to step 2 and repeat the process

8. LAF: Steepest Descent Algorithm (2)

- Weight vector at iteration n+1:

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \mu \frac{\partial J(\mathbf{w}(n))}{\partial \mathbf{w}^*(n)}$$

$$\frac{\partial J(\mathbf{w}(n))}{\partial \mathbf{w}^*(n)} = -\mathbf{p} + \mathbf{R}\mathbf{w}(n)$$

Therefore: $\mathbf{w}(n+1) = \mathbf{w}(n) + \mu(\mathbf{p} - \mathbf{R}\mathbf{w}(n))$

where μ is the **stepsize**, controls the incremental correction.

- **Question:**

  How to choose μ ?

  How many iterations to obtain the optimum weight vector ?...

8. LAF: Stability of the Algorithm

- To meet the stability, choosing μ in accordance with:

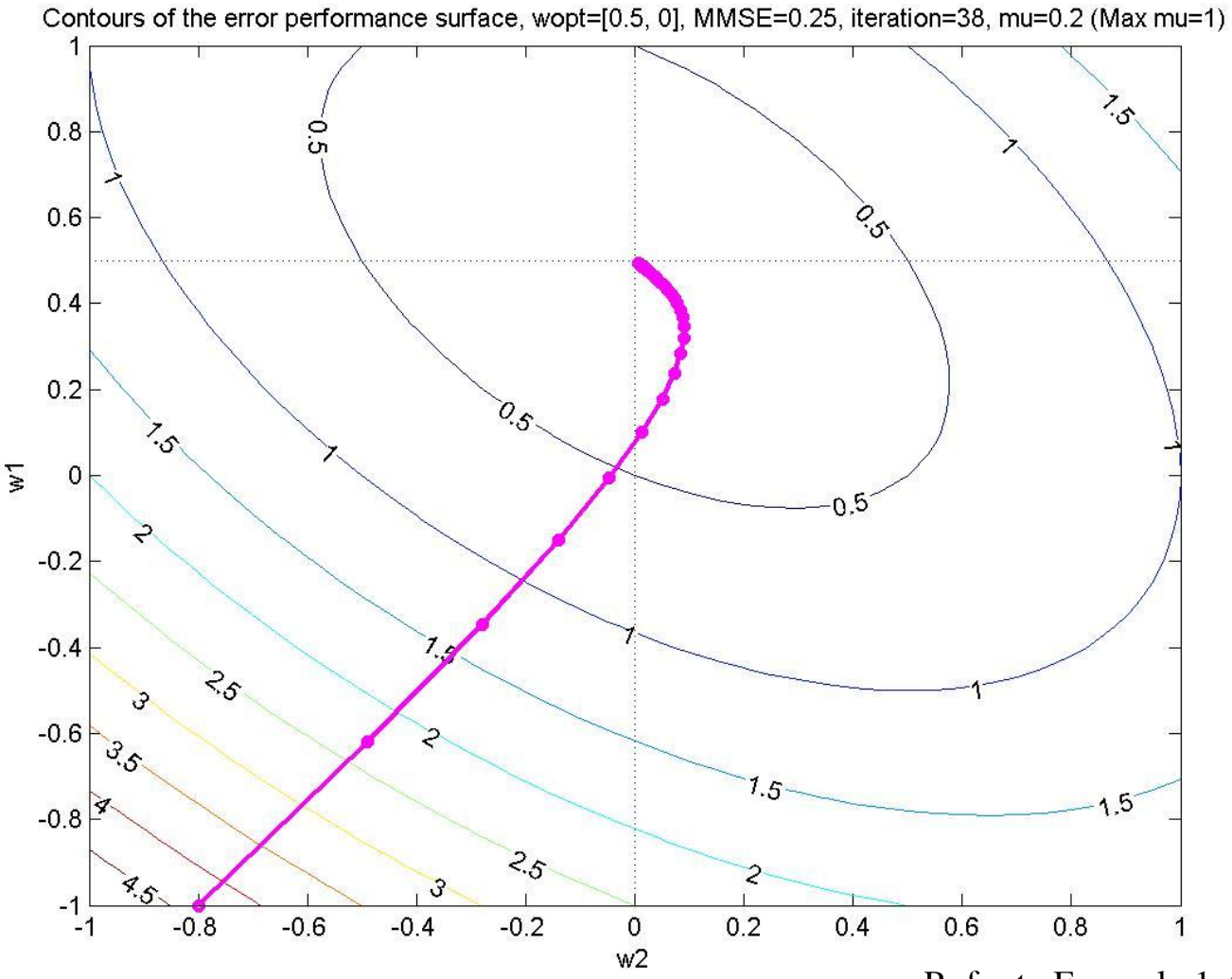
$$0 < \mu < \frac{2}{\lambda_{\max}}$$

where is the maximum eigenvalue of the correlation matrix **R**.

- Other useful criterion for choosing μ:

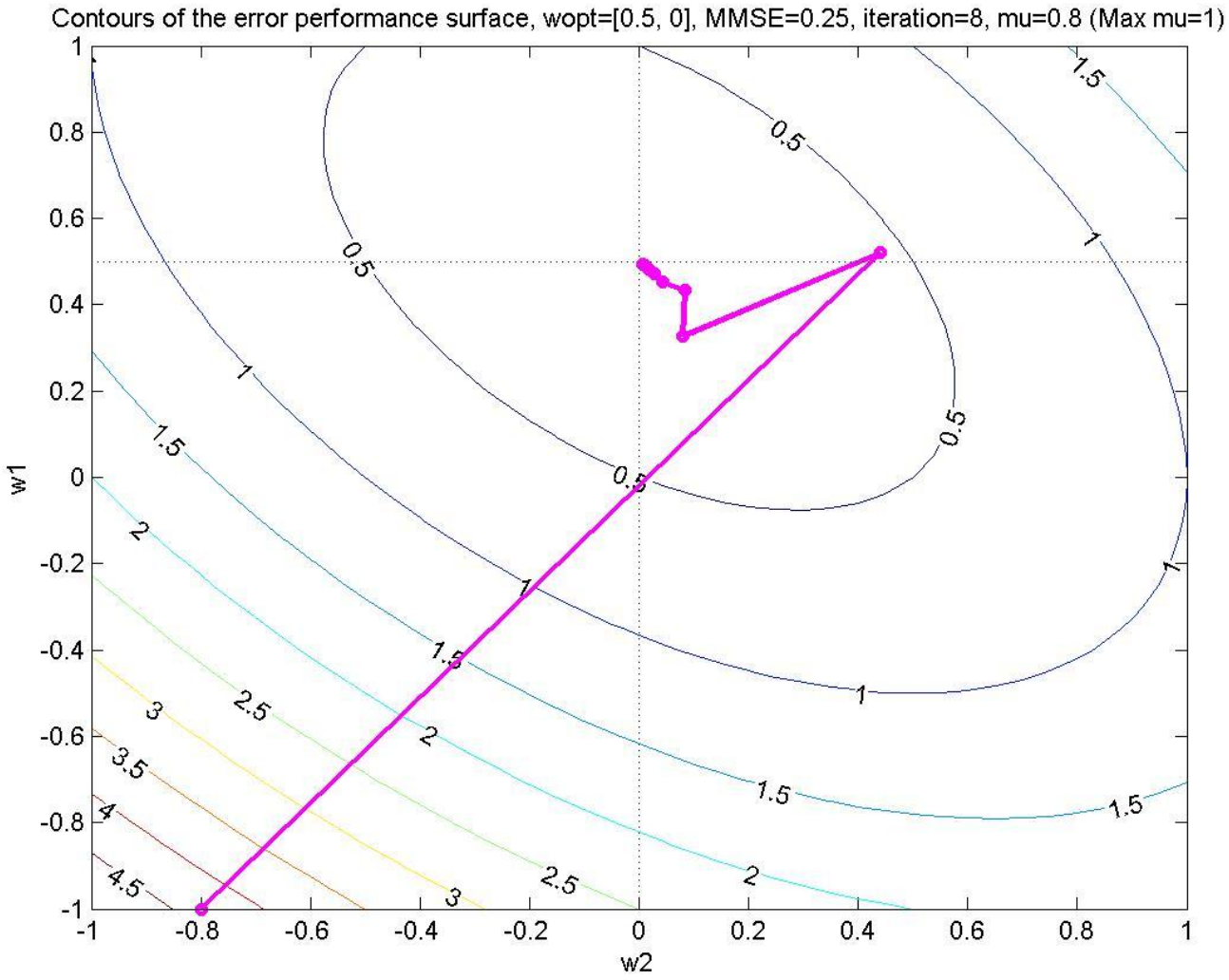
$$0 < \mu < \frac{2}{trace(\mathbf{R})} = \frac{2}{Mr(0)}$$

