3. Optimal and Adaptive Filtering

3.1. Wiener-Hopf filter

3.1.1. Introduction

Several **estimation problems** can be modeled relying on a similar formulation:

Given a set of data from an observed noisy process $x[n]$ and a desired target process $d[n]$ that we want to estimate, produce an estimation $y[n]$ of the target process by linear time-invariant filtering ($T[n] = h[n]$) of the observed samples.

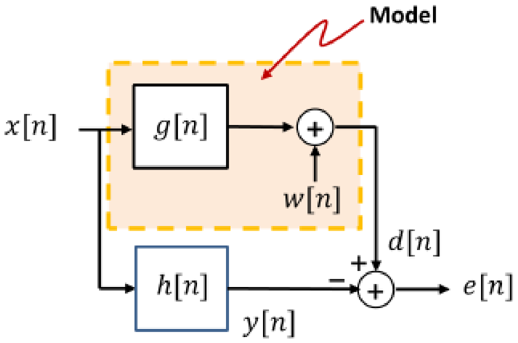
We assume **known stationary signal and noise spectra** (**correlation**), as well as **additive noise**. We will first assume Finite Impulse Response (FIR) filters, and afterwards we will delve into non-stationary scenarios.

Filter configuration

This formulation can be applied to a large family of problems that are commonly sorted into four wide classes:

- **System identification**

We want to **identify a given system**, that can be real or some abstraction. We model this system as an LTI system plus an additive noise source $w[n]$.

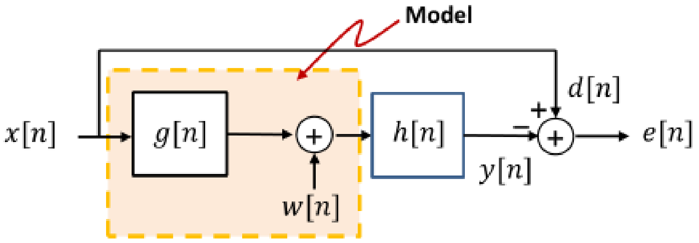


Design and use: we excite the system with a known signal $x[n]$ and obtain the filter that models the system.

The application assumes a noisy reference and noise-free observations.

- **System inversion**

We want to **estimate a system and apply its inverse to the signal**. We model this system as an LTI system plus an additive noise source $w[n]$.

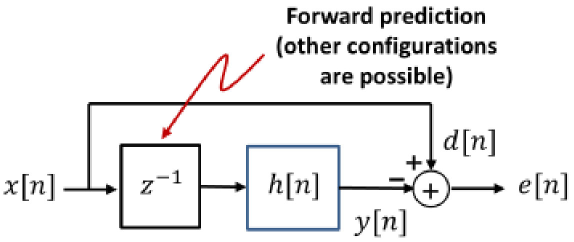


Design: we excite the system with a known signal $x[n]$ and obtain the filter that models the system.
Use: the filter is concatenated to the system to recover the estimated signal.

The application assumes noisy observations and a noise-free reference.

- **Signal prediction**

We estimate the value of a random signal at a given time instance $x[n_0]$, based on other time instance values ($x[n_0-1], x[n_0-2], \ldots$).

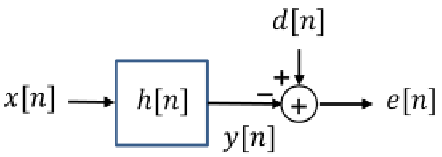


Design: we compare the current signal value $x[n_0]$ with its estimation $y[n_0]$.
Use: The current signal value $x[n_0]$ may not be available and we produce an estimation. If $x[n_0]$ is available, we produce the prediction error $e[n_0]$.

The application assumes that observations and reference belong to the same noisy process.

- **Signal cancellation**

We estimate the value of a primary signal which contains an interference. This interference has been isolated through other sensors in additional signals.



Design: we compare the primary signal $d[n]$ with the interference $x[n]$.
Use: we obtain the clean signal as the estimation error $e[n]$.

The application assumes that the noisy interferences are the observations, while the noisy signal together with the interferences are the reference.

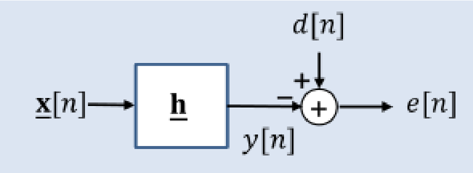
3.1.2. Minimum Mean Square Error (MMSE) prediction

Given the generic formulation, we restrict the analysis to the **FIR filter** case. It is the **optimal solution** if $x[n]$ and $d[n]$ are Gaussian jointly distributed processes. Then, the filter is assumed to have a finite number of coefficients N. We use the **MSE** as an optimization criterion because it is mathematically treatable, leads to useful solutions and can be used as a benchmark for other solutions.

We will use the following notation:

$$h[n] * x[n] = \underline{h}^T \underline{x}[n], \quad \underline{x}[n] = \begin{bmatrix} x[n] \\ x[n-1] \\ \dots \\ x[n-N+1] \end{bmatrix}.$$

So, say we have this generic signal case:



Then, the error is

$$e[n] = d[n] - y[n] = d[n] - \underline{h}^T \underline{x}[n],$$

and we want to minimize it:

$$\min_{\underline{h}} \mathbb{E}\left[e^2[n]\right] = \min_{\underline{h}} \mathbb{E}\left[\left(d[n] - \underline{h}^T \underline{x}[n]\right)^2\right].$$

In order to do this, we will first prove what is called the **principle of orthogonality**: if the MSE is minimum, then it holds that

$$\mathbb{E}\left[e[n]\underline{x}[n]\right] = \underline{0}.$$

If the MSE is minimum with respect to the filter, we know that the gradient is $\underline{0}$. Then, if we develop this mathematically,

$$\nabla_{\underline{h}} \mathbb{E}\left[e^2[n]\right] = \mathbb{E}\left[\nabla_{\underline{h}}\left(d[n] - \underline{h}^T \underline{x}[n]\right)^2\right] = \mathbb{E}\left[-2\left(d[n] - \underline{h}^T \underline{x}[n]\right)\underline{x}[n]\right] = \underline{0} \iff -2\mathbb{E}\left[e[n]\underline{x}[n]\right] = \underline{0}.$$

So, we know that **the error is orthogonal to the observations**.

Now we want to develop some useful results of the MMSE prediction in a specific signal scenario:

- The observation process $x[n]$ can be split into two parts, $x[n] = a[n] + b[n]$.
- The reference process $d[n]$ can be split into two parts, $d[n] = a'[n] + c[n]$.

These parts of each process have the following correlation properties:

- $r_{ab}[l] = \mathbb{E}\left[a[n+l]b[n]\right] = 0$
- $r_{a'c}[l] = \mathbb{E}\left[a'[n+l]c[n]\right] = 0$
- $r_{aa'}[l] = \mathbb{E}\left[a[n+l]a'[n]\right] \neq 0$
- $r_{ac}[l] = \mathbb{E}\left[a[n+l]c[n]\right] = 0$
- $r_{a'b}[l] = \mathbb{E}\left[a'[n+l]b[n]\right] = 0$
- $r_{bc}[l] = \mathbb{E}\left[b[n+l]c[n]\right] = 0$

When using the filter that minimizes the MSE, $\underline{h}_{opt}$, the following properties hold:

1. At any point in time, **the signal estimation and the error signal are not correlated**:

$$\mathbb{E}\left[e[n]y[n]\right] = \left[y[n] = \underline{h}_{opt}^T \underline{x}[n]\right] = \mathbb{E}\left[e[n]\underline{h}_{opt}^T \underline{x}[n]\right] = \underline{h}_{opt}^T \mathbb{E}\left[e[n]\underline{x}[n]\right] = \underline{0}.$$

2. The variance of the reference signal is greater or equal than the variance of the error signal:

$$\begin{aligned}\mathbb{E}\left[d^2[n]\right] = [d[n] = y[n] + e[n]] = \mathbb{E}\left[(y[n]+e[n])^2\right] &= \\ \mathbb{E}\left[d^2[n]\right] + 2\mathbb{E}\left[y[n]e[n]\right] + \mathbb{E}\left[e^2[n]\right] = [\text{we are using the optimal filter}] &= \\ \mathbb{E}\left[d^2[n]\right] + \mathbb{E}\left[e^2[n]\right] \geq \mathbb{E}\left[e^2[n]\right]&.\end{aligned}$$

3. If the observation and the reference signals are not correlated, the variance of the estimation is zero:

$$\begin{aligned}\mathbb{E}\left[y^2[n]\right] = \mathbb{E}\left[y[n](d[n]-e[n])\right] = \mathbb{E}\left[y[n]d[n]\right] - \mathbb{E}\left[y[n]e[n]\right] &= \\ \mathbb{E}\left[y[n]d[n]\right] = \mathbb{E}\left[\underline{h}_{opt}^T\underline{x}[n]d[n]\right] = \underline{h}_{opt}^T\mathbb{E}\left[\underline{x}[n]d[n]\right] &= \\ \left[\mathbb{E}\left[\underline{x}[n]d[n]\right] = \underline{0}\right] = 0&.\end{aligned}$$

4. The minimum variance of the error signal is $\varepsilon = r_d[0] - \underline{h}_{opt}^T\underline{r}_{xd}$:

$$\begin{aligned}\varepsilon := \mathbb{E}\left[e^2[n]\right]\Big|_{\min} = \mathbb{E}\left[e[n](d[n]-y[n])\right] = \mathbb{E}\left[e[n]d[n]\right] - \mathbb{E}\left[e[n]y[n]\right] &= \\ \mathbb{E}\left[e[n]d[n]\right] = \mathbb{E}\left[\left(d[n]-\underline{h}_{opt}^T\underline{x}[n]\right)d[n]\right] = \mathbb{E}\left[d^2[n]\right] - \mathbb{E}\left[\underline{h}_{opt}^T\underline{x}[n]d[n]\right] &= \\ r_d[0] - \underline{h}_{opt}^T\mathbb{E}\left[\underline{x}[n]d[n]\right] &= \\ r_d[0] - \underline{h}_{opt}^T\underline{r}_{xd}&.\end{aligned}$$

We will analyze the previous properties for the signal setting we stated before:

(sic.)

