Adaptive Signal Processing

Jose Krause Perin

Stanford University

August 6, 2018

Announcements

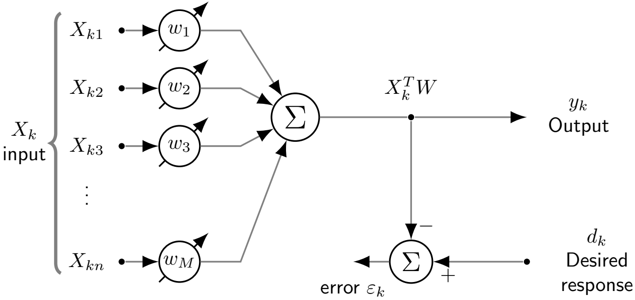
- Thank you for the feedback on the teaching evaluation
- Logistics for next lectures

Date	Lecture	Lecture Topic	Reading
6-Aug	10	Adaptive signal processing	
8-Aug	11	The discrete Fourier transform (DFT)	8.1–8.7
13-Aug	12	Spectrum analysis with the DFT	10.1–10.6
15-Aug	13	Review and conclusions	

- HW6 will be assigned today and covers primarily lectures 10 and 11
- You can choose to take the final exam on either 8/16 or 8/17

Outline

The adaptive linear combiner



We would like to *tune* the weights $w_1, \ldots, w_M$ to minimize the error ϵ_k according to some performance metric.
In this lecture we will address the following questions:

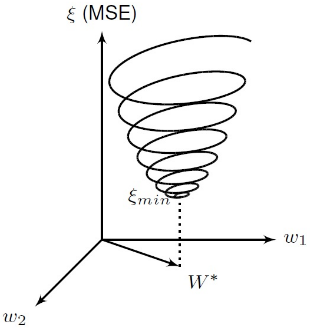
1. What is the best set of weights $w_1^\star, \ldots, w_M^\star$?
2. How to *adapt* the weights $w_1, \ldots, w_M$ to approximately achieve $w_1^\star, \ldots, w_M^\star$?

The mean square error

The **mean-square error (MSE)** is a convenient and sensible performance metric

$$\begin{aligned} \mathrm{MSE} = \xi &= \mathbb{E}(\varepsilon_k^2) && \text{(by definition)} \\ &= \mathbb{E}((d_k - X_k^T W)^2) && \text{(as } \varepsilon_k = d_k - X_k^T W) \\ &= \mathbb{E}(d_k^2) - 2\,\mathbb{E}(d_k X_k^T) W + W^T\,\mathbb{E}(X_k X_k^T) W \end{aligned}$$

The MSE is a **quadratic function** of the weights. There's only one minimum.



The performance surface

$$\text{MSE} = \xi = \mathbb{E}(d_k^2) - 2\,\mathbb{E}(d_k X_k^T)W + W^T\,\mathbb{E}(X_k X_k^T)W$$

To ease the notation we make a few definitions

$$P \equiv \mathbb{E}(d_k X_k) \qquad \text{(Cross-correlation between input and desired response)}$$

$$R \equiv \mathbb{E}(X_k X_k^T) \qquad \text{(Autocorrelation matrix)}$$

More explicitly:

$$P = \begin{bmatrix} \mathbb{E}(d_k x_{1k}) \\ \mathbb{E}(d_k x_{2k}) \\ \vdots \\ \mathbb{E}(d_k x_{Mk}) \end{bmatrix} \qquad R = \begin{bmatrix} \mathbb{E}(x_{1k}x_{1k}) & \mathbb{E}(x_{1k}x_{2k}) & \dots & \mathbb{E}(x_{1k}x_{nk}) \\ \mathbb{E}(x_{2k}x_{1k}) & \mathbb{E}(x_{2k}x_{2k}) & \dots & \mathbb{E}(x_{2k}x_{nk}) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbb{E}(x_{nk}x_{1k}) & \mathbb{E}(x_{nk}x_{2k}) & \dots & \mathbb{E}(x_{nk}x_{nk}) \end{bmatrix}$$

Substituting $P = \mathbb{E}(d_k X_k)$ and $R = \mathbb{E}(X_k X_k^T)$ in our equation for the MSE results in

$$\mathrm{MSE} = \xi = \mathbb{E}(d_k^2) - 2P^T W + W^T R W$$

The minimum of the quadratic function is where the first derivative is zero:

$$\frac{d}{dW}\mathrm{MSE} = 0 \implies W^\star = R^{-1}P \qquad \text{(Wiener solution)}$$

The **Wiener solution** is the set of weights $W^\star$ that minimize the MSE.