8. LAF: Examples (1)

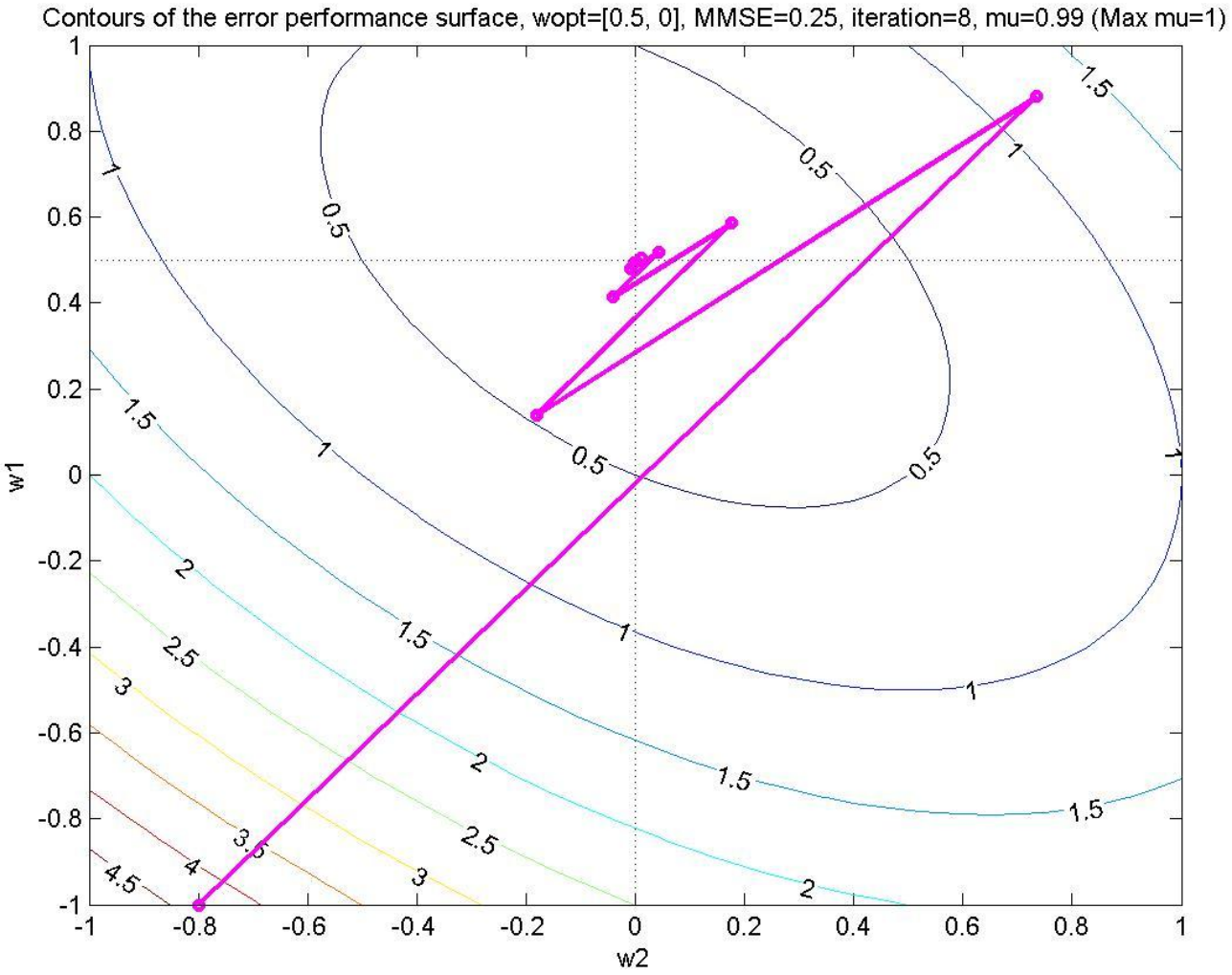


Refer to Example 1, Chapter 5

8. LAF: Examples (2)

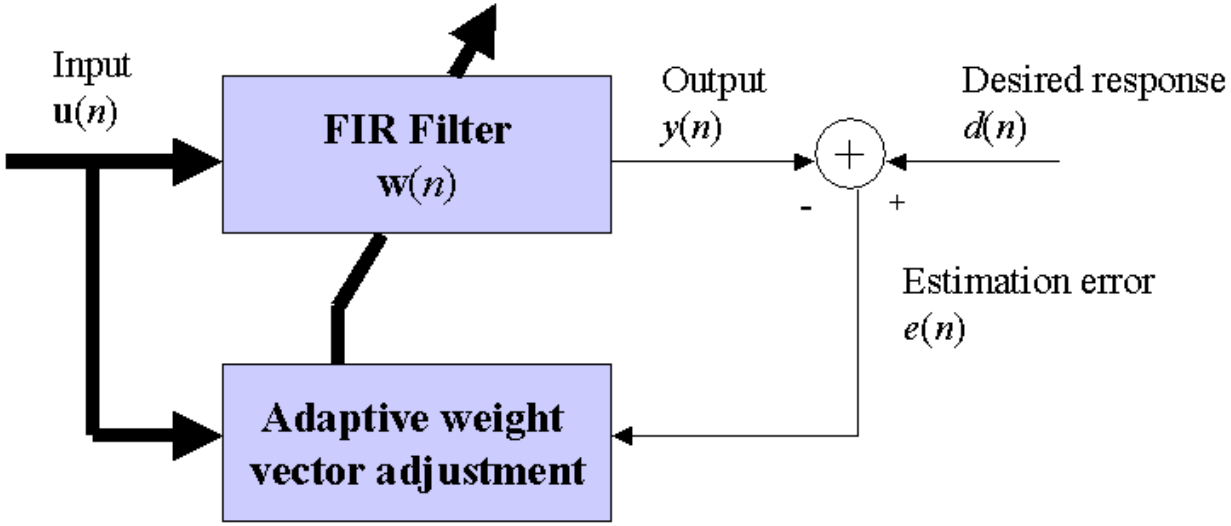


8. LAF: Examples (3)



8. LAF: Least Mean Square (LMS) Algorithm (1)

- The LMS algorithm is a stochastic gradient algorithm consisting of 2 basic processes:
  - **Filtering process:** computing the output of FIR filter $\mathbf{y}(n)=\mathbf{w}^H(n)\mathbf{u}(n)$ and generating an estimation error $e(n)$
  - **Adaptation process:** automatic adjustment of the weight vector of the FIR filter in accordance with the estimation error $e(n)$



8. LAF: LMS Algorithm (2)

- The **steepest descent algorithm** is given by:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu(\mathbf{p} - \mathbf{R}\mathbf{w}(n))$$

- Note that this equation requires the knowledge of the autocorrelation matrix **R** and the crosscorrelation vector **p**.

- In the **LMS** formulation, these matrix and vector are replaced by the **instantaneous values**, *i.e.*,

  $\mathbf{R}(n) = \mathbf{u}(n)\mathbf{u}^H(n)$ and $\mathbf{p}(n) = \mathbf{u}(n)d^*(n)$.

- With this replacement, the weight update equation becomes:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu\mathbf{u}(n)e^*(n)$$

8. LAF: Convergence Analysis (1)

- Because of the use of instantaneous estimates of the autocorrelation matrix and the crosscorrelation vector, the convergence of the LMS algorithm is noisy.
- Similar to the steepest descent algorithm, the convergence of the LMS algorithm is controlled by the stepsize μ.
- The bounds on μ for the stability of LMS adaptation are $0 \le \mu \le 2/\lambda_{\max}$, where $\lambda_{\max}$ is the maximum eigenvalue of the input autocorrelation matrix.
- Other useful criterion for choosing μ:

$$0 < \mu < \frac{2}{trace(\mathbf{R})} = \frac{2}{Mr(0)} = \frac{2}{\sum_{k=0}^{M-1} E\left[|u(n-k)|^2\right]}$$

8. LAF: Convergence Analysis (2)

- Convergence criterion in mean square:

$$J(n) = E\left[\left|e(n)\right|^2\right] \rightarrow \text{constant, as } n \rightarrow \infty$$

- The average time constant of the LMS algorithm is given by:

$$\tau_{mse,av} = \frac{1}{2\mu\lambda_{av}}, \text{ where } \lambda_{av} = \frac{1}{M}\sum_{k=1}^{M}\lambda_k$$

- Thus smaller values of the stepsize parameter, μ, result in longer convergence times.

8. LAF: Excess Mean-Squared Error (1)

- Let $J(n)$ denote the mean-squared error due to LMS algorithm at iteration n:

$$J(n) = E\left[|e(n)|^2\right] = J_{\min} + J_{ex}(n)$$

where $J_{\min}$ is the **minimum mean-squared error** produced by the optimum Wiener filter, and $J_{\text{ex}}(n)$ is the **excess mean-squared error**.

$\Rightarrow$ The LMS algorithm alaways produces mean-squared error $J(n)$ that is in excess of minimum mean-squared error $J_{\min}$

8. LAF: Excess Mean-Squared Error (2)

- When the LMS algorithm is convergent in the mean square, $J_{ex}(n) \to J_{ex}(\infty)$: is steady-state value as $n \to \infty$:

$$J_{ex}(\infty) = J(\infty) - J_{\min} = J_{\min} \sum_{i=1}^{M} \frac{\mu\lambda_i}{2 - \mu\lambda_i}$$

Smaller values of the stepsize μ, will therefore provide lower excess mean-squared error.