3.1.3. The Wiener-Hopf filter

The Wiener-Hopf solution

So far, we have analyzed some properties of the optimal filter, but we are yet to obtain it:

$$\begin{gathered}\left.\begin{array}{c} e[n] = d[n] - \underline{h}^T\underline{x}[n] \\ \mathbb{E}\left[\underline{x}[n]e[n]\right] = \underline{0}\end{array}\right\} \implies \mathbb{E}\left[\underline{x}[n]\left(d[n]-\underline{h}^T\underline{x}[n]\right)\right] = \underline{0}. \\ \mathbb{E}\left[\underline{x}[n]\left(d[n]-\underline{h}^T\underline{x}[n]\right)\right] = \mathbb{E}\left[\underline{x}[n]d[n]\right] - \mathbb{E}\left[\underline{x}[n]\underline{h}^T\underline{x}[n]\right] = \underline{0} \iff \\ \mathbb{E}\left[\underline{x}[n]d[n]\right] - \mathbb{E}\left[\underline{x}[n]\underline{x}^T[n]\underline{h}\right] = \underline{r}_{xd}[0] - \mathbb{E}\left[\underline{x}[n]\underline{x}^T[n]\right]\underline{h} = \underline{0} \iff \\ \underline{r}_{xd}[0] - \underline{\underline{R}}_x[0]\underline{h} = \underline{0} \iff \boxed{\underline{h}_{opt} = \underline{\underline{R}}_x^{-1}\underline{r}_{xd}.}\end{gathered}$$

So, this is the optimal filter in the sense of MSE minimization, with the matrix and the vector involved being

$$\underline{r}_{xd} = \mathbb{E}\left[\underline{x}[n]d[n]\right] = \begin{bmatrix}\mathbb{E}\left[x[n]d[n]\right] \\ \mathbb{E}\left[x[n-1]d[n]\right] \\ \cdots \\ \mathbb{E}\left[x[n-N+1]d[n]\right]\end{bmatrix} = \begin{bmatrix} r_{xd}[0] \\ r_{xd}[-1] \\ \cdots \\ r_{xd}[-N+1]\end{bmatrix} : \text{the Cross-correlation vector}$$

$$\underline{\underline{R}}_x = \mathbb{E}\left[\underline{x}[n]\underline{x}^T[n]\right] = \begin{bmatrix} r_x[0] & r_x[1] & \cdots & r_x[N-1] \\ r_x[-1] & r_x[0] & \cdots & r_x[N-2] \\ \vdots & \vdots & \ddots & \vdots \\ r_x[-N+1] & r_x[-N+2] & \cdots & r_x[0]\end{bmatrix} : \text{the Correlation matrix}$$

The **optimal filter**, hence, depends on the **second order statistics** of the processes. We will analyze the properties of this correlation matrix, and also we will study what to do when such statistics are not available.

The error performance surface

The Wiener-Hopf filter is optimal at minimizing the MSE of the prediction; that is, the variance (power) of the error signal $e[n]$ (assuming it is zero-mean). We now want to study how does the MSE behave for an arbitrary filter. In order to do this, let us first develop a useful result: for any filter, the MSE can be expressed as

$$\mathbb{E}\left[e^2[n]\right] = \varepsilon + \left(\underline{h}_{opt} - \underline{h}\right)^T \underline{\underline{R}}_x \left(\underline{h}_{opt} - \underline{h}\right).$$

So, let's see this result:

$$\begin{aligned}
\mathbb{E}\left[e^2[n]\right] = \mathbb{E}\left[\left(d[n] - \underline{h}^T \underline{x}[n]\right)\left(d[n] - \underline{h}^T \underline{x}[n]\right)\right] &= \\
\mathbb{E}\left[d^2[n]\right] - 2\mathbb{E}\left[\underline{h}^T \underline{x}[n] d[n]\right] + \mathbb{E}\left[\underline{h}^T \underline{x}[n] \underline{h}^T \underline{x}[n]\right] &= \\
r_d[0] - 2\mathbb{E}\left[\underline{h}^T \underline{x}[n] d[n]\right] + \mathbb{E}\left[\underline{h}^T \underline{x}[n] \underline{x}^T[n] \underline{h}\right] &= \\
r_d[0] - 2\underline{h}^T \mathbb{E}\left[\underline{x}[n] d[n]\right] + \underline{h}^T \mathbb{E}\left[\underline{x}[n] \underline{x}^T[n]\right] \underline{h} &= \\
r_d[0] - 2\underline{h}^T \underline{r}_{xd} + \underline{h}^T \underline{\underline{R}}_x \underline{h} = \left[\varepsilon = r_d[0] - \underline{h}_{opt}^T \underline{r}_{xd}\right] &= \\
r_d[0] - \underline{h}_{opt}^T \underline{r}_{xd} + \underline{h}_{opt}^T \underline{r}_{xd} - 2\underline{h}^T \underline{r}_{xd} + \underline{h}^T \underline{\underline{R}}_x \underline{h} &= \\
\varepsilon + \underline{h}_{opt}^T \underline{\underline{R}}_x \underline{h}_{opt} - 2\underline{h}^T \underline{\underline{R}}_x \underline{h}_{opt} + \underline{h}^T \underline{\underline{R}}_x \underline{h} &= \\
\varepsilon + \underline{h}_{opt}^T \underline{\underline{R}}_x \underline{h}_{opt} - \underline{h}^T \underline{\underline{R}}_x \underline{h}_{opt} - \underline{h}_{opt}^T \underline{\underline{R}}_x \underline{h} + \underline{h}^T \underline{\underline{R}}_x \underline{h} &\implies \\
\boxed{\mathbb{E}\left[e^2[n]\right] = \varepsilon + \left(\underline{h}_{opt} - \underline{h}\right)^T \underline{\underline{R}}_x \left(\underline{h}_{opt} - \underline{h}\right).}&
\end{aligned}$$

So, as we can see, the MSE of **any filter** is a quadratic function of the filter coefficients and always lies in an $N-$dimensional surface. As $\underline{\underline{R}}_x$ is positive definite, the quadratic function is **convex** and hence it has a unique extreme that is a **minimum**, which is exactly what we wanted. The reference signal $d[n]$ only impacts on the position and value of the optimal solution, and not on the shape of the surface. We can also note that, as $\underline{\underline{R}}_x$ is positive definite, **any deviation from the optimum filter increases the MSE**. This increase depends only on $\underline{\underline{R}}_x$, and so, only on $x[n]$. This fact will be very useful in the design of adaptive filters.

Example: Imagine we have the following **signal cancellation** scenario. We estimate the value of a primary voice signal which contains an interference (voice+noise) in a helicopter, taken with a microphone. This interference has been isolated through other sensors in additional signals.

So, first of all we set our scenario:

- The microphone signal receives **voice**, **engine sound** and **sensor noise**.
- The reference sensor (noise isolation) receives **engine sound** and **noise**, but its engine sound is different than the one received through the microphone.

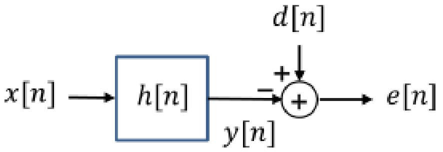
Our signals are:

- **Voice**: uncorrelated with the other signals.
- **Engine$_{\text{S}}$ (sensor) / Engine$_{\text{M}}$ (microphone)**: correlated signals with a helicopter-cabin effect.
- **Noise$_{\text{S}}$**: everything in the reference sensor that does not appear in the microphone. Uncorrelated.
- **Noise$_{\text{M}}$**: intrinsic system noise, low power, uncorrelated.