At the **Wiener solution**, the MSE is minimum

$$\mathrm{MSE}|_{W=W^\star} = \xi_{min} = E(d_k^2) - P^T R^{-1} P \qquad \text{(Minimum MSE)}$$

The orthogonality principle

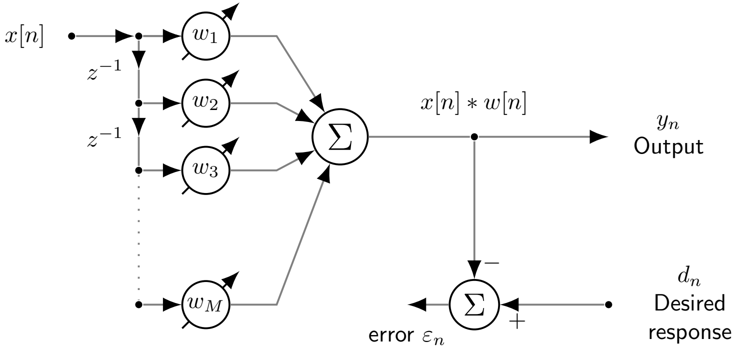
At the Wiener solution, the error is **orthogonal** to the input.
In statistical terms, the error is **uncorrelated** to the input.

$$\mathbb{E}(\varepsilon_k X_k)\Big|_{W=W^\star} = 0 \qquad \text{(orthogonality principle)}$$

This property is known as the **orthogonality principle**, and it is true for all least-squares solution.

An FIR filter as a linear combiner

The input to the adaptive linear combiner is $X = [x[n], x[n-1], \ldots, x[n-M+1]]^T$



The resulting FIR filter has coefficients $\{w_1, \ldots, w_M\}$.

The optimal filter coefficients are given by the Wiener solution

$$W^\star = R^{-1}P$$

And we can compute the vector P and matrix R as we did before

$$P = \begin{bmatrix} \mathbb{E}(d[n]x[n]) \\ \mathbb{E}(d[n]x[n-1]) \\ \vdots \\ \mathbb{E}(d[n]x_[n-N]) \end{bmatrix}$$

$$R = \begin{bmatrix} \mathbb{E}(x[n]x[n]) & \mathbb{E}(x[n]x[n-1]) & \dots & \mathbb{E}(x[n]x[n-N]) \\ \mathbb{E}(x[n-1]x[n]) & \mathbb{E}(x[n-1]x[n-1]) & \dots & \mathbb{E}(x[n-1]x[n-N]) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbb{E}(x[n-N]x[n]) & \mathbb{E}(x[n-N]x[n-1]) & \dots & \mathbb{E}(x[n-N]x[n-N]) \end{bmatrix}$$

Recall that the autocorrelation function is defined by

$$\phi_{xx}[m] = \mathbb{E}(x[n+m]x^*[n])$$

Now we're just writing it in matrix form

$$R = \begin{bmatrix} \phi_{xx}[0] & \phi_{xx}[1] & \dots & \phi_{xx}[N] \\ \phi_{xx}[1] & \phi_{xx}[0] & \dots & \phi_{xx}[N-1] \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{xx}[N] & \phi_{xx}[N-1] & \dots & \phi_{xx}[0] \end{bmatrix}$$

In Matlab:

```
>> phi_xx = xcorr(x, x, N)
>> R = toeplitz(phi_xx(N+1:end))
```

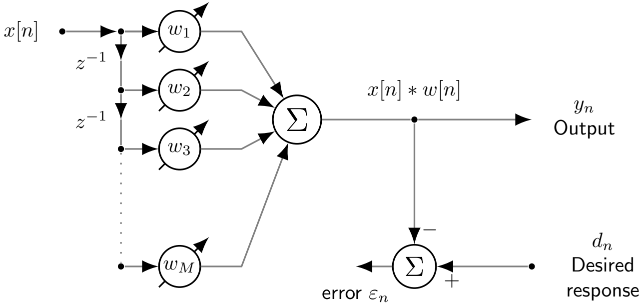
Theory

- The adaptive linear combiner and its performance surface
- Adaptation algorithms