- **Misadjustment** K: is a measure providing how close the mean-squared error of the LMS algorithm is to the minimum mean-squared error $J_{\min}$ (after completing adaptation process)

$$K = \frac{J_{ex}(\infty)}{J_{\min}} = \sum_{i=1}^{M} \frac{\mu\lambda_i}{2 - \mu\lambda_i}$$

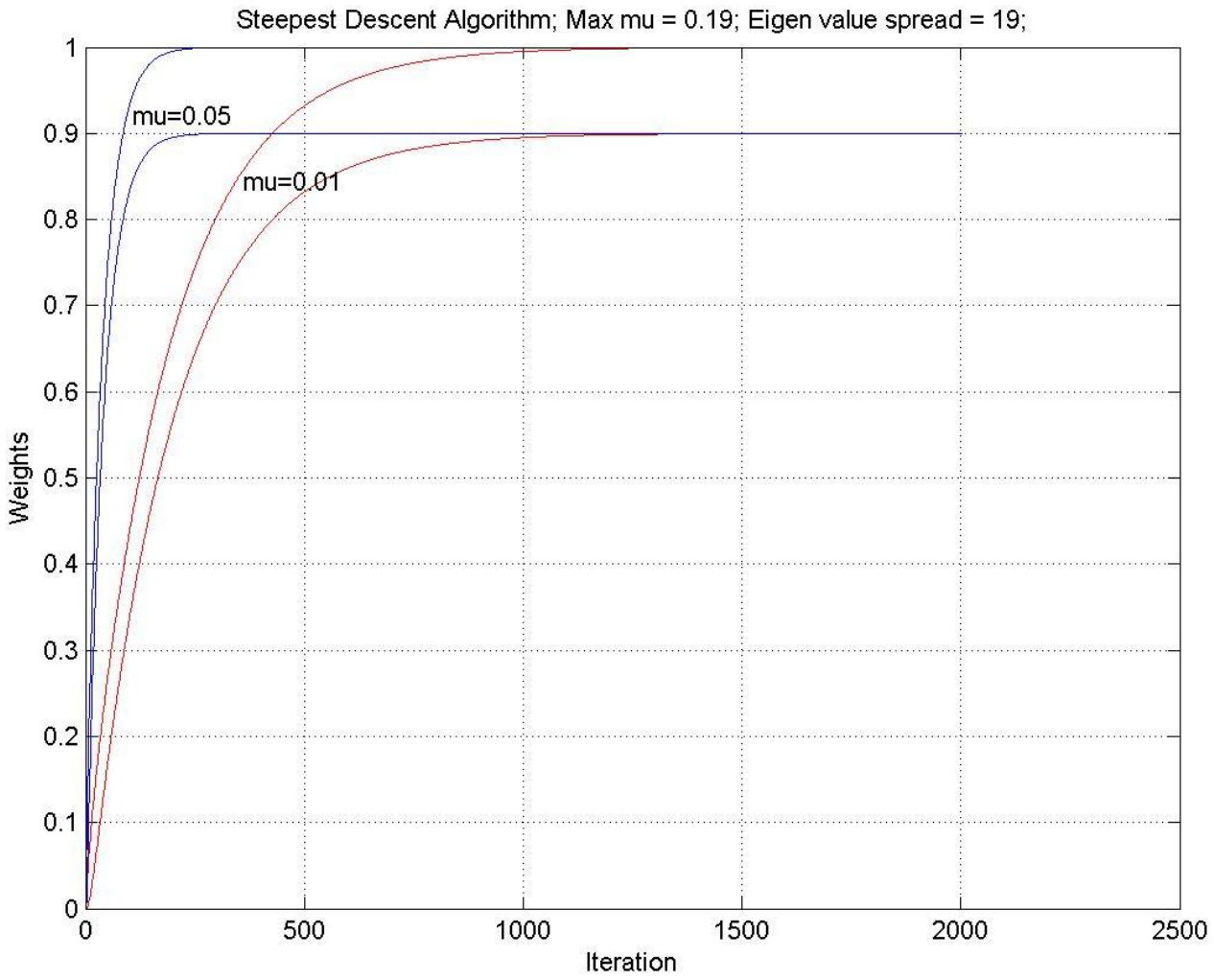
8. LAF: Example of Adaptive Predictor (1)

- Consider the first-order AR model, $u(n)+a_1u(n-1)=v(n)$, where $v(n)$ is zero-mean white noise. Evaluate the performance of the steepest-descent, and LMS algorithms when,
  - $a_1 = 0.9$;

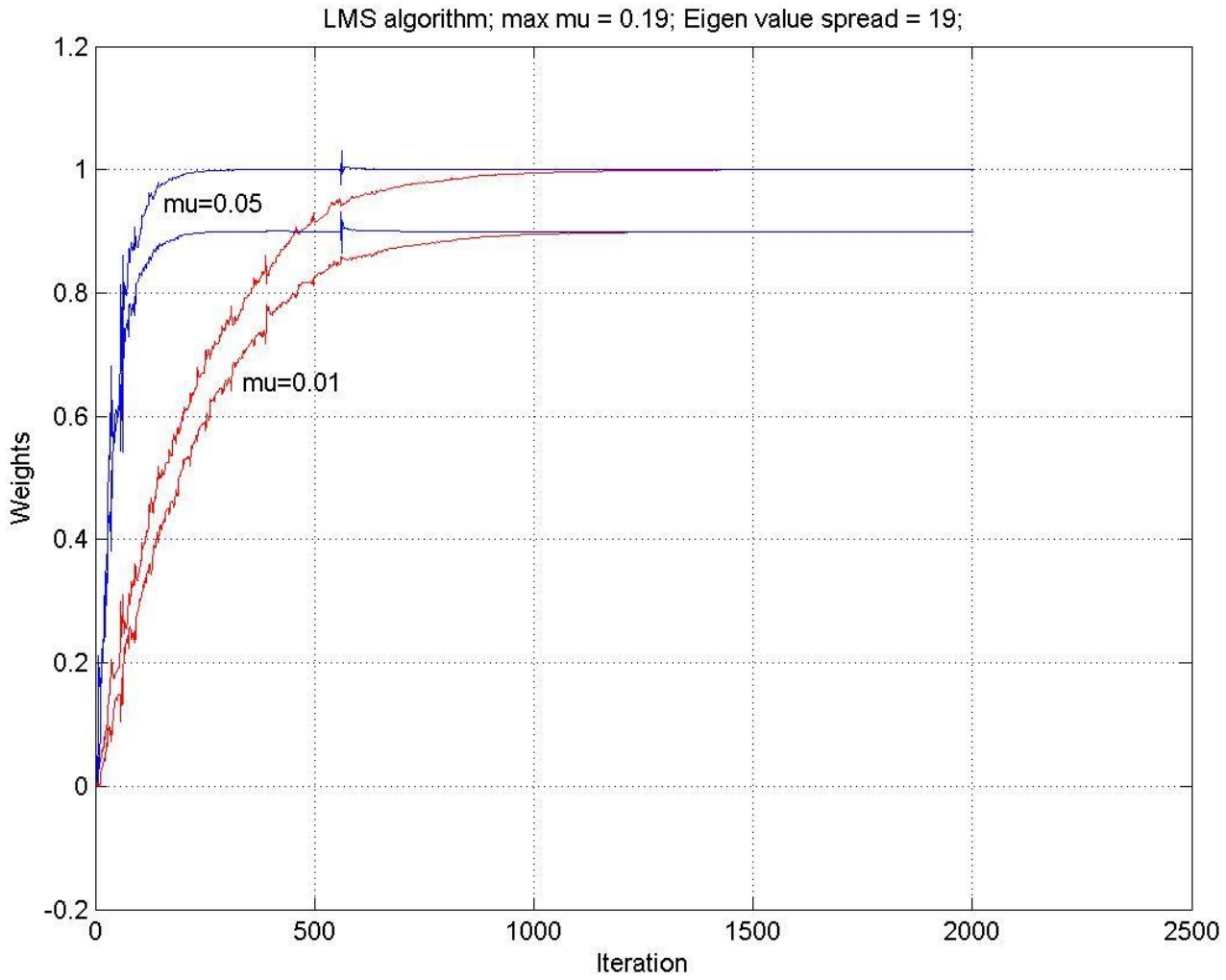
  use different μ values. Assume that the variance of $v(n)$ is unity.

  See p. 406, [1]