Then, mathematically we can represent all of this by

$$\begin{aligned}
\text{MIC: } m[n] &= v[n] + e_M[n] + \cancel{w_M[n]} \\
\text{SENSOR: } s[n] &= e_S[n] + w_S[n]
\end{aligned}$$

Which filter configuration do we need? A **cancellation scenario** like the following:



We want to obtain $e[n]$ as the **clean voice signal**. So, $x[n] = s[n]$ and $d[n] = m[n]$ will do the trick, as our filter will transform $x[n] = s[n] = e_S[n] + w_S[n]$ into a signal with an approximate value for the engine noise received in the microphone.

So, how do we solve the problem? It is already solved! We have three ways of solving it:

1. **Initial solution**: we depart from the fact that we want to minimize the MSE, so $\nabla_{\underline{h}} \left(e^2[n]\right) = \underline{0}$, and we solve the problem via the following equation:

$$\nabla_{\underline{h}} \left[\left((v[n] + e_M[n]) - \underline{h}^T (\underline{e}_S[n] + \underline{w}_S[n])\right)^2\right] = \underline{0}.$$

2. **Partial solution**: we depart from the principle of error-observation orthogonality, $\mathbb{E}\left[\underline{x}[n]e[n]\right] = 0$, and we solve the problem via:

$$\mathbb{E}\left[\left[\underline{e}_S[n] + \underline{w}_S[n]\right]\left[(v[n] + e_M[n]) - \underline{h}^T (\underline{e}_S[n] + \underline{w}_S[n])\right]\right] = \underline{0}.$$

3. **Final result**: we directly use the (already proven) fact that the optimal filter is

$$\underline{h}_{\text{opt}} = \underline{\underline{R}}_x^{-1} \underline{r}_{xd}.$$

For simplicity purposes, we will use the third option. Then,

$$\begin{aligned} \underline{\underline{R}}_x = \mathbb{E}\left[\underline{x}[n]\underline{x}^T[n]\right] = \left[x[n] = s[n] = e_S[n] + w_S[n]\right] &= \\ \mathbb{E}\left[\left(\underline{e}_S[n] + \underline{w}_S[n]\right)\left(\underline{e}_S[n] + \underline{w}_S[n]\right)^T\right] &= \\ \mathbb{E}\left[\underline{e}_S[n]\underline{e}_S^T[n]\right] + 2\mathbb{E}\left[\underline{e}_S[n]\underline{w}_S^T[n]\right] + \mathbb{E}\left[\underline{w}_S[n]\underline{w}_S^T[n]\right] &= \\ \left[\text{engine and noise uncorrelated}\right] &= \\ \mathbb{E}\left[\underline{e}_S[n]\underline{e}_S^T[n]\right] + \mathbb{E}\left[\underline{w}_S[n]\underline{w}_S^T[n]\right] = \boxed{\underline{\underline{R}}_{e_S} + \underline{\underline{R}}_{w_S}.} \end{aligned}$$

For $\underline{r}_{xd}$, we do the same:

$$\begin{aligned} \underline{r}_{xd} = \mathbb{E}\left[\underline{x}[n]d[n]\right] = \left[\underline{x}[n] = \underline{e}_S[n] + \underline{w}_S[n]\right] &= \\ \left[d[n] = v[n] + e_M[n]\right] = \mathbb{E}\left[\left[\underline{e}_S[n] + \underline{w}_S[n]\right]\left[v[n] + e_M[n]\right]\right] &= \\ \mathbb{E}\left[\underline{e}_S[n] \cdot v[n] + \underline{e}_S[n] \cdot e_M[n] + \underline{w}_S[n] \cdot v[n] + \underline{w}_S[n] \cdot e_M[n]\right] &= \\ \left[\text{voice and noise uncorrelated with the other signals}\right] &= \\ \mathbb{E}\left[\underline{e}_S[n] \cdot e_M[n]\right] = \boxed{\underline{r}_{e_S e_M}.} \end{aligned}$$

The final solution is, then,

$$\boxed{\left[\underline{\underline{R}}_{e_S} + \underline{\underline{R}}_{w_S}\right]\underline{h}_{\text{opt}} = \underline{r}_{e_S e_M}.}$$

The double function of the filter is evident in this solution: it **adapts the correlated part of** $s[n]$ (with $m[n]$) while **cancelling the uncorrelated one**.

Wiener-Hopf filter using a finite number of samples

As we already know, the optimal filter depends on the second ordre statistics of the signals. However, in a typical case we only have a (small) **finite number of available samples** from both the observable and reference signals. In that case, we can **minimize an estimation of the mean square error** over the available set of samples.

Let us assume that we have M samples of the reference signal and, given that the filter has N coefficients, $M+N-1$ samples of the observation signal. We can define:

$$\text{MSE} \equiv \frac{1}{M}\sum_{m=0}^{M-1} e^2[m] = \frac{1}{M}\sum_{m=0}^{M-1}\left(d[m]-\underline{h}^T\underline{x}[m]\right)^2.$$

Let us write this expression as a combination of vectors. If we arrange the M sample equations

$$e[n] = d[n] - \underline{h}^T\underline{x}[n]$$

in a vector, we can express the error as

$$\underline{e}^T = \underline{d}^T - \underline{h}^T\underline{\underline{X}},$$

given that

$$\underline{\underline{X}} = \begin{bmatrix} x[0] & x[1] & \cdots & x[M-1] \\ x[-1] & x[0] & \cdots & x[M-2] \\ \vdots & \vdots & \ddots & \vdots \\ x[-N+1] & x[-N+2] & \cdots & x[M-N] \end{bmatrix}, \ \underline{d} = \begin{bmatrix} d[0] \\ d[1] \\ \vdots \\ d[M-1] \end{bmatrix}, \ \underline{e} = \begin{bmatrix} e[0] \\ e[1] \\ \vdots \\ e[M-1] \end{bmatrix}$$

The **optimal filter** should minimize the MSE:

$$\begin{gathered}
\text{MSE} = \frac{1}{M}\sum_{m=0}^{M-1} e^2[m] = \frac{1}{M}\underline{e}^T\underline{e} = \frac{1}{M}\left(\underline{d}^T - \underline{h}^T\underline{\underline{X}}\right)\left(\underline{d}^T - \underline{h}^T\underline{\underline{X}}\right)^T, \\
\nabla_{\underline{h}}\text{MSE} = \underline{0} \iff \nabla_{\underline{h}}\frac{1}{M}\left(\underline{d}^T - \underline{h}^T\underline{\underline{X}}\right)\left(\underline{d} - \underline{\underline{X}}^T\underline{h}\right) = \underline{0} \iff \\
\iff \nabla_{\underline{h}}\frac{1}{M}\left(\underline{d}^T\underline{d} - \underline{d}^T\underline{\underline{X}}^T\underline{h} - \underline{h}^T\underline{\underline{X}}\underline{d} + \underline{h}^T\underline{\underline{X}}\underline{\underline{X}}^T\underline{h}\right) = \underline{0} \iff \\
\iff \frac{1}{M}\left(-2\underline{\underline{X}}\underline{d} + 2\underline{\underline{X}}\underline{\underline{X}}^T\underline{h}\right) = \underline{0} \iff \boxed{\underline{h} = \left(\underline{\underline{X}}\,\underline{\underline{X}}^T\right)^{-1}\underline{\underline{X}}\,\underline{d} := \underline{h}_{\text{opt}}.}
\end{gathered}$$

By comparison with the optimal expression having infinite samples, we can see that we are implicitly **estimating the cross-correlation vector and the correlation matrix**, based on available samples:

$$\underline{h}_{\text{opt}} = \underline{\underline{R}}_x^{-1}\underline{r}_{xd}, \qquad \underline{h}_{\text{opt}} = \left(\underline{\underline{X}}\,\underline{\underline{X}}^T\right)^{-1}\underline{\underline{X}}\,\underline{d}.$$

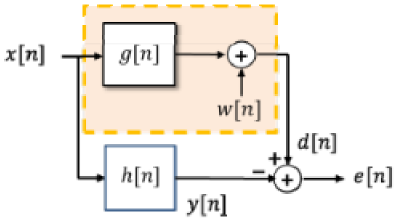
The estimations would be

$$\begin{gathered}
\hat{\underline{r}}_{xd}(\underline{x},\underline{d}) = \frac{1}{M}\underline{\underline{X}}\,\underline{d} = \frac{1}{M}\sum_{m=1}^{M}\underline{x}[m]d[m], \\
\hat{\underline{\underline{R}}}_x(\underline{x}) = \frac{1}{M}\underline{\underline{X}}\,\underline{\underline{X}}^T = \frac{1}{M}\sum_{m=1}^{M}\underline{x}[m]\underline{x}^T[m].
\end{gathered}$$

We can interpret the optimal filter (MMSE) using a finite number of samples as an estimate of the Wineer-Hopf filter using exact second-order statistics:

$$\boxed{\underline{h}_{\text{opt}} = \underline{\underline{R}}_x^{-1}\underline{r}_{xd} \implies \hat{\underline{h}}_{\text{opt}} = \left(\underline{\underline{X}}\,\underline{\underline{X}}^T\right)^{-1}\underline{\underline{X}}\,\underline{d}.}$$