Application examples

- Linear equalization
- Noise canceling
- Inverse control

Adaptation algorithms



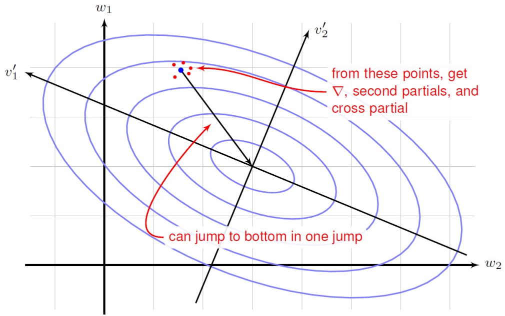
Problem: We know that $W^\star = R^{-1}P$, but in practice, R and P are either unknown or hard to compute/estimate. How can we find some W such that $W \approx W^\star$?

1. Newton's method
2. Steepest descent
3. The **least-mean squares (LMS)** algorithm

Adaptation algorithms: Newton's method

$$W \leftarrow W - \mu R^{-1} \nabla \qquad \text{(adaptation equation)}$$

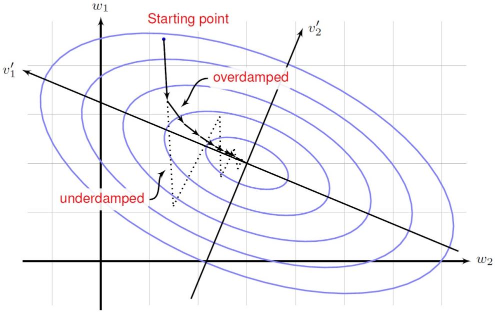
where μ is the adaptation constant, and ∇ is the gradient of the MSE, i.e., $\nabla \equiv \frac{\partial}{\partial W}\mathrm{MSE}$.



R^{-1} scales and directs the step to $W^{\star}$. Convergence happens in one step!
Problem: requires knowledge of R and ∇.

Adaptation algorithms: steepest descent

$$W \leftarrow W - \mu\nabla \qquad \text{(adaptation equation)}$$



- W is adapted in the direction of **steepest descent** of the gradient
- Must estimate the gradient ∇ somehow.

Adaptation algorithms: the LMS algorithm

The **least mean squares (LMS)** algorithm is a form of steepest descent where the gradient is estimated from the **instantaneous error**

$$\varepsilon_n^2 = (d_n - X_n^T W)^2 \qquad \text{(instantaneous error)}$$

$$\hat{\nabla} = \frac{\partial \varepsilon_n}{\partial W} = 2(d_n - X_n^T W)X_n = 2\varepsilon_n X_n \qquad \text{(gradient estimate)}$$

$$W \leftarrow W + 2\mu e_n X_n \qquad \text{(LMS weight update)}$$

Gradient estimate is very noisy, but on average the weights *generally* move to the Wiener solution.

Adaptation algorithms: the LMS algorithm

The LMS algorithm:

Initialize the weights to some value $W \leftarrow W_0$

For each input and desired response pair (X_n, d_n):

Compute the output $y_n = X_n^T W$

Compute the instantaneous error $e_n = (d_n - y_n)$

Update the weights: $W \leftarrow W + 2\mu e_n X_n$

end

If X is complex, then the adaptation equation changes slightly $W \leftarrow W + 2\mu e_n X_n^*$.

Learning curve

- The learning curve is a plot of the average MSE $\mathbb{E}(e_n^2)$ over time.
- To obtain an empirical learning curve, we run the LMS algorithm N times with different weight initializations.
- For each run, we obtain a MSE curve $\mathrm{MSE}^{(1)}, \ldots, \mathrm{MSE}^{(N)}$ i.e., $\mathrm{MSE}^{(i)} = e_n^2$.
- Then we average the result

$$\mathbb{E}(\xi_k^2) \approx \frac{\mathrm{MSE}^{(1)} + \ldots + \mathrm{MSE}^{(N)}}{N} \tag{1}$$