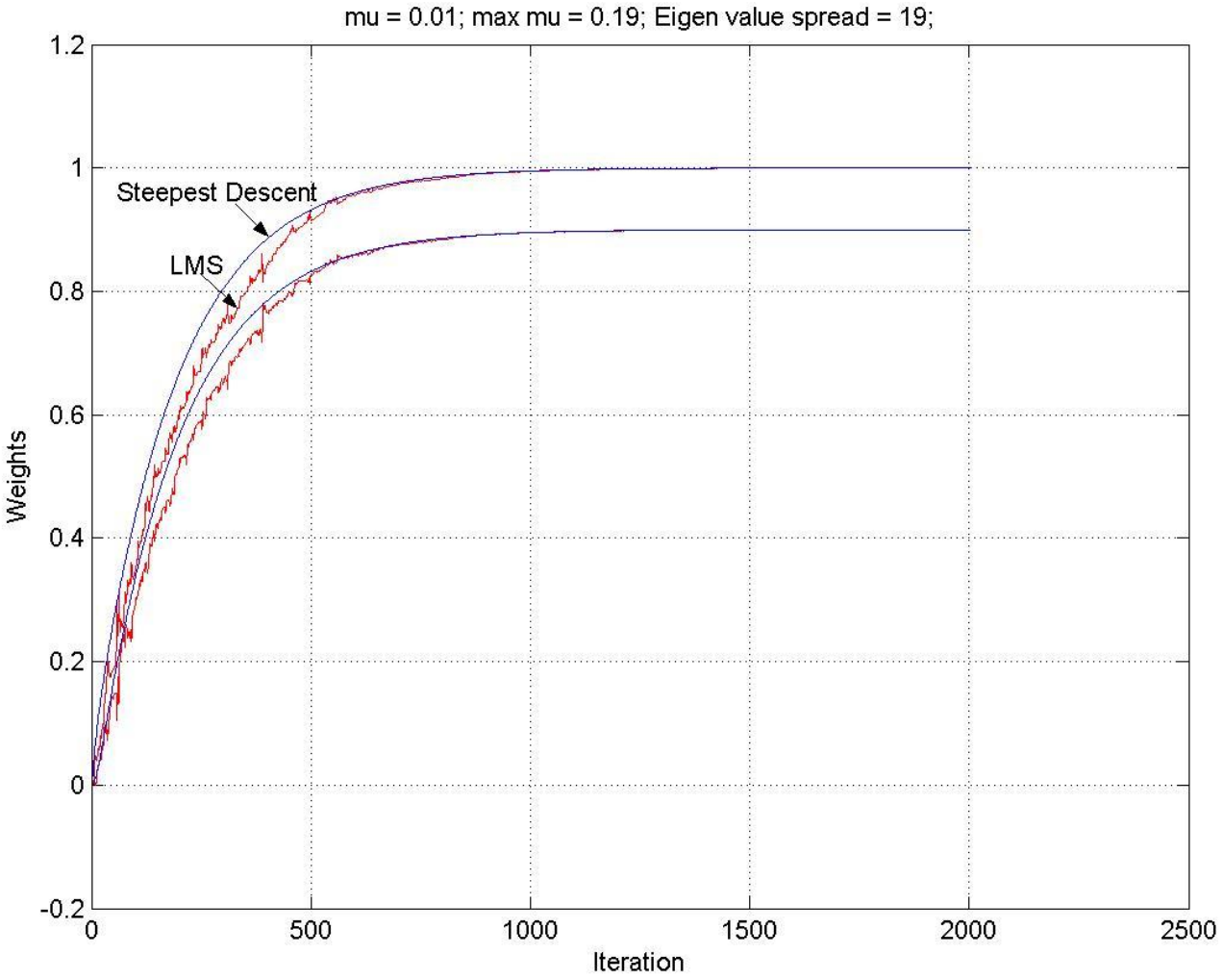
8. LAF: Example of Adaptive Predictor (2)



8. LAF: Example of Adaptive Predictor (3)

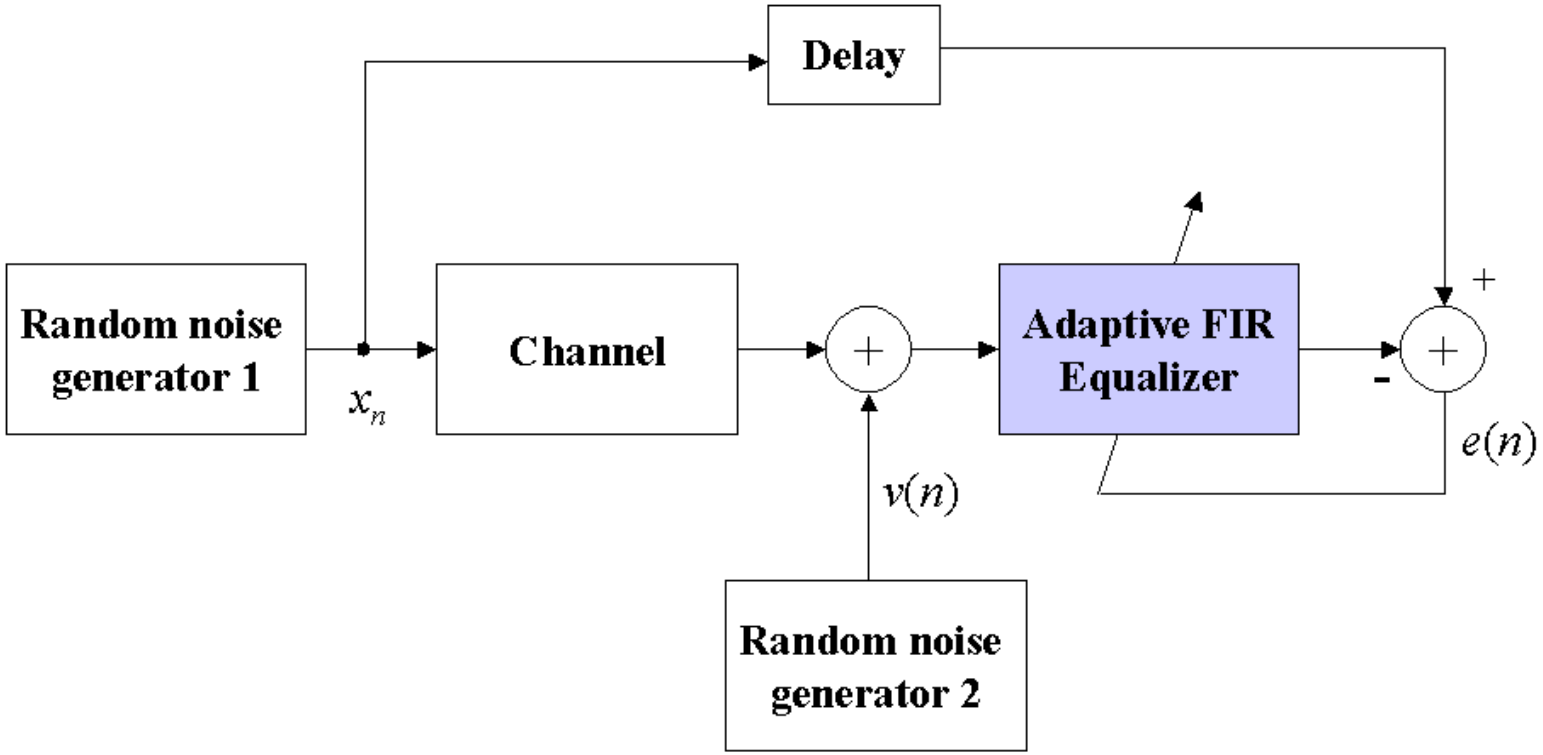


8. LAF: Example of Adaptive Predictor (4)



8. LAF: Example of Adaptive Equalizer (1)

- The following block diagram shows a typical adaptive equalizer. The input x_n is a zero-mean Bernoulli random variable with unit variance. The noise $v(n)$ has zero-mean and a variance of 0.001.



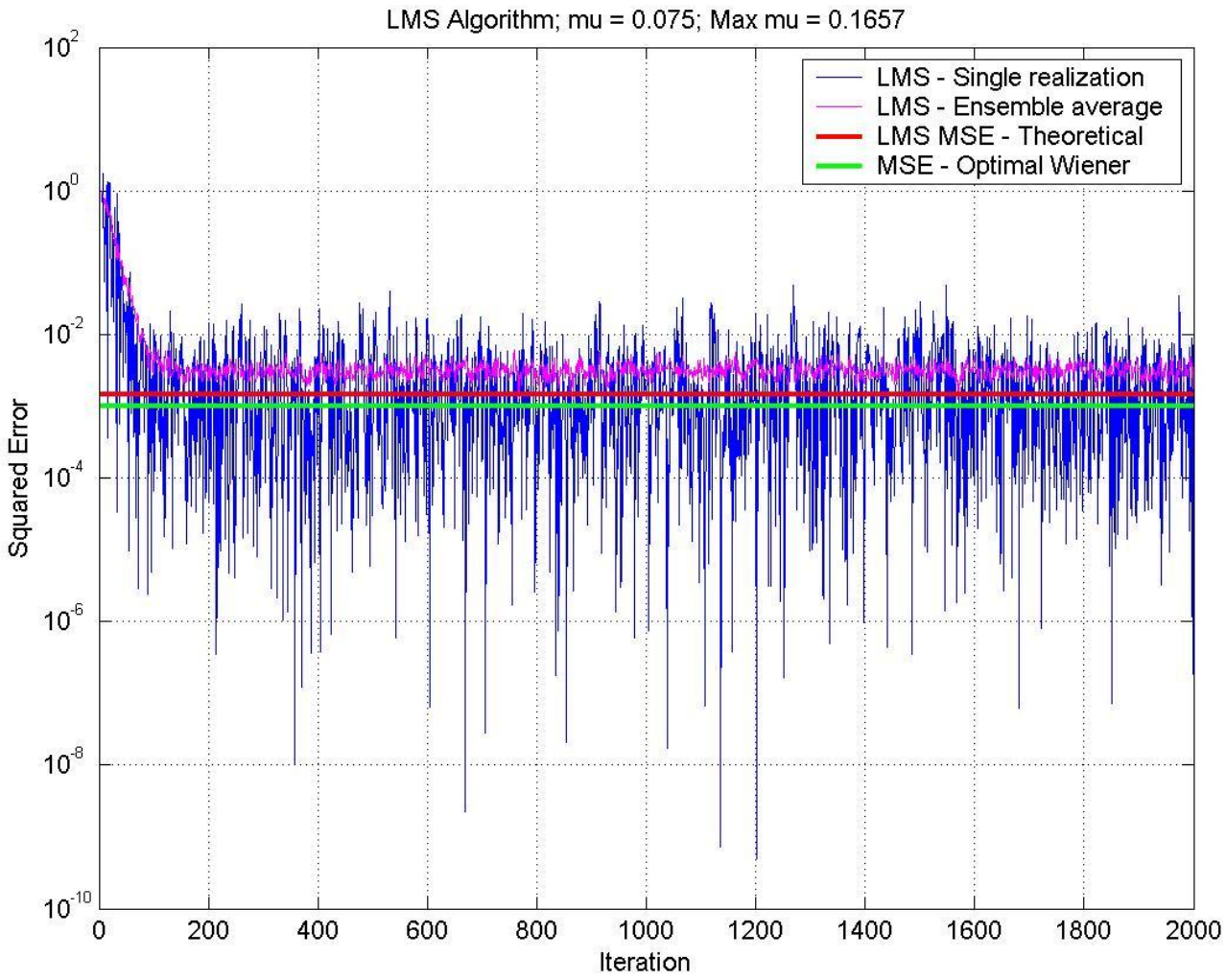
8. LAF: Example of Adaptive Equalizer (2)