In order to assess this estimator, let us fix a (simple) **system identification scenario**:



The application assumse noise-free observations (the known signal $\underline{\underline{X}}$) and noisy reference ($\underline{d}^T = \underline{g}^T \underline{\underline{X}} + \underline{w}^T$). The additive noise is modeled as white and Gaussian. Then, as we did in the previous chapter of the course,

$$\underline{d}^T = \underline{g}^T \underline{\underline{X}} + \underline{w}^T \implies \underline{w}^T = \underline{d}^T - \underline{g}^T \underline{\underline{X}}, \text{ with } \underline{\underline{C}}_w = \sigma^2 \underline{\underline{\mathrm{Id}}}.$$

$$f(\underline{x}; \underline{g}) = \frac{1}{\sqrt{(2\pi)^N \sigma^{2N}}} \cdot \exp\left[-\frac{\left(\underline{d}^T - \underline{g}^T \underline{\underline{X}}\right)\left(\underline{d}^T - \underline{g}^T \underline{\underline{X}}\right)^T}{2\sigma^2} \right] \implies$$

$$\implies \mathcal{L}(\underline{x}; \underline{g}) = \ln f(\underline{x}; \underline{g}) = -\frac{N}{2} \ln 2\pi\sigma^2 - \frac{1}{2\sigma^2}\left(\underline{d}^T - \underline{g}^T \underline{\underline{X}}\right)\left(\underline{d} - \underline{\underline{X}}^T \underline{g}\right) \implies$$

$$\implies \nabla_{\underline{g}} \mathcal{L}(\underline{x}; \underline{g}) = -\frac{1}{2\sigma^2} \nabla_{\underline{g}} \left(\underline{d}^T \underline{d} - \underline{d}^T \underline{\underline{X}}^T \underline{g} - \underline{g}^T \underline{\underline{X}}\underline{d} + \underline{g}^T \underline{\underline{X}\underline{X}}^T \underline{g} \right) =$$

$$= \frac{1}{2\sigma^2}\left[2\underline{\underline{X}}\underline{d}^T - 2\underline{\underline{X}\underline{X}}^T \underline{g} \right] = \frac{\underline{\underline{X}\underline{X}}^T}{\sigma^2} \left[\left(\underline{\underline{X}\underline{X}}^T \right)^{-1} \underline{\underline{X}}\underline{d}^T - \underline{g} \right].$$

Hence the efficient estimator is (left) and the Fisher information matrix is (right):

$$\boxed{\underline{g} = \left(\underline{\underline{X}\underline{X}}^T\right)^{-1} \underline{\underline{X}}\underline{d}^T, \qquad \underline{\underline{I}} = \sigma^2 \left(\underline{\underline{X}\underline{X}}^T\right)^{-1}.}$$

As we can see, the estimator that we have developed for the Wiener-Hopf filter is a MVUE estimator (it's **efficient**).

3.2. Linear Prediction

Introduction

In **signal prediction**, we estimate the value of a random signal at a given time instance ($x[n_0]$), based on other time instance values ($x[n_0 - 1], x[n_0 - 2], \ldots$).

Design: we compare the current signal value $x[n_0]$ with its estimation $y[n_0]$.
Use: the current signal value $x[n_0]$ may not be available and we produce an estimation. If $x[n_0]$ is available, we produce the estimation error $e[n_0]$.

The application assumes that observations and reference belong to the same noisy process. In the context of **linear prediction**, we can define three possible scenarios:

- **Forward prediction:** the current sample is estimated using only previous samples. For example, to forecast a given parameter value.
- **Backward prediction:** the current sample is estimated using only future samples. For example, for "remembering" a given value. Implies some delay.
- **Linear smoothing** (or interpolation)**:** the current sample is estimated combining past and future samples. For example, to recover a damaged signal.

Note: commonly in signal processing applications, what is important is the **ability to obtain a good estimation** of a sample, pretending that it is known, rather than forecasting it.

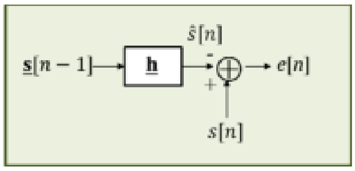
The Wiener-Hopf filter as a predictor

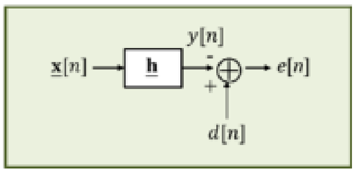
Let us analyze the FIR Wiener-Hopf filter in the context of **forward prediction**. Let us assume that we want to predict a given stationary process $s[n]$. In that case, $d[n] = s[n]$ and $\underline{x}[n] = \underline{s}[n-1]$. With this scenario, the Wiener-Hopf solution implies:

$$\underline{r}_{xd} = \mathbb{E}\left[\underline{s}[n-1]s[n]\right] = \begin{bmatrix} \mathbb{E}\left[s[n-1]s[n]\right] \\ \mathbb{E}\left[s[n-2]s[n]\right] \\ \vdots \\ \mathbb{E}\left[s[n-N]s[n]\right] \end{bmatrix} = \begin{bmatrix} r_s[-1] \\ r_s[-2] \\ \vdots \\ r_s[-N] \end{bmatrix} = \boxed{\underline{r}_s[-1]}$$

$$\underline{\underline{R}}_x = \mathbb{E}\left[\underline{s}[n-1]\underline{s}^T[n-1]\right] = \begin{bmatrix} r_s[0] & r_s[1] & \cdots & r_s[N-1] \\ r_s[-1] & r_s[0] & \cdots & r_s[N-2] \\ \vdots & \vdots & \ddots & \vdots \\ r_s[-N+1] & r_s[-N+2] & \cdots & r_s[0] \end{bmatrix} = \boxed{\underline{\underline{R}}_s}$$

So we have the following relations between the two schemes:





$$\begin{array}{ccccc} d[n] & \Longrightarrow & s[n]: & \text{reference signal} \Longrightarrow \text{ current sample} \\ \underline{x}[n] & \Longrightarrow & \underline{s}[n-1]: & N \text{ data samples } \Longrightarrow N \text{ previous samples} \\ \underline{h} & \Longrightarrow & \underline{h}: & \text{filter } (N \text{ taps}) \Longrightarrow \text{ predictor filter } (N \text{ taps}) \\ y[n] & \Longrightarrow & \hat{s}[n]: & \text{filtered signal} \Longrightarrow \text{ current predicted sample} \\ e[n] & \Longrightarrow & e[n]: & \text{prediction error} \Longrightarrow \text{ prediction error} \end{array}$$

When the optimal filter is used:

- Error is orthogonal to data:

$$\mathbb{E}\left[\underline{s}[n-1]e[n]\right] = \underline{0}$$

- The power of the error is lower than the power of the reference signal:

$$\mathbb{E}\left[s^2[n]\right] \geq \mathbb{E}\left[e^2[n]\right]$$

- The minimum error power is:

$$\varepsilon = r_s[0] - \underline{h}_{\text{opt}}^T \underline{r}_s$$

The expression for the optimal filter is

$$\underline{\underline{R}}_s \underline{h}_{\text{opt}} = \underline{r}_s,$$

and the power of the error for any filter $\underline{h}$ is

$$\mathbb{E}\left[e^2[n]\right] = \varepsilon + \left(\underline{h} - \underline{h}_{\text{opt}}\right)^T \underline{\underline{R}}_s \left(\underline{h} - \underline{h}_{\text{opt}}\right)$$

Linear prediction for signal coding (LPC)

Assuming **stationarity**, the Wiener-Hopf filter minimizes the MSE between the process and its estimation (MMSE: **minimum error power**): signals are usually processed by (close to) stationary segments called **frames**. In speech coding, this is typically 20ms.

As the power of the error is lower than the power of the reference signal. That allows defining a **coding gain** G_C:

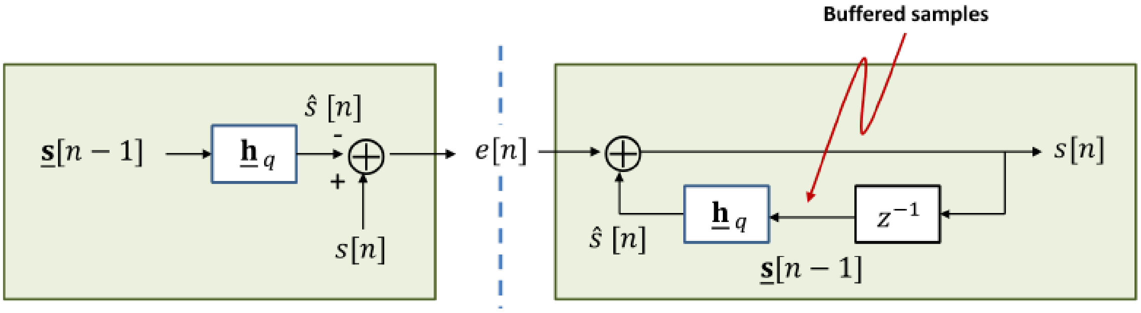
$$\sigma_s^2 = \mathbb{E}\left[s^2[n]\right] \geq \mathbb{E}\left[e^2[n]\right] = \sigma_e^2 \implies \boxed{G_C = \frac{\sigma_s^2}{\sigma_e^2}.}$$

Given a filter different from the optimal one (for example, the **quantized filter** $\underline{h}_q$), the obtained error power and actual coding gain can be computed:

$$\mathbb{E}\left[e^2[n]\right] = \varepsilon + \left(\underline{h}_q - \underline{h}_{\text{opt}}\right)^T \underline{\underline{R}}_s \left(\underline{h}_q - \underline{h}_{\text{opt}}\right)$$

Coder/Decoder structure

For each frame (assumed to be a stationary signal), the decoder receives the filter that has been used for predicting the signal and the prediction error. Assuming **amplitude-discrete signals** $s[n], \hat{s}[n], e[n] \in \mathbb{Z}$, the receiver can reconstruct the original signal $s[n]$ without loss.