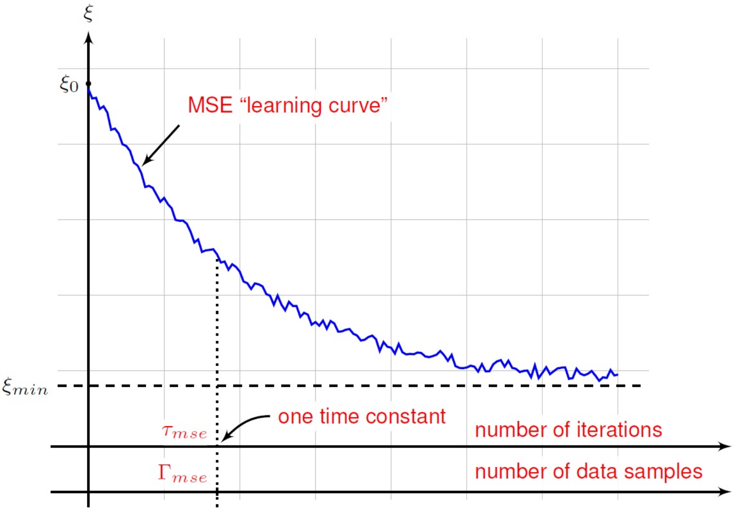
- The learning curve is a sum of decaying exponentials with time constants

$$(\tau_{MSE})_n \approx \frac{1}{4\mu\lambda_n} \text{ iterations} \qquad \text{(Steepest descent \& LMS)}$$

where λ_n is the nth eigenvalue of matrix R.

Learning curve

Example of learning curve



Stability of the LMS algorithm

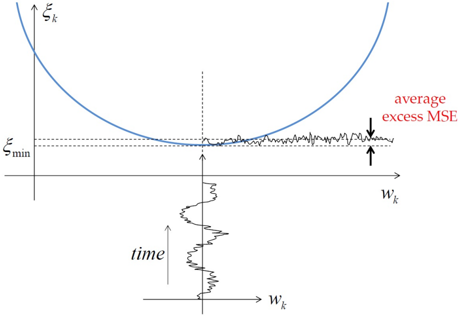
- The constant $\mu > 0$ is known as the **adaptation constant**.
- If μ is too small, the algorithm will take too long to converge (small steps).
- If μ is too large, the algorithm can become unstable.
- It can be shown that stability is guarantee if

$$0 < \mu < \frac{1}{\mathrm{trace} R} \qquad \text{(stability condition)}$$

  where trace is the sum of all elements in the main diagonal of a matrix.
- When applying the LMS algorithm to determine the coefficients of an Mth-order FIR filter we have that $\mathrm{trace} R = (M+1)\phi_{xx}[0]$.

Excess noise and misadjustment

Error in the gradient estimate leads to excess MSE



$$\text{Misadjustment} = \frac{\text{excess noise}}{\text{minimum MSE}} \qquad \text{(definition)}$$

For the LMS algorithm:

$$M = \mu \text{trace}(R)$$

Performance metrics

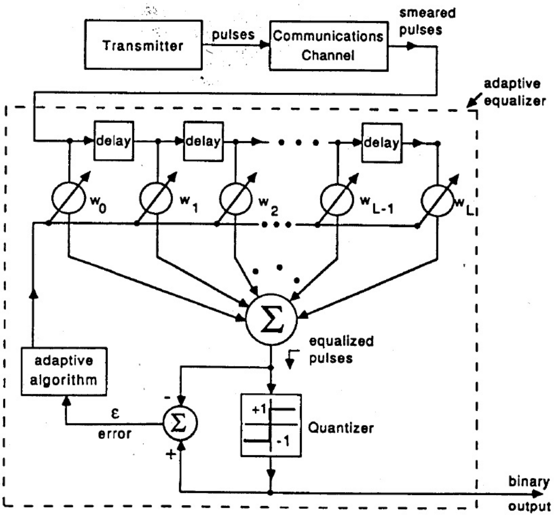
Parameters to tune

- Number of weights
- Adaptation constant μ

What to look for

- Minimum MSE: is the number of weights high enough?
- Time constants: how fast will the adaptive algorithm reach the minimum MSE?
- Excess MSE or misadjustment: how oscillatory are the solutions produced by the adaptive algorithm near the optimal solution?