Characterize the performance of the LMS algorithm when the impulse response of the channel is given by the raised cosine, $W = 2.9$:

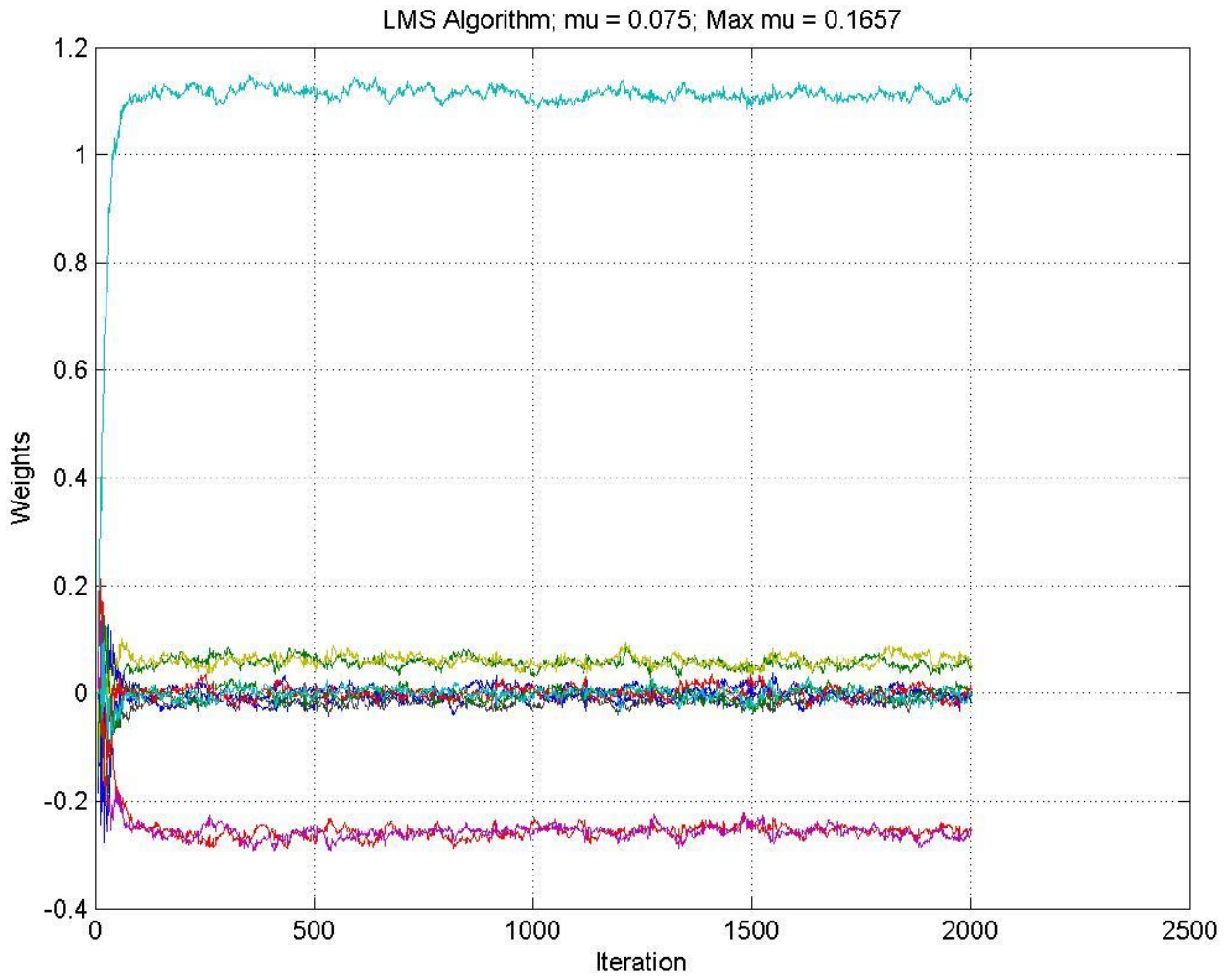
$$h(n) = \begin{cases} 0.5\left\{1 + \cos\left[\dfrac{2\pi}{W}(n-2)\right]\right\} & n = 1, 2, 3 \\ 0 & \text{otherwise} \end{cases}$$

See p. 412, [1]

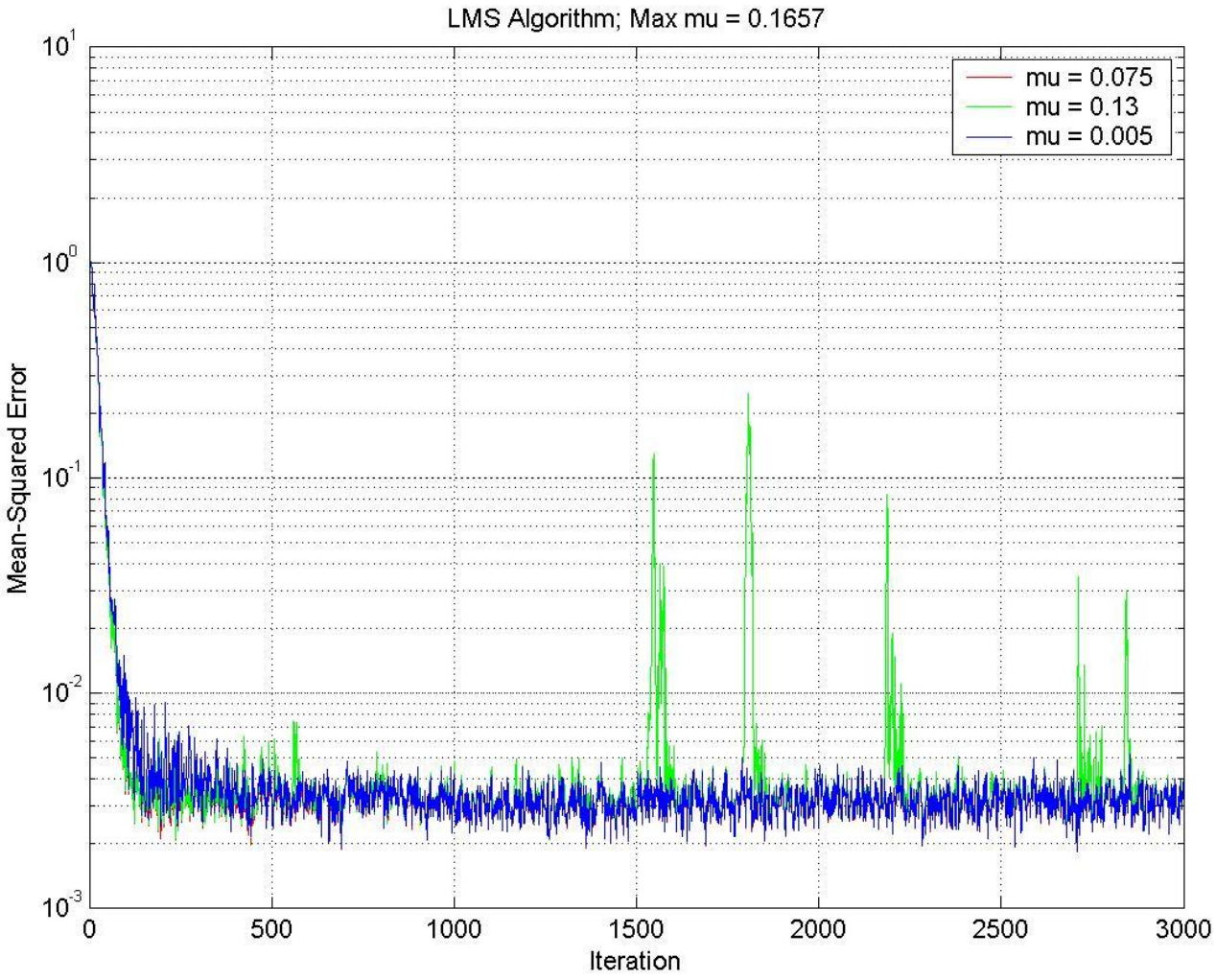
8. LAF: Example of Adaptive Equalizer (3)