Buffered samples
$\underline{\mathbf{s}}[n-1]$
$\underline{\mathbf{h}}_q$
$\hat{s}[n]$
-
+
$s[n]$
$e[n]$
$s[n]$
$\hat{s}[n]$
$\underline{\mathbf{h}}_q$
z^{-1}
$\underline{\mathbf{s}}[n-1]$

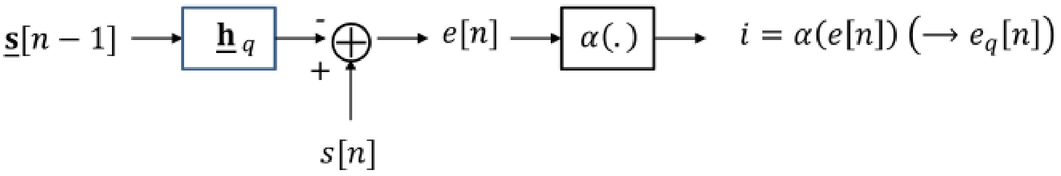
Internal variables are kept when starting to process a new frame. We can see on the left the **coder system**, and on the right the **decoder system**. Let's see how the first few iterations of the codec structure work:

If $n = 0$, the **first coding step** starts by observing that $\underline{s}^T[n-1] = \underline{s}^T[-1] = \underline{0}$, so $\hat{s}[n] = \hat{s}[0] = \underline{h}^T\underline{s}[-1] = 0$. Then, $e[n] = e[0] = s[0] - \hat{s}[0] = s[0]$, so the coder transmits the filter and this predicted value. Then, the decoder kicks in. The **first decoding step** starts by setting $e[n] = e[0] = s[0]$ and $\underline{s}[n-1] = \underline{s}[-1] = \underline{0}$, so then $\hat{s}[n] = \hat{s}[0] = \underline{h}^T\underline{s}[-1] = 0$ and then, it can reconstruct the $n-$th sample as $e[0] + \hat{s}[0] = s[0]$.

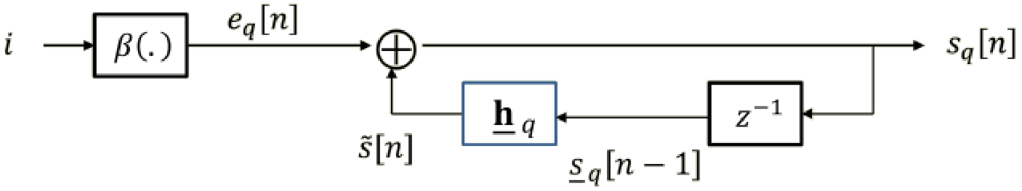
If $n = 1$, $\underline{s}^T[n-1] = \underline{s}^T[0] = (s[0], 0, \ldots, 0)$. Then, the prediction for $s[1]$ is $\hat{s}[1] = \underline{h}^T\underline{s}[0] = h_1 \cdot s[0]$. The prediction error is $e[1] = s[1] - \hat{s}[1] = s[1] - h_1 s[0]$. The decoder receives this error and the filter, and it can easiliy reconstruct the signal, as $e[n] = e[1] = s[1] - h_1 s[0]$ and it uses $\underline{s}^T[n-1] = \underline{s}^T[0]$. Then, its prediction is exactly the same for the lower part of the filter, $\hat{s}[n] = \hat{s}[1] = \underline{h}^T\underline{s}[0] = h_1 s[0]$ and so, it recovers the value of the signal as $e[1] + \hat{s}[1] = s[1] - h_1 s[0] + h_1 s[0] = s[1]$.

Quantization of the prediction error

So far, when we talked about quantization we have concentrated on the **quantization of values** that come directly from a **signal** (voice, audio, image) or are **model coefficients** (filter coefficients). As we are interested in transmitting the prediction error, the following question is quite relevant: should the same strategy apply in the case of **quantizing prediction error samples**? The **coding scheme**, in that case, can be the following one:


$\underline{s}[n-1]$
$\underline{\mathbf{h}}_q$
-
+
$e[n]$
$\alpha(.)$
$i = \alpha(e[n]) \;(\longrightarrow e_q[n])$
$s[n]$

In this kind of situation, how does the decoder work? Well, it tries to recover a (quantized) version of the input signal following the next scheme:



The decoder uses $\varepsilon_q[n]$ as the quantization error and $e_q[n]$ as the quantized error, so $e[n] = e_q[n] + \varepsilon_q[n]$. For simplicity, let us assume that the exact values of the N coefficients of $\underline{h}$ are available at the receiver side and so are the first N samples of the signal $\underline{s}[N-1]$. Then, the scheme evolves like this: at the N-th step,

- The coder sets $\underline{s}^T[n-1]$ to be $\underline{s}^T[N-1] = (s[N-1], \dots, s[0])$. Then, its prediction is $\hat{s}[n] = \hat{s}[N] = \underline{h}^T\underline{s}[N-1]$. Then, the prediction error is $e[n] = e[N] = s[N] - \hat{s}[N]$, which is exactly $s[N] - \underline{h}^T\underline{s}[N-1]$. This prediction error is quantized and hence, $e_q[n] = e_q[N] = e[N] - \varepsilon_q[N]$.
- The decoder predicts $\tilde{s}[N] = \underline{h}^T\underline{s}_q[N-1]$. But, as we have already said, the first N samples of the signal are available to the decoder. So, $\tilde{s}[N] = \underline{h}^T\underline{s}[N-1] = \hat{s}[N]$. Then, its reconstruction of the input signal is $s_q[N] = e_q[N] + \tilde{s}[N] = e[n] - \varepsilon_q[N] + \hat{s}[N]$. By definition, $e[N] = s[N] - \hat{s}[N]$, and so, $s_q[N] = s[N] - \hat{s}[N] - \varepsilon_q[N] + \hat{s}[N] = \boxed{s[N] - \varepsilon_q[N],}$ which is a kind of quantized version of the input signal.

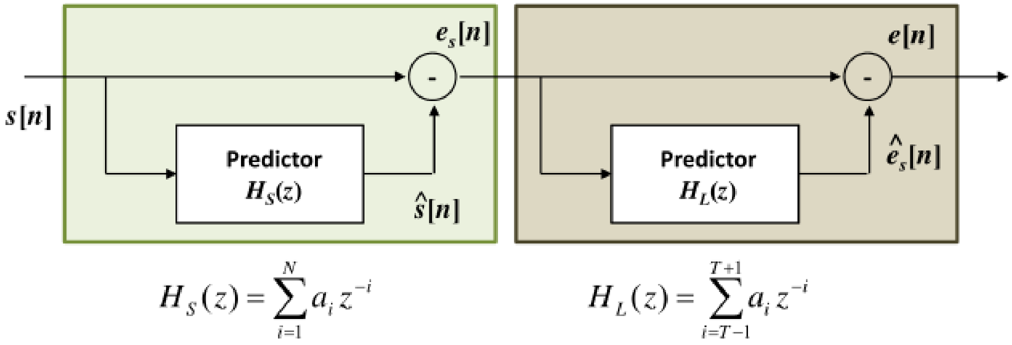
At the $(N+1)-$th step,

- The coder sends $e_q[N+1] = e[N+1] - \varepsilon_q[N+1]$.
- The decoder predicts $\tilde{s}[N+1] = \underline{h}^T\underline{s}_q[N] = \underline{h}^T(s[N] - \varepsilon_q[N], s[N-1], \dots, s[1])^T$, which after the filtering is equal to $\tilde{s}[N+1] = \underline{h}^T\underline{s}[N] - h_0\varepsilon_q[N] = \hat{s}[N+1] - h_0\varepsilon_q[N]$. So, its reconstruction of the input signal is $s_q[N+1] = e_q[N+1] + \tilde{s}[N+1] = e[N+1] - \varepsilon_q[N+1] + \hat{s}[N+1] - h_o\varepsilon_q[N]$. If we further develop this, we get that $s_q[N+1] = s[N+1] - \hat{s}[N+1] - \varepsilon_q[N+1] + \hat{s}[N+1] - h_0\varepsilon_q[N]$, and so $\hat{s}[N+1]$ cancels out and we get $s_q[N+1] = s[N+1] - \varepsilon_q[N+1] - h_0\varepsilon_q[N]$.