Linear equalization with decision-directed learning



Eye diagram before and after equalization

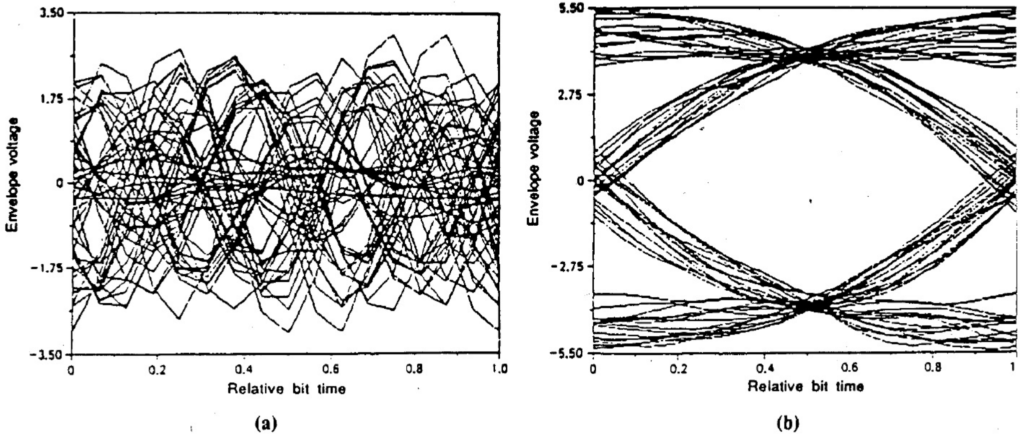
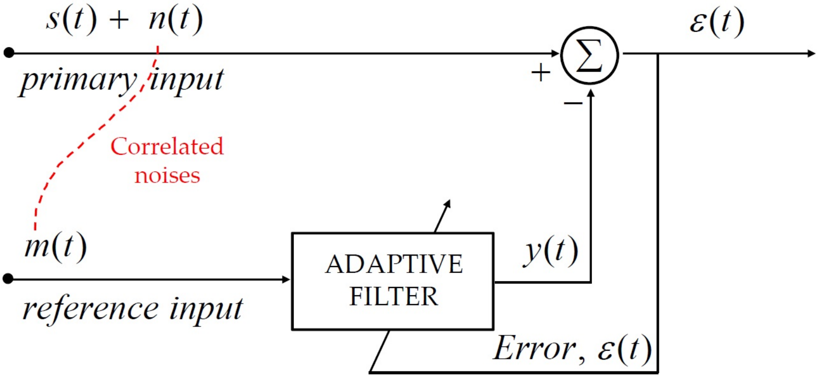


Figure 1: (a) Before and (b) after equalization.

Noise canceling



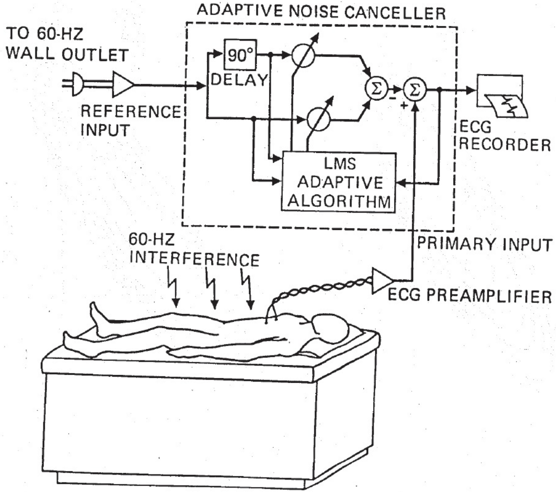
$$\varepsilon = s + n - y \qquad \text{(error)}$$

$$\mathbb{E}(\varepsilon^2) = \mathbb{E}((s + n - y)^2) \qquad \text{(Mean square error)}$$

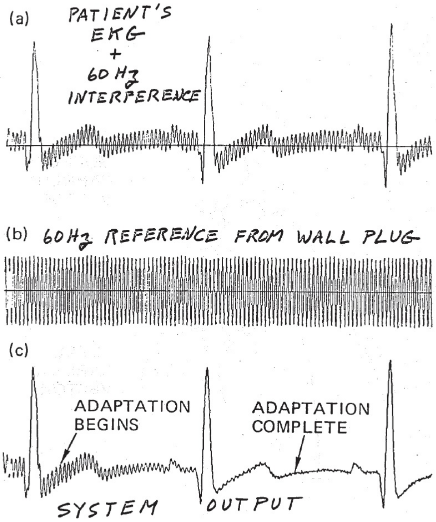
$$= \mathbb{E}(s^2) + \mathbb{E}((n - y)^2) \qquad (\mathbb{E}(s(n - y)) = 0 \text{ from assumption})$$

Examples of noise canceling applications

Canceling 60Hz interference from biological signals.