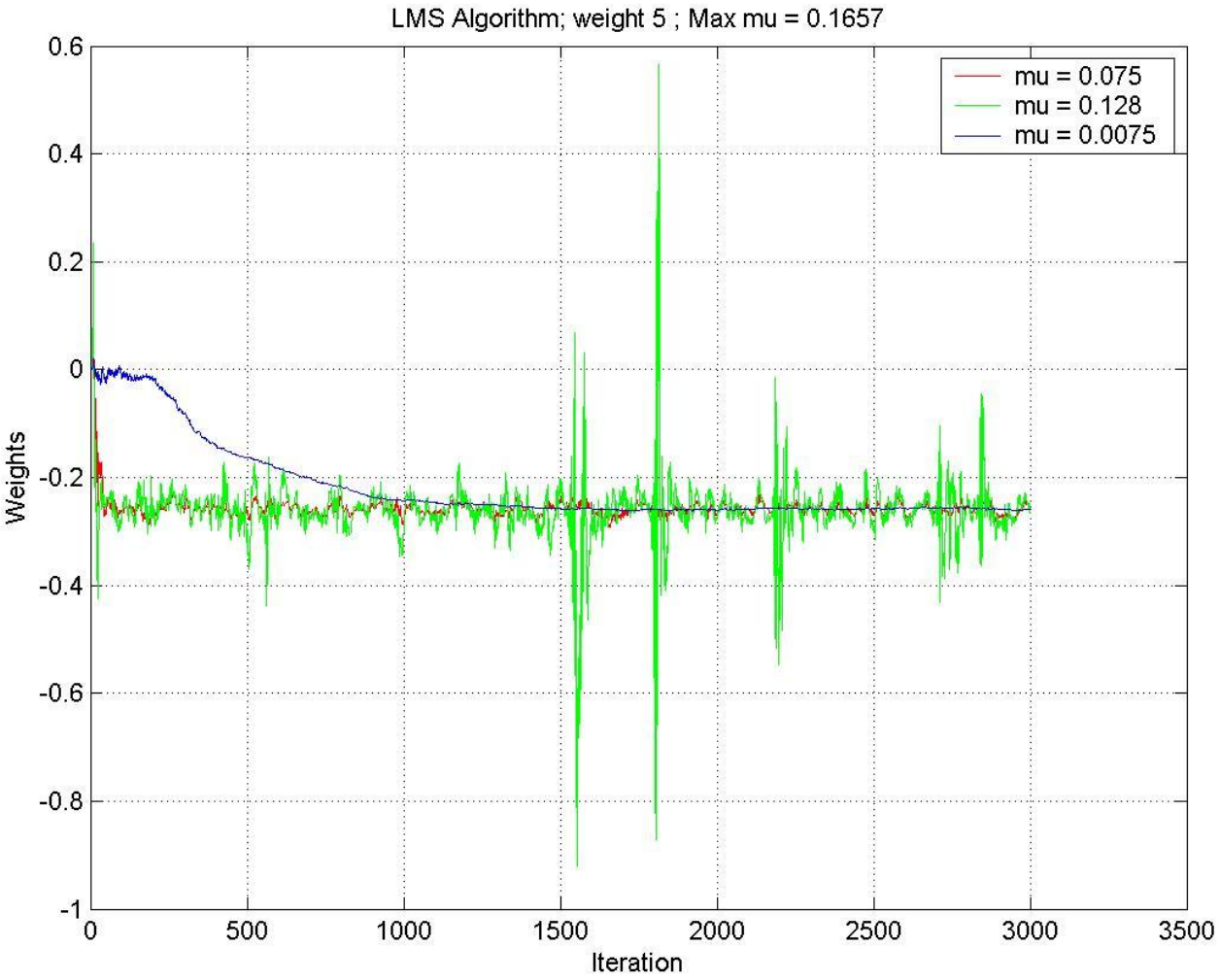
8. LAF: Example of Adaptive Equalizer (4)



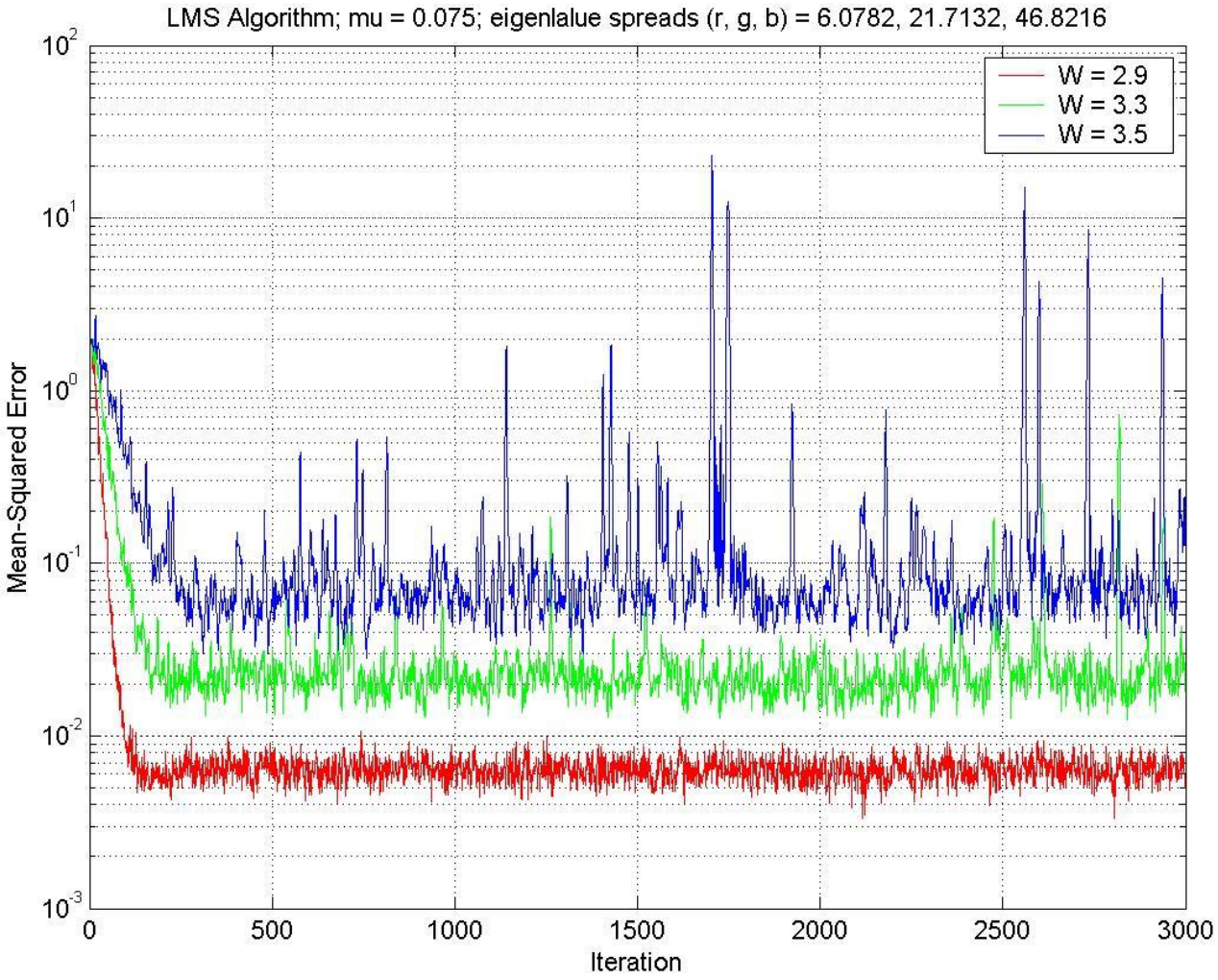
8. LAF: Example of Adaptive Equalizer (5)



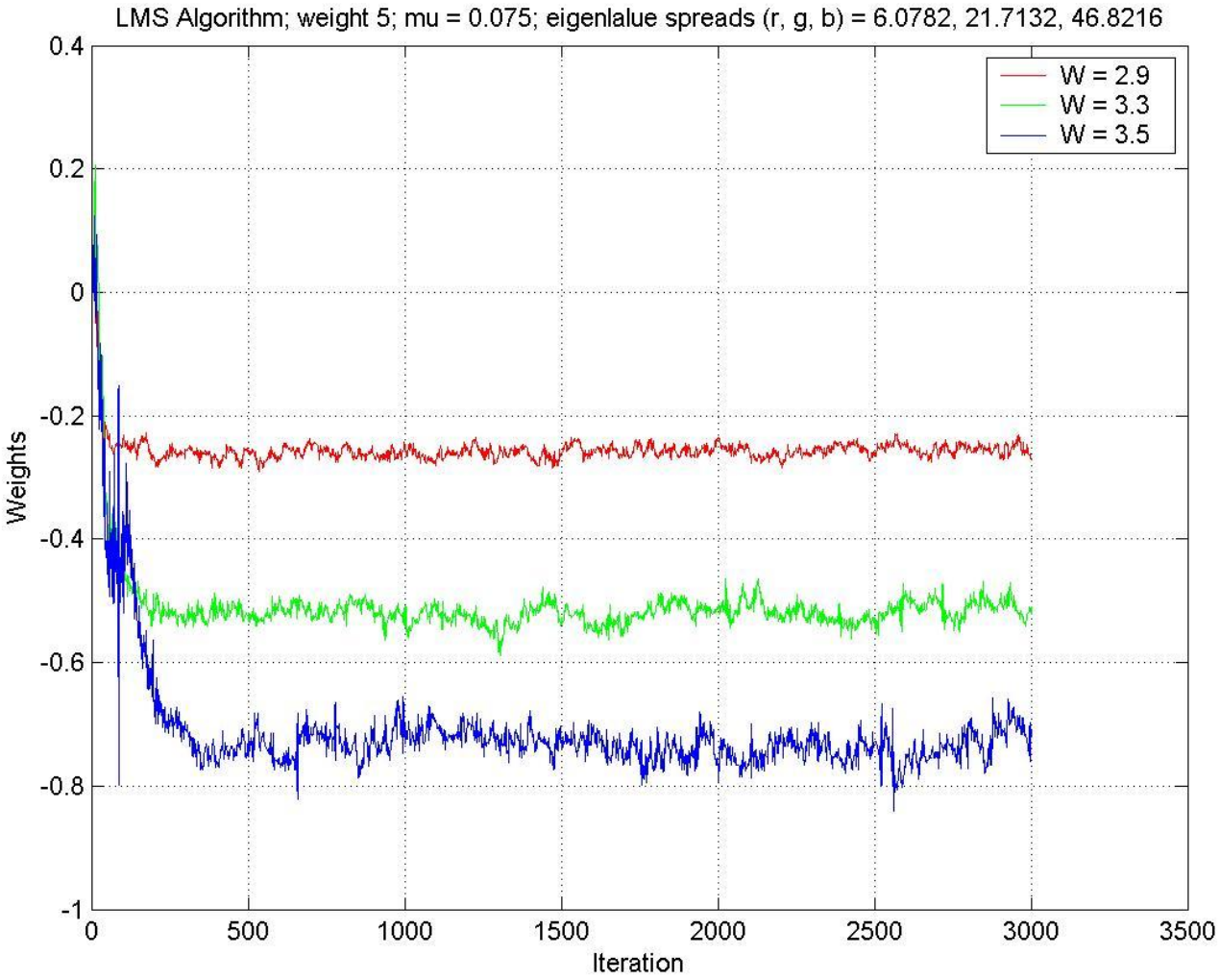
8. LAF: Example of Adaptive Equalizer (6)