Linear prediction coding of speech signals

- in speech signals high temporal correlation between close samples (can be appreciated in autocorrelation or spectral density)
- linear prediction -> higher prediction gains in voiced signals
- large increase in the performance comes from the fact of including nearby samples in the predictor (N=8-10) as well as samples that are near to one pitch period T apart. samples in the middle of this range do not improve the prediction gain
- **short term & long term prediction**
  - short term: information about the periodicity of the signal is not explained

- to achieve better results usually a long term predictor is concatenated to the short term one



- short&long term predictor: intermediate samples not used
- long term: usually around 1-3 coefficients. can be used for pitch estimation (more on the slides 28-31 of the 3.2 section)

3.3. Adaptive Filtering

Introduction

Need for adaptive filtering

In the four scenarios that were presented as examples of the Wiener-Hopf filter, we can distinguish two different classes:

- **System identification and inversion**: if the system (or the system model) that is to be processed varies in time, the W-H solution has to adapt to these variations.
- **System prediction and cancellation**: if the processes that are analyzed are non-stationary, the W-H solution has to track and adapt to their statistical variations.

Assessment of adaptive filtering

The goal of an adaptive filter is to **first find and then track** the optimum filter as quick and accurately as possible. There are different algorithms for implementing the filter adaptation. Therefore, we need **criteria for assessing the quality** of these algorithms:

- **Speed of convergence**: (or speed of adaptation) it measures the ability of the algorithm to bring the adaptive solution to the omptimal one, independently of the initial conditions. It is a transient-phase property.+
- **Misadjustment:** (or quality of adaptation) it measures the stability of the reached solution, once convergence is achieved. It is due to the randomness of the input data. It is a steady-phase property.
- **Tracking:** if the processes that are analyzed are non-stationary, the W-H solution has to track and adapt to their statistical variations. It is a steady-state property.
- **Complexity:** commonly, it is measured in terms of the number of operations that the algorithm requires to process a new sample, or time update. Additional concepts such as memory usage and parallelization properties can be analyzed.

Steepest descent

Most adaptive filtering algorithms are obtained by simple **modifications or iterative methods** for solving deterministic optimization problems. In the sequel, we are going to study several aspects of **gradient-based optimization techniques**, from the theoretical point of view and still in a stationary scenario, as bases for the creation and understanding of adaptive methods.

Study of the error performance surface

The Wiener-hopf solution filter is optimal in the sense that **it minimizes the MSE of the prediction**; that is, the variance (power) of the error $e[n]$. Recall that, for any filter $\underline{h}$, the MSE can be expressed as:

$$\mathbb{E}\left[e^2[n]\right] = \varepsilon + \left(\underline{h}_{\text{opt}} - \underline{h}\right)^T \underline{\underline{R}}_x \left(\underline{h}_{\text{opt}} - \underline{h}\right)$$

This expression represents an hyper-surface in $\mathbb{R}^N$ with a minimum at $\underline{h}_{\text{opt}}$.

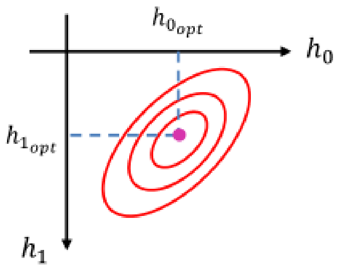
Example: Let us analyze the case for $N = 2$; that is, $\underline{h}^T = (h_0, h_1)$.

$$\left.\begin{aligned} \underline{h}_{\text{opt}} - \underline{h} = \Delta\underline{h} &= \begin{bmatrix} \Delta h_0 \\ \Delta h_1 \end{bmatrix} \\ \underline{\underline{R}}_x = \left[x[n] \in \mathbb{R}\right] &= \begin{bmatrix} r_x[0] & r_x[1] \\ r_x[1] & r_x[0] \end{bmatrix} \end{aligned}\right\} \Longrightarrow \mathbb{E}\left[e^2[n]\right] = \varepsilon + \Delta\underline{h}^T \underline{\underline{R}}_x \Delta\underline{h}$$

Then, the MSE is a quadratic function that looks like this:

$$\begin{gathered} \mathbb{E}\left[e^2[n]\right] = \varepsilon + r_x[0](\Delta h_0)^2 + 2\Delta h_0 \Delta h_1 r_x[1] + r_x[0](\Delta h_1)^2 = \\ \varepsilon + r_x[0]\left(h_{0_{\text{opt}}} - h_0\right)^2 + 2r_x[1]\left(h_{0_{\text{opt}}} - h_0\right)\left(h_{1_{\text{opt}}} - h_1\right) + r_x[0]\left(h_{1_{\text{opt}}} - h_1\right)^2 \end{gathered}$$

This defines a paraboloid in 3-dimensional space. The **level curves** of this paraboloid are ellipses in the plane:



What is the optimum filter if:

- We have an observed signal $x[n]$ with **low correlation** between consecutive samples:

$$\underline{\underline{R}}_x = \begin{bmatrix} 1.1 & 0.1 \\ 0.1 & 1.1 \end{bmatrix}, \; r_d[0] = 0.9486, \; \underline{r}_{xd} = \begin{bmatrix} 0.5272 \\ -0.4458 \end{bmatrix}$$

Given this data we can compute:

$$\begin{gathered} \underline{h}_{\text{opt}} = \underline{\underline{R}}_x^{-1} \underline{r}_{xd} \Longrightarrow \underline{h}_{\text{opt}} = \begin{bmatrix} 0.5204 \\ -0.4526 \end{bmatrix} \\ \varepsilon = r_d[0] - \underline{h}_{\text{opt}}^T \underline{r}_{xd} \Longrightarrow \varepsilon = 0.4725 \end{gathered}$$

- We have an observed signal $x[n]$ with **high correlation** between consecutive samples:

$$\underline{\underline{R}}_x = \begin{bmatrix} 40 & 39 \\ 39 & 40 \end{bmatrix}, \; r_d[0] = 0.9486, \; \underline{r}_{xd} = \begin{bmatrix} 0.5272 \\ -0.4458 \end{bmatrix}$$

Given this data we can compute:

$$\underline{h}_{\text{opt}} = \underline{\underline{R}}_x^{-1}\underline{r}_{xd} \implies \underline{h}_{\text{opt}} = \begin{bmatrix} 0.487 \\ -0.486 \end{bmatrix}$$
$$\varepsilon = r_d[0] - \underline{h}_{\text{opt}}^T\underline{r}_{xd} \implies \varepsilon = 0.5153$$

The minimization algorithm

An iterative algorithm that obtains the minimum of the error performance surface should fulfill the following criteria: (k is the iteration)

$$\lim_{k\to\infty} \underline{h}^k = \underline{h}_{\text{opt}}, \quad \lim_{k\to\infty} \mathbb{E}\left[e^2[n]\right] = \varepsilon$$

The proposed recursion uses the information in the gradient of the function to be minimized:

$$\underline{h}^{k+1} = \underline{h}^k - \frac{1}{2}\mu\nabla_{\underline{h}}\mathbb{E}\left[e^2[n]\right]\Big|_{\underline{h}^k}$$

This is the **steepest descent** algorithm; it is based on the Taylor expansion around $\underline{h}^k$. The positive parameter μ is known as the **step size**, and it determines the **speed of convergence** towards the optimum. The gradient $\nabla_{\underline{h}}$ of the MSE is used at each iteration, evaluated at each $\underline{h}^k$. At every point, it is perpendicular to the level curves, so **it does not always aim at the minimum**. Let's calculate this gradient in our setting, using $e[n] = d[n] - \underline{h}^T\underline{x}[n]$:

$$\nabla_{\underline{h}}\mathbb{E}\left[e^2[n]\right] = \nabla_{\underline{h}}\mathbb{E}\left[\left(d[n] - \underline{h}^T\underline{x}[n]\right)\left(d[n] - \underline{h}^T\underline{x}[n]\right)\right] =$$
$$\nabla_{\underline{h}}\mathbb{E}\left[d^2[n] - d[n]\underline{h}^T\underline{x}[n] - \underline{h}^T\underline{x}[n]d[n] + \underline{h}^T\underline{x}[n]\underline{h}^T\underline{x}[n]\right] =$$
$$\mathbb{E}\left[-d[n]\underline{x}[n] - \underline{x}[n]d[n] + 2\underline{x}[n]\underline{x}^T[n]\underline{h}\right] = -2\underline{r}_{xd} + 2\underline{\underline{R}}_x\underline{h}.$$

If we restrict this expression to $\underline{h}^k$ and substitute the result into the recursion, we get:

$$\boxed{\underline{h}^{k+1} = \underline{h}^k + \mu\left(\underline{r}_{xd} - \underline{\underline{R}}_x\underline{h}^k\right).}$$