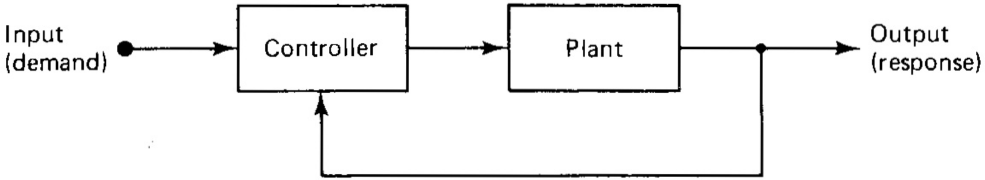


Examples of noise canceling applications



Inverse control

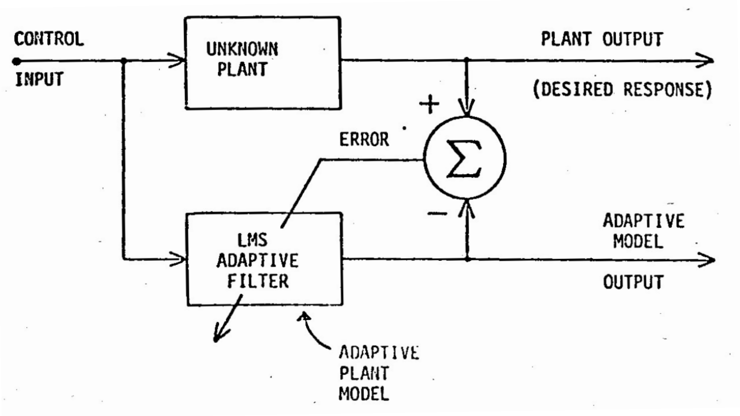
- In control problems we would like to make a plant (a system) respond to a given command and produce a desired output (e.g., setting the room temperature, controlling the blood pressure of a patient, etc)



- We use adaptive filters for two basic operations in control problems
  - Plant identification
  - Plant inversion

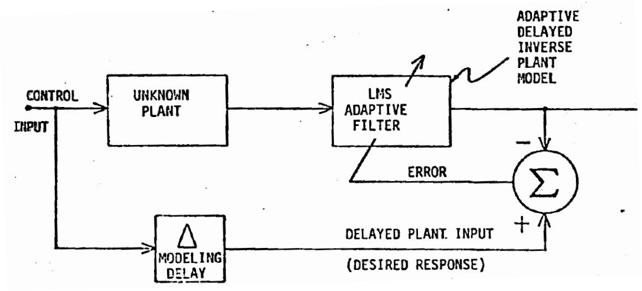
Basic operations: plant identification

The adaptive filter models the plant



Basic operations: plant inverse

The adaptive filter models the inverse of the plant



We need to consider what happens if the plant is **non-minimum phase**. That is, if it has zeros outside the unit circle (unstable inverse).

- For any discrete-time system $H(z)$, we can write $H(z) = H_{min}(z)H_{ap}(z)$, where $H_{min}(z)$ is a minimum phase system, and $H_{ap}(z)$ is an all-pass system.
- The adaptive filter will converge to $H_{min}^{-1}(z)$, and it'll not compensate for phase distortion (and delay) due to $H_{ap}(z)$.

Summary

- The linear combiner is the basis of adaptive systems and adaptive filtering
- We use the mean square error (MSE) as the performance metric
- The Wiener solution is the optimal set of weights that minimizes the MSE
- The LMS algorithm is a simple way to train the adaptive filter to approximate the Wiener solution
- The LMS algorithm uses the instantaneous error to obtain an estimate of the gradient
- This estimate is very noisy, but on average it converges to the Wiener solution
- We adjust the adaption constant to control how fast the LMS algorithm converges and how noisy the solutions near the Wiener solution (excess noise and misadjustment)