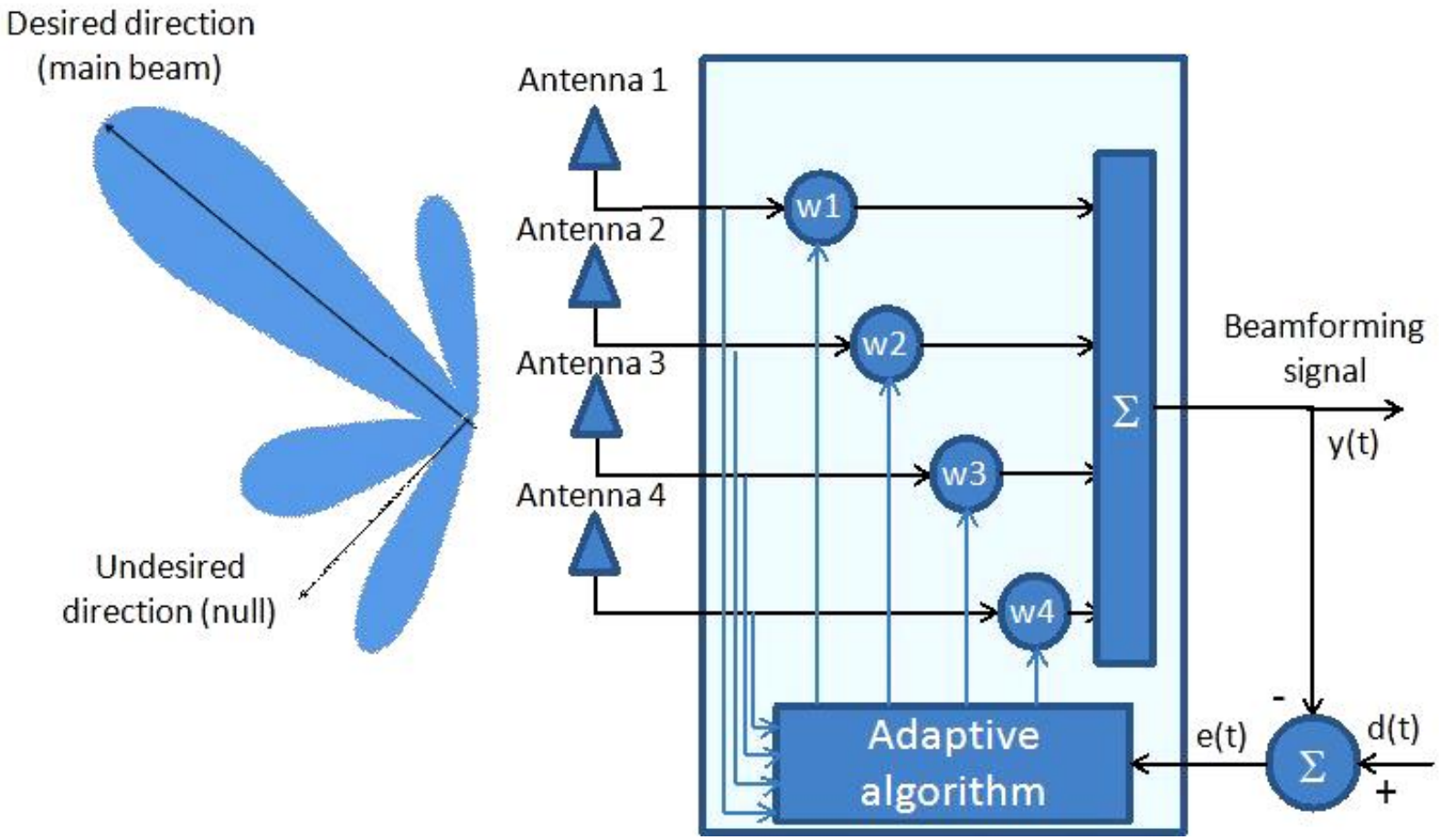
8. LAF: Example of Adaptive Equalizer (7)



8. LAF: Example of Adaptive Equalizer (8)



8. LAF: Example of Adaptive Beamformer (1)



8. LAF: Example of Adaptive Beamformer (2)

The fixed beamforming approaches, mentioned in chapter 5, which included the maximum SIR, the MSE method, and the MV method were assumed to apply to fixed arrival angle emitters. If the arrival angles don't change with time, the optimum array weights won't need to be adjusted. However, if the desired arrival angles change with time, it is necessary to devise an optimization scheme that operates on-the-fly so as to keep recalculating the optimum array weights. The receiver signal processing algorithm then must allow for the continuous adaptation to an ever-changing electromagnetic environment. The **adaptive beamforming** (with adaptive algorithms) takes the fixed beamforming process one step further and **allows for the calculation of continuously updated weights**. The adaptation process must satisfy a specified optimization criterion.

Several examples of popular optimization techniques include LMS, SMI, recursive least squares (RLS), the constant modulus algorithm (CMA), conjugate gradient, etc.

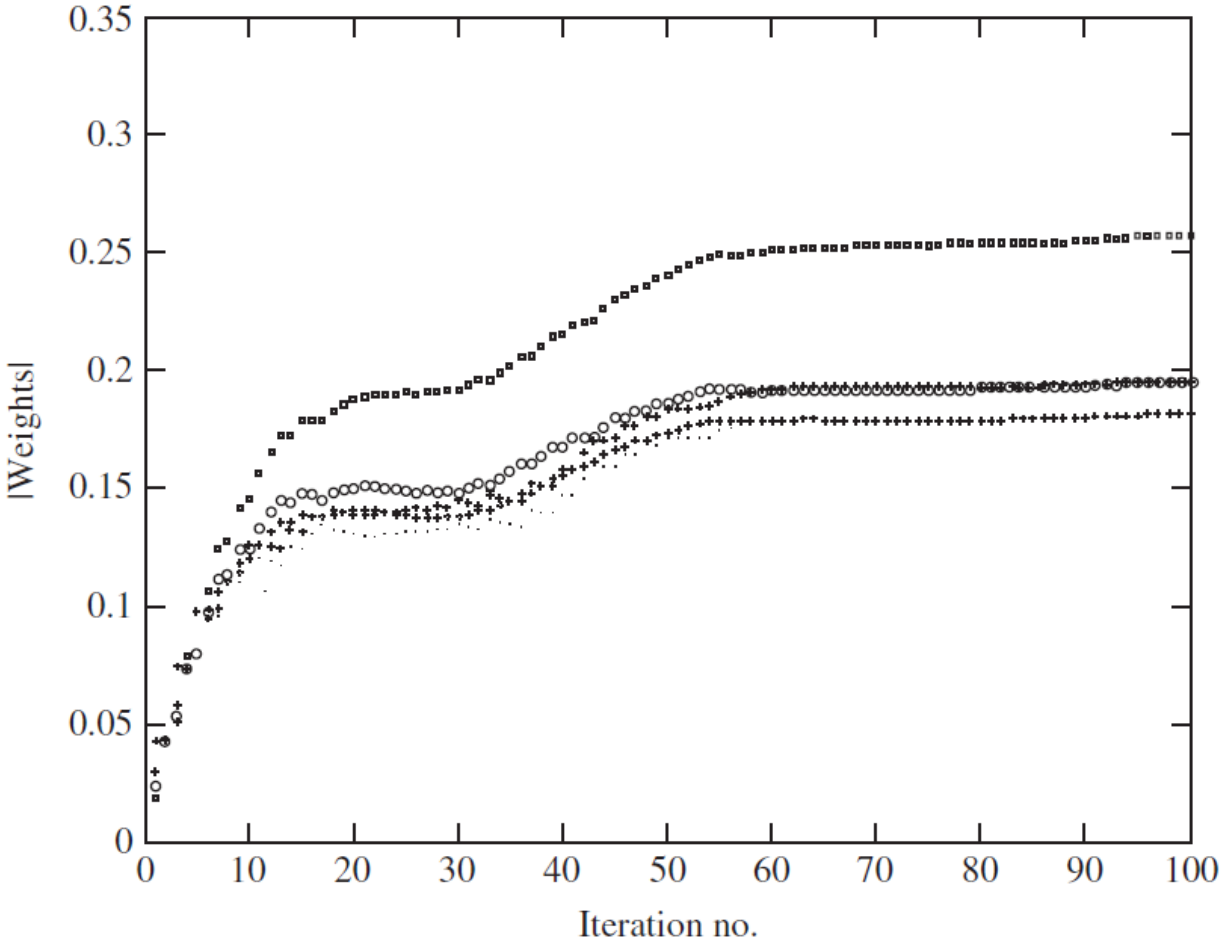
8. LAF: Example of Adaptive Beamformer (3)