Convergence analysis

Let us start analyzing the **one-dimension** case, $N = 1 \implies \underline{h} = h_0$:

$$\underline{\underline{R}}_x = r_x[0] \equiv \lambda \implies \mathbb{E}\left[e^2[n]\right] = \varepsilon + \lambda\left(h_{0_{\text{opt}}} - h_0\right)^2,$$
$$h_0^{k+1} = h_0^k - \frac{1}{2}\mu\frac{\partial}{\partial h_0}\mathbb{E}\left[e^2[n]\right]\Big|_{h_0^k},$$
$$h_0^{k+1} = h_0^k - \frac{1}{2}\mu\left(2\lambda h_0^k - 2\lambda h_{0_{\text{opt}}}\right) = h_0^k + \mu\lambda\left(h_{0_{\text{opt}}} - h_0^k\right) = (1-\mu\lambda)h_0^k + \mu\lambda h_{0_{\text{opt}}}$$

If we now subtract $h_{0_{\text{opt}}}$ from both sides, we get

$$h_0^{k+1} - h_{0_{\text{opt}}} = (1-\mu\lambda)\left(h_0^k - h_{0_{\text{opt}}}\right).$$

This expression allows for a change of variable that simplifies the analysis of convergence: if we call z to the difference in both sides of the equation, we have

$$z^{k+1} = (1-\mu\lambda)z^k \implies \boxed{z^{k+1} = (1-\mu\lambda)^k z^0.}$$

And, as the $\underline{h}^k$ converge to $\underline{h}_{\text{opt}}$, the z^k converge to 0. The **convergence range** is:

$$|1-\mu\lambda|<1 \iff \left\{\begin{matrix} 1-\mu\lambda<1 \\ -1+\mu\lambda<1 \end{matrix}\right\} \iff 0<\mu\lambda<2 \implies \boxed{0<\mu<\frac{2}{\lambda}.}$$

The extreme cases' behaviour is pretty obvious: if $\mu=0$, then the z^k don't move at all; if $\mu=\frac{2}{\lambda}$, then the z^k change signs at each iteration because of the $(-1)^k$. In the intermediate cases, the following happens:

- $0<\mu<\frac{1}{\lambda}$: the z^k tend towards 0 slowly, as $(1-\mu\lambda)$ is between 0 and 1.
- $\mu=\frac{1}{\lambda}$: $z^1=0$, as $(1-\mu\lambda)=0$.
- $\frac{1}{\lambda}<\mu<\frac{2}{\lambda}$: the z^k hop from one side to the other of 0, while approaching it.

To extend the previous analysis to the $N-$dimensional case, we need to establish some properties of the **correlation matrix**:

- It is **semipositive definite**: for any vector $\underline{v}$,

$$\underline{v}^T\underline{\underline{R}}_x\underline{v}=\underline{v}^T\mathbb{E}\left[\underline{x}\underline{x}^T\right]\underline{v}=\mathbb{E}\left[\underline{v}^T\underline{x}\underline{x}^T\underline{v}\right]=\left[\underline{v}^T\underline{x}=\alpha\right]=\mathbb{E}\left[\alpha^2\right]\geq 0.$$

- Because of the previous point, its **eigenvalues are non-negative**:

$$\underline{\underline{R}}_x\underline{u}=\lambda\underline{u} \implies \underline{u}^T\underline{\underline{R}}_x\underline{u}=\underline{u}^T\lambda\underline{u} \implies \lambda=\frac{\underline{u}^T\underline{\underline{R}}_x\underline{u}}{\underline{u}^T\lambda\underline{u}}=\frac{\alpha^2}{\|\underline{u}\|^2}\geq 0.$$

- Via the **spectral theorem**, we know that it can be decomposed into $\underline{\underline{R}}_x=\underline{\underline{U}}\underline{\underline{\Lambda}}\underline{\underline{U}}^T$, where $\underline{\underline{\Lambda}}$ is a diagonal matrix and $\underline{\underline{U}}$ is an orthogonal or unitary matrix.

We are going to perform a **change of variable** analogous to that in the 1-D case. Nevertheless, since we are in an $N-$dimensional problem, we have to account for a **displacement** and a **rotation**:

$$\begin{aligned}
\underline{h}^{k+1} &= \underline{h}^k+\mu\left(\underline{r}_{xd}-\underline{\underline{R}}_x\underline{h}^k\right)\\
\underline{h}^{k+1}-\underline{h}_{\text{opt}} &= \left(\underline{\underline{\mathrm{Id}}}-\mu\underline{\underline{R}}_x\right)\underline{h}^k+\mu\underline{r}_{xd}-\underline{h}_{\text{opt}}\\
\underline{h}^{k+1}-\underline{h}_{\text{opt}} &= \left(\underline{\underline{\mathrm{Id}}}-\mu\underline{\underline{R}}_x\right)\underline{h}^k+\mu\underline{\underline{R}}_x\underline{h}_{\text{opt}}-\underline{h}_{\text{opt}}\\
\underline{h}^{k+1}-\underline{h}_{\text{opt}} &= \left(\underline{\underline{\mathrm{Id}}}-\mu\underline{\underline{R}}_x\right)\underline{h}^k-\left(\underline{\underline{\mathrm{Id}}}-\mu\underline{\underline{R}}_x\right)\underline{h}_{\text{opt}}\\
&\boxed{\underline{h}^{k+1}-\underline{h}_{\text{opt}}=\left(\underline{\underline{\mathrm{Id}}}-\mu\underline{\underline{R}}_x\right)\left(\underline{h}^k-\underline{h}_{\text{opt}}\right).}
\end{aligned}$$

Now, we can compensate for the **displacement**. In order to obtain the **rotation**, we decompose $\underline{\underline{R}}_x$ into its spectral decomposition, $\underline{\underline{U}}\underline{\underline{\Lambda}}\underline{\underline{U}}^T$, and

$$\begin{aligned}
\underline{h}^{k+1}-\underline{h}_{\text{opt}} &= \left(\underline{\underline{\mathrm{Id}}}-\mu\underline{\underline{R}}_x\right)\left(\underline{h}^k-\underline{h}_{\text{opt}}\right)\\
\underline{h}^{k+1}-\underline{h}_{\text{opt}} &= \left(\underline{\underline{\mathrm{Id}}}-\mu\underline{\underline{U}}\underline{\underline{\Lambda}}\underline{\underline{U}}^T\right)\left(\underline{h}^k-\underline{h}_{\text{opt}}\right)\\
\underline{\underline{U}}^T\left(\underline{h}^{k+1}-\underline{h}_{\text{opt}}\right) &= \underline{\underline{U}}^T\left(\underline{\underline{\mathrm{Id}}}-\mu\underline{\underline{U}}\underline{\underline{\Lambda}}\underline{\underline{U}}^T\right)\left(\underline{h}^k-\underline{h}_{\text{opt}}\right)\\
\underline{\underline{U}}^T\left(\underline{h}^{k+1}-\underline{h}_{\text{opt}}\right) &= \left(\underline{\underline{U}}^T-\mu\underline{\underline{U}}^T\underline{\underline{U}}\underline{\underline{\Lambda}}\underline{\underline{U}}^T\right)\left(\underline{h}^k-\underline{h}_{\text{opt}}\right)\\
\underline{\underline{U}}^T\left(\underline{h}^{k+1}-\underline{h}_{\text{opt}}\right) &= \left(\underline{\underline{U}}^T-\mu\underline{\underline{\Lambda}}\underline{\underline{U}}^T\right)\left(\underline{h}^k-\underline{h}_{\text{opt}}\right)\\
\underline{\underline{U}}^T\left(\underline{h}^{k+1}-\underline{h}_{\text{opt}}\right) &= \left(\underline{\underline{\mathrm{Id}}}-\mu\underline{\underline{\Lambda}}\right)\underline{\underline{U}}^T\left(\underline{h}^k-\underline{h}_{\text{opt}}\right)\\
&\boxed{\underline{\underline{U}}^T\left(\underline{h}^{k+1}-\underline{h}_{\text{opt}}\right)\equiv\underline{z}^k \implies \underline{z}^{k+1}=\left(\underline{\underline{\mathrm{Id}}}-\mu\underline{\underline{\Lambda}}\right)\underline{z}^k.}
\end{aligned}$$

With this result, the different dimensions of the optimization problem have been decoupled, so every component can be analyzed separately:

$$z_i^{k+1} = (1-\mu\lambda_i) z_i^k \implies z_i^{k+1} = (1-\mu\lambda_i)^k z_i^0 \implies \lim_{k\to\infty} z_i^k = \lim_{k\to\infty} (1-\mu\lambda_i)^k z_i^0$$

We have already solved the 1-D problem, so we know that for each dimension we have $0 < \mu < \frac{2}{\lambda_i}$. Since we want to use the same μ for all dimensions, the (theoretical) bounds for μ are

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

In a practical environment we usually do not compute the eigenvalues of $\underline{\underline{R}}_x$, and instead use simpler and **more conservative** policies. For example, as we know that the trace operator is invariant through base changes, and $\lambda_{\max} \le \sum_k \lambda_k = \operatorname{tr}\left(\underline{\underline{R}}_x\right)$,

$$\begin{aligned} \lambda_{\max} \le \sum_{k=1}^{N} \lambda_k = \sum_{l=0}^{N-1} r_x[0] = N \cdot r_x[0] \implies \\ \implies \boxed{0 < \mu < \frac{2}{N \cdot r_x[0]}.} \end{aligned}$$

The **speed of convergence** can be quantized as the number of iterations (N_{iter}) that are necessary to reduce the distance between the achieved solution and the optimum to a given value ε. In a given dimension, we can write:

$$\begin{aligned} z_i^{k+1} &= (1-\mu\lambda_i)^k z_i^0, \\ |1-\mu\lambda_i|^{N_{\text{iter}}} &= \varepsilon \iff \boxed{N_{\text{iter}} = \frac{\ln\varepsilon}{\ln|1-\mu\lambda_i|}.} \end{aligned}$$

When generalized to the N dimensions, it can be shown that for small values of μ,

$$\boxed{N_{\text{iter}} \propto -\ln\varepsilon \frac{\lambda_{\max}}{\lambda_{\min}}.}$$

so, the speed of convergence is **proportional to the dispersion of the eigenvalues**. (examples of this in slides 26-31 of section 3.3).

Least Mean Square (LMS) approach

Stochastic approximation of the gradient

The steepest descent recursion for the Wiener-Hopf filter uses the correlation matrix and the cross-correlation vector, but **in a real setting, neither of those are known** and both have to be estimated via their **instantaneous approximations**. In this application, as signals are assumed to be non-stationary, we cannot use an estimator with a large memory (no accumulation of previous data). With these instantaneous estimations, we produce the **stochastic approximation of the gradient**:

$$\left.\begin{aligned} \underline{\underline{R}}_x &= \mathbb{E}\left[\underline{x}[n]\underline{x}^T[n]\right] & \implies & \quad \hat{\underline{\underline{R}}}_x(\underline{x}) = \underline{x}[n]\underline{x}^T[n] \\ \underline{r}_{xd} &= \mathbb{E}\left[\underline{x}[n]d[n]\right] & \implies & \quad \hat{\underline{r}}_{xd}(\underline{x}, d) = \underline{x}[n]d[n] \end{aligned}\right\} \implies$$

$$\implies \nabla_{\underline{h}} \mathbb{E}\left[e^2[n]\right]\Big|_{\underline{h}^k} = -2\underline{r}_{xd} + 2\underline{\underline{R}}_x \underline{h}^k \approx \boxed{-2\underline{x}[n]d[n] + 2\underline{x}[n]\underline{x}^T[n]\underline{h}^{n}.}$$

Note that now $\underline{h}^n$ is **estimated at each new sample** and the iterations and the samples are not different indexes anymore. Using this stochastic approximation, the recursion for the algorithm is:

$$\underline{h}^{k+1} = \underline{h}^k + \mu\left(\underline{r}_{xd} - \underline{\underline{R}}_x \underline{h}^k\right) \implies \underline{h}^{n+1} = \underline{h}^n + \mu\Big(\underline{x}[n]d[n] - \underline{x}[n]\underline{x}^T[n]\underline{h}^n\Big).$$

Using the definition of the error, $e[n] = d[n] - \underline{h}^T\underline{x}[n]$, and manipulating a bit the previous expression, we get

$$\underline{h}^{n+1} = \underline{h}^n + \mu\underline{x}[n]\Big(d[n] - \underline{x}^T[n]\underline{h}^n\Big),$$

and, as $\underline{u}^T\underline{v} = \underline{v}^T\underline{u}$, this is the same as

$$\boxed{\underline{h}^{n+1} = \underline{h}^n + \mu\underline{x}[n]e[n].}$$

This is called the **Least Mean Squares (LMS)** approach. The **LMS algorithm** is, therefore,

1. Filter the received signal: $y[n] = \underline{x}^T[n]\underline{h}^n$.
2. Compute the error: $e[n] = d[n] - y[n]$.
3. Update the coefficients: $\underline{h}^{n+1} = \underline{h}^n + \mu\underline{x}[n]e[n]$.
4. Return to 1 (if any more observations are received).

Convergence analysis of the LMS algorithm

We study the convergence of the LMS algorithm in a **stationary scenario**. As the gradient is estimated, the resulting value is **random**. Therefore, we need to **study the algorithm convergence in statistical terms**, in an expected value sense:

$$\begin{aligned}\underline{h}^{n+1} &= \underline{h}^n + \mu\Big(\underline{x}[n]d[n] - \underline{x}[n]\underline{x}^T[n]\underline{h}^n\Big),\\ \mathbb{E}\left[\underline{h}^{n+1}\right] &= \mathbb{E}\left[\underline{h}^n\right] + \mu\mathbb{E}\left[\underline{x}[n]d[n]\right] - \mu\mathbb{E}\left[\underline{x}[n]\underline{x}^T[n]\underline{h}^n\right].\end{aligned}$$

If we assume that the observations and the filter coefficients are approximately independent, we obtain the following equation:

$$\begin{aligned}\mathbb{E}\left[\underline{h}^{n+1}\right] &= \mathbb{E}\left[\underline{h}^n\right] + \mu\mathbb{E}\left[\underline{x}[n]d[n]\right] - \mu\mathbb{E}\left[\underline{x}[n]\underline{x}^T[n]\right]\mathbb{E}\left[\underline{h}^n\right],\\ &\boxed{\mathbb{E}\left[\underline{h}^{n+1}\right] = \mathbb{E}\left[\underline{h}^n\right] + \mu\underline{r}_{xd} - \mu\underline{\underline{R}}_x\mathbb{E}\left[\underline{h}^n\right].}\end{aligned}$$

In the expected value sense, we have obtained **the same iteration equation** as with the steepest descent method. The **step size** μ has to fulfill the same restrictions as in the Steepest Descent (SD) algorithm to achieve convergence:

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

The **speed of convergence** is the same in both cases (LMS, in the expected value sense, and SD):

$$N_{\text{iter}} \propto -\ln\varepsilon\frac{\lambda_{\max}}{\lambda_{\min}}$$

And, as in the SD algorithm, a **more conservative policy** is adopted for the step size:

$$0 < \mu < \frac{2}{N\hat{r}_x[0]}$$

In some cases, the dynamics of the input signal (that is, $r_x[0]$) are not constant due to non-stationarity. In such a case, the step size should be updated to guarantee convergence. So, the **normalized LMS** approach is:

- Use a **conservative value** for the step size: $\mu = \frac{2\alpha}{N\hat{r}_x[0]}$, with $0 < \alpha < 1$.
- **Dynamically estimate** the input power:
  - With an instantaneous estimation: $N\hat{r}_x[0] = \underline{x}^T[n]\underline{x}[n]$.
  - With a time-averaged estimation: $\hat{r}_x[0;n] = \gamma\hat{r}_x[0;n-1] + (1-\gamma)|x[n]|^2$.

Misadjustment of the LMS algorithm

Although the LMS algorithm converges in the expected value sense, the fact of estimating the gradient produces an **increase in variance** of the minimum error achieved. This is known as the **LMS steady-state excess MSE** (in absolute value, first) or as the **LMS misadjustment** (in relative terms, second):

$$\mathbb{E}\left[\hat{e}^2[n]\right] - \varepsilon \approx \frac{\mu\varepsilon\sum_{i=1}^N \lambda_i}{2-\mu\sum_{i=1}^N \lambda_i} = \frac{\mu N r_x[0]}{2-\mu N r_x[0]}.$$

$$M = \frac{\mathbb{E}\left[\hat{e}^2[n]\right] - \varepsilon}{\varepsilon} \approx \frac{\mu N r_x[0]}{2-\mu N r_x[0]} \implies \text{if } \mu \ll \frac{2}{N r_x[0]} \implies$$

$$\implies \boxed{M \approx \frac{\mu}{2}\sum_{i=1}^{N}\lambda_i = \frac{\mu}{2}N r_x[0].}$$

The μ parameter in the LMS algorithm:

- Is bounded to ensure convergence.
- The speed of convergence increases with μ.
- The misadjustment is proportional to μ.

Moreover, eigenvalue dispersion affects the speed of convergence, but not the misadjustment. The latter is mainly affected by increases in the power of the signal. (examples of this in slides 39-48 of section 3.3).

3.4. Applications of Optimal Filtering

This applications and how to solve them are all in the course slides of section 3.4. There, we solve the following problems:

- **Affine predictor:** Comparison between linear and affine prediction for non-zero mean signals.
- **Wiener-Hopf solution for highly correlated data:** Avoiding the use of close to singular correlation matrices.
- **Short term / Long term correlation:** Embedding a signal into noise. We separate a broadband noise signal from a narrowband signal that we want.