Example: Adaptive beamforming using LMS algorithm:

An $M = 8$-element array with spacing $d = .5\lambda$ has a received signal arriving at the angle $\theta_0 = 30\circ$, an interferer at $\theta_1 = -60\circ$. Use MATLAB to write an LMS routine to solve for the desired weights. Assume that the desired received signal vector is defined by $\mathbf{x}_s(k) = \mathbf{a}_0 s(k)$, where $s(k) = \cos(2\pi\, t(k)/T)$; with $T = 1$ ms and $t = (1{:}100)T/100$. Assume the interfering signal vector is defined by $\mathbf{x}_i(k) = \mathbf{a}_1 i(k)$, where $i(k) = \text{randn}(1,100)$. Both signals are nearly orthogonal over the time interval T. Let the desired signal $d(k) = s(k)$. Assumed that the step size $\mu = .02$.

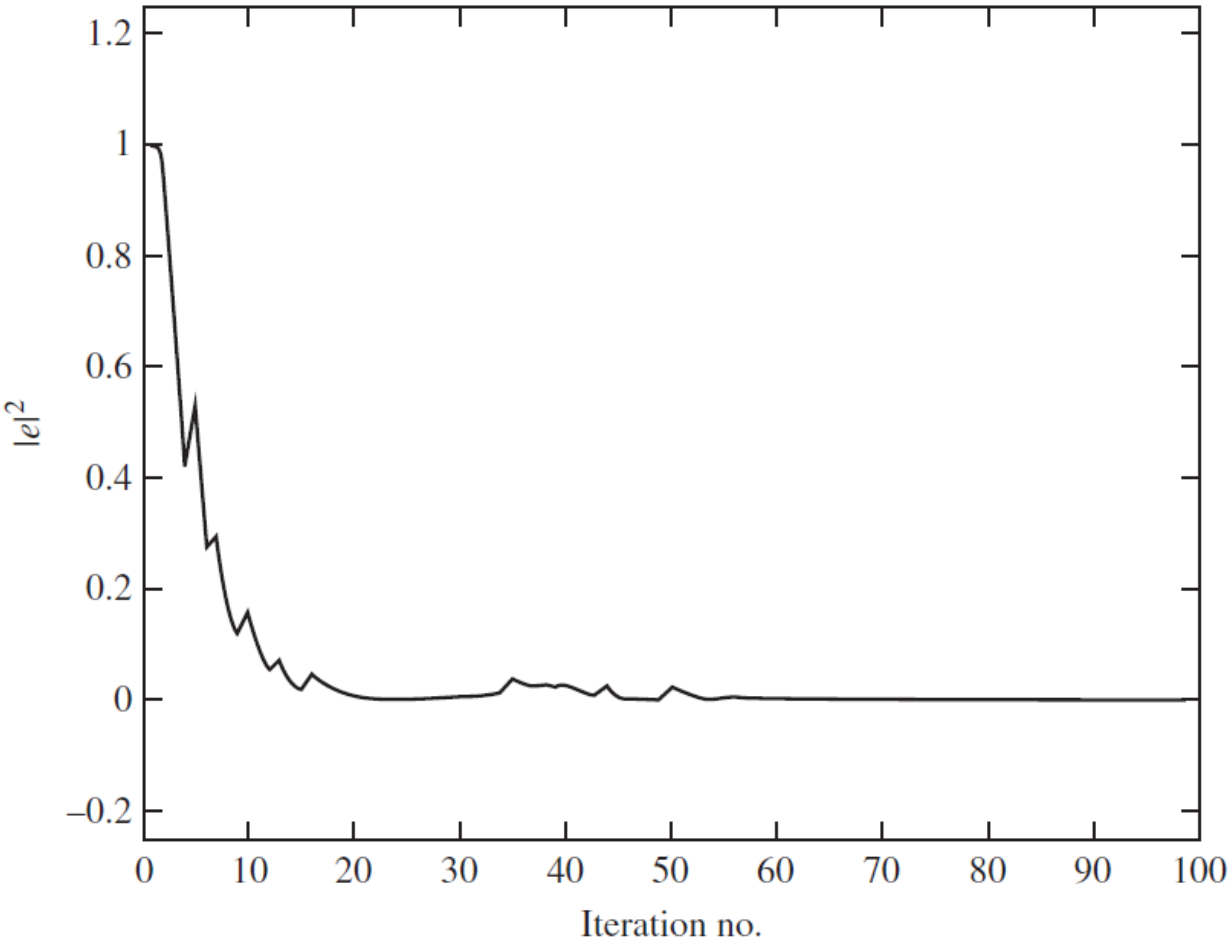
8. LAF: Example of Adaptive Beamformer (4)

Magnitude of array weights (convergence after about 60 iterations):



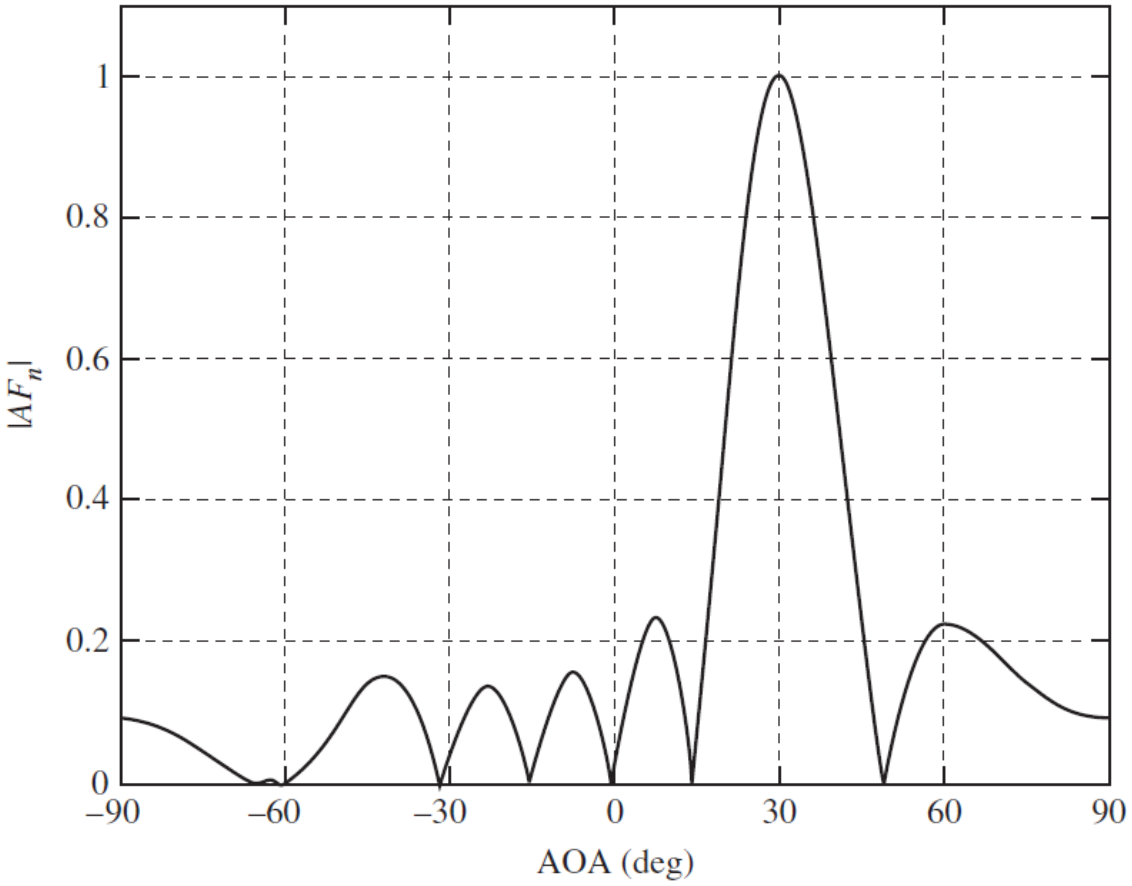
8. LAF: Example of Adaptive Beamformer (5)

Mean-square error (convergence after about 60 iterations):



8. LAF: Example of Adaptive Beamformer (6)

Final array factor (after convergence) which has a peak at the desired direction of 30◦ and a null at the interfering direction of −60◦:



8. LAF: Example of Adaptive Beamformer (7)

The array output acquires and tracks the desired signal after about 60 iterations: