Menu of the day:

- 3 lectures on adaptive signal processing:
  - Today: basics, the normal equations, steepest-descent
  - Next Week: more on steepest-descent, stochastic gradient methods (LMS)
  - May 7th: other stochastic gradient methods, applications

- May 7th: MIDTERM (Argh!!!) during the EXERCISE SESSION (Room: to be announced). Exercises on topics up to next week (April 30th)
Revision of the Wiener filter

- Remember what a Wiener filter is:
  - $X[n]$ is the desired process, $W[n]$ is the noise process
  - $Y[n]$ is the measurement process:
    $$Y[n] = h(X, W),$$
    where \(h\) is a certain function (e.g. \(Y[n] = X[n] + W[n]\))
  - All the processes involved are stationary
  - We consider a linear estimator, i.e.
    $$\hat{X}[n] = \sum_{k \in S} h[k]Y[n - k],$$
    where \(S\) is certain set:
    * \(S = \mathbb{Z}\): non-causal case. Possible with images, where \(n\) is not really “time”
    * \(S = \{0, 1, \ldots\}\): causal case (all the past of the measurements). This something that we would use in practice..
    * \(S = \{0, 1, 2\}\): “homework” case (has to be solved at the exam..)
    * \(S = \{0, 1, \ldots, n\}\): Kalman filter case (tends to Wiener for large \(n\))
Revision of the Wiener filter

- The Wiener filter is optimal among all the linear estimators that consider the same set $S$. Optimal means that the error variance

$$J = \mathbb{E}[(\hat{X}[n] - X[n])^2]$$

is minimized.

- The solution is obtained by minimizing $J$ with respect to $h$ or by using the projection theorem

$$\sum_{k \in S} h[k] R_Y[m - k] = R_{XY}[m], \quad m \in S,$$

normal equations.

- Remark that: Correlation between $Y$ and $X =$ Possibility to estimate $X$ from $Y$.

- What if there is no correlation? Then the best linear estimate is $\hat{X} = 0$. What about non linear estimates?

- There may be non linear estimators that give an estimate which better than the Wiener filter (i.e. lower $J$).

- If $X[n]$ and $Y[n - k], k \in S$ are jointly gaussian, this is not possible, i.e. The Wiener filter is the best estimator among all the possible estimators (linear and non linear).
Revision of the Wiener filter

- We all love the Wiener filter, but...
  - Where do we take the correlations $R_Y[n], R_{XY}[n]$? In practice we can estimate these quantities only from measurements in real conditions. These are in general unknown.
  - Suppose that we can find the correlations and we build the Wiener filter, what happens if the parameters are changing with time? If the filter is fixed, the estimate is not the optimal one (actually, it can be a very bad one...)

- We need a system that is able to
  - LEARN the relation between $X$ and $Y$ when it is unknown
  - ADAPT the relation, when it is changing in order to recompute the optimal filter

- Let's see this with an example...
Echo cancellation

- A telephone communication system in the good old days...

- $x$ is the signal that goes from B to A, $d$ is the signal that goes from A to B

- $s$ is the signal generated by the person at position A. If everything works, $d = s$...
Echo cancellation

- A hands-free system for telephony...

- Now there is a signal $v$ that goes from the loudspeaker to the microphone. $v$ is not negligible with respect to $s$ and an echo is perceived at the telephone $B$

- We have $d = s + v$
**Echo cancellation**

- The Room Impulse Response models the relation between $x$ and $v$. We assume that the response of the room is linear, i.e.

$$v = h \ast x$$

and $h$ is the RIR

- Example: measured RIR in a room $4 \times 3$ m

- How can we get rid of the term $v$ in the transmitted signal $d$?
Echo cancellation

- The received signal $x$ is used to obtain an estimate $y$ of the signal $v$ coming from the loudspeaker to the microphone

\[ e[n] = d[n] - f \ast x[n] = s[n] + (h - f) \ast x[n] \approx s[n] \]
Echo cancellation

• It works if the filter $f \approx h$, i.e. if we measure the RIR precisely.

• Problem: The RIR $h$ may vary over space and time. If there is a mismatch between $h$ and $f$, we add more echo instead of cancelling it!

• Example, variation for displacements of 30 cm
**Echo cancellation**

- We need a filter that is able to learn the RIR and adapt with respect to variations.
- We want to cancel the echo “on average”. It is more appropriate to use a probabilistic model.

![Diagram of echo cancellation]

- The process $V$ is dependent on $X$.
- The process $S$ is (reasonably) uncorrelated with respect to $X$ and $V$. We are in the same setup of the Wiener filter!
**Wiener vs. Adaptive**

- Change of notation (sorry!) to be compatible the the literature. Wiener was

\[
E[n] = X[n] - \sum_i h[i]Y[n - i]
\]

- Now, we will use

\[
E[n] = D[n] - \sum_i f_n[i]X[n - i]
\]

- The cast of the movie:
  - \(D\): the "desired" process.
  - \(X\): the observed process.
  - \(Y\): the estimate of \(D[n]\) based on \(X\).
  - \(E\): the error process, \(D - Y\).
  - \(f_n\): adaptive filter at time index \(n\)
Wiener vs. Adaptive

• Why does the filter cancel echo?
  – To cancel the echo we would like $E[n] = S[n]$, i.e. the transmitted process equal to the generated process
  – If we minimize the error variance (as with the Wiener filter) we have,
    \[
    = \mathbb{E}[S[n]^2] + 2\mathbb{E}[S[n](V[n] - Y[n])] + \mathbb{E}[(V[n] - Y[n])^2] \\
    = \mathbb{E}[S[n]^2] + \mathbb{E}[E_V[n]^2]
    \]
  – Conclusion: the minimum of $\mathbb{E}[E[n]^2]$ with respect to $f_n$ coincide with the minimum of $\mathbb{E}[E_V[n]^2]$.

• As for the Wiener filter the goal is to minimize a cost function of the form
  \[ J_n(f_n) = E[E^2[n]] \]
**Wiener vs. Adaptive**

- What is new with respect to Wiener filtering?
  - LEARN and ADAPTATION algorithm by means of...
  - ACCESS to the processes $D[n]$ and/or $E[n]$ (only the correlation $R_{DX}[n]$ was needed to “describe” the desired output for the Wiener filter)

- The adaptive filter has 3 “ports”
Adaptive filtering: basics

- OK, that’s cute, but how do we do the adaptation?

- We have to generalize the Wiener filter to the non stationary case:
  
  Method I: minimization of the cost function $J_n(f_n) = E[E^2[n]]$ with respect to $f_n$
  
  The estimate of $D[n]$ is
  
  $$Y[n] = \sum_{i \in S} f_n[i]X[n - i]$$
  
  with $S$ a set of integers as done before. We set the gradient equal to zero, i.e.
  
  $$\frac{\partial J_n}{\partial f_n[i]} = \frac{\partial E[E[n]^2]}{\partial f_n[i]}$$, \quad i \in S.
  
  we obtain a generalized version of the normal equations
  
  $$\sum_{j \in S} f_n[j]R_X(n - j, n - i) = R_{DX}(n, n - i) \quad \forall i \in S, \forall n \in \mathbb{Z}.$$ 

  If the number of elements of $S$ is $L$, we have $L$ equations in $L$ unknowns. For each time index $n$ we solve the system of equations to compute the optimal filter $f_n^*$ at time $n$. 
Adaptive Signal Processing

Adaptive filtering: basics

- OK, but we need the correlations

\[ R_X[n, m] = \mathbb{E}[X[n]X[m]], \]

\[ R_{DX}[n, m] = \mathbb{E}[D[n]X[m]], \]

where do we take them?

A simple solution is to take the estimates:

\[ \hat{R}_X[n - i, n - j] = \frac{1}{M} \sum_{k=0}^{M-1-i+j} x[n - k]x[n - i + j - k] \quad i \geq j \]

\[ \hat{R}_{DX}[n, n - i] = \frac{1}{M} \sum_{k=0}^{M-1-i} d[n - k]x[n - i - k] \quad i \geq 0 \]

i.e. we take the latest \( M \) samples of the realizations of \( X[n] \) and \( D[n] \) to estimate the correlations (assuming ergodicity and that the processes are changing slowly) The parameter \( M \) is a tradeoff between estimation error and ability to track variations
**Adaptive filtering: basics**

- **Method II: projection theorem**

  As done with the Wiener filter, we can find the optimal filter by defining an Hilbert space and applying the projection theorem. Geometric interpretation:

\[ Y[n] \quad D[n] \quad E[n] \quad \mathcal{W} = \text{span}_{i \in S} \{ X[n - i] \} \]

The error \( E[n] \) is minimized when it is orthogonal to the set of measurements, i.e.

\[
\mathbb{E} \left[ \left( D[n] - \sum_{j \in S} f_{n}[j] X[n - j] \right) X[n - i] \right] = 0, \quad i \in S
\]
Adaptive filtering: “FIR” case

- We want to look for a solution with low complexity. Let’s consider a set $S$ which is finite, i.e.

\[ Y[n] = \sum_{i=0}^{L-1} f_n[i] X[n - i] \]

This is an FIR filter which is varying with $n$

- Let’s simplify the notation by using vectors:

\[
\begin{align*}
    f_n &= \begin{bmatrix} f_n[0] & f_n[1] & \ldots & f_n[L - 1] \end{bmatrix}^T, \\
    X_n &= \begin{bmatrix} X[n] & X[n - 1] & \ldots & X[n - L + 1] \end{bmatrix}^T.
\end{align*}
\]

Thus, the filtered process $Y[n]$ is

\[ Y[n] = f_n^T X_n = X_n^T f_n \]
Adaptive filtering: “FIR” case

- The normal equations become:

\[ R_{X,n} f_n = r_{DX,n}, \]

where

\[
R_{X,n} = \begin{bmatrix}
R_X[n,n] & R_X[n-1,n] & \cdots & R_X[n-L+1,n] \\
R_X[n,n-1] & R_X[n-1,n-1] & \cdots & R_X[n-L+1,n-1] \\
\vdots & \vdots & \ddots & \vdots \\
R_X[n,n-L+1] & R_X[n-1,n-L+1] & \cdots & R_X[n-L+1,n-L+1]
\end{bmatrix},
\]

\[
r_{DX,n} = \begin{bmatrix}
R_{DX}[n,n] \\
R_{DX}[n,n-1] \\
R_{DX}[n,n-2] \\
\vdots \\
R_{DX}[n,n-L+1]
\end{bmatrix}.
\]
Adaptive Signal Processing

Adaptive filtering: “FIR” case

- $R_{X,n}$ is the correlation matrix of a random vector: it is always positive semi-definite, i.e.
  \[ v^T R_{X,n} v \geq 0, \quad \forall v \in \mathbb{R}^L \]

- We consider only real processes, $R_{X,n}$ is symmetric (it would be hermitian in the complex case)

- If the processes $X[n]$ and $D[n]$ are jointly stationary, the auto and cross correlations $R_{X}$, $R_{XD}$ are time independent and the optimal filter $f$ is time invariant. The matrix $R_{X,n}$ has a Toeplitz structure (constant along the diagonals)

- If the correlation functions $R_{X}[n,m]$ and $R_{DX}[n,m]$ are given (e.g. we estimate them from measured realizations) and $R_{X,n}$ is invertible:
  \[ f^*_n = R^{-1}_{X,n} r_{DX,n} \]

- Are we done with the problem? NO
  - The estimation of $R_{X}[n,m]$ and $R_{DX}[n,m]$ from the realization is not very efficient
  - The solution of the normal equations may complex if $L$ is large...
Adaptive filtering: Echo cancellation (cont.)

- Assume that the size of the room at position A is on the order of few meters.
- If the walls are moderately reverberant, we can consider that the intensity of an acoustic ray is negligible after some reflections. Let’s say,
  \[ \text{max length ray path} \approx 30m \]
- Speed of sound is \( c \approx 343 m/s \).
  \[ \text{maximum time-of-flight between loudspeaker and microphone} \approx \frac{30}{343} \approx 100 \text{ms} \]
- Sampling frequency is \( F_S = 8 \text{KHz} \) (telephone quality)
  \[ L \approx 800 \]
- The matrix \( \mathbf{R}_{x,n} \) has size \( 800 \times 800 \).
  Would you solve such a system at the midterm?!
- The direct computation of the inverse has high complexity. If the sampling frequency is increased to improve the quality, the size of the matrix increases proportionally making the inversion even more critical.
Steepest-descent algorithms: how to solve a linear system

- Assume we fix the time index \( n \) (we neglect it in the notation) and assume that we know the correlation functions.

- The goal is to minimize the cost function \( J(f) \). This correspond to determine the solution of the normal equations

\[
R_X f = r_{DX}.
\]

- Given \( J(f) \), we determine a procedure that starts from an initial guess and then improves it recursively until converging to the minimum of \( J \).

- The procedure that we seek is of the form:

\[
\text{(new guess)} = \text{(old guess)} + \text{(a correction term)}.
\]
Steepest-descent algorithms: how to solve a linear system

- In our case,
  \[ f^{(i+1)} = f^{(i)} + \mu p, \quad i > 0. \]

  We use the superscript "(i)" to denote the index \( i \) of the recursion. The filter at iteration \( i \) is represented by \( f^{(i)} \).

- The vector \( p \) is the update direction and the scalar \( \mu \) is the step-size since it affects how small or how large the correction term is.

- The criterion for selecting \( \mu \) and \( p \) is to obtain \( J(f^{(i+1)}) < J(f^{(i)}) \).

- The index \( i \) runs from 0 onwards. The initial condition is specified at \( i = 0 \). Usually, the initial condition \( f^{(0)} \) is taken to be zero.
Steepest-descent algorithms: how to solve a linear system

- Consider again the cost function $J(f)$, what happens if we use a vector $f$ that is not the optimal one? The cost function is not minimal:

  $$J(f) = \mathbb{E}[E[n]^2] = \mathbb{E}[(D[n] - f^T X_n)^2] = \mathbb{E}[(D[n] - f^T X_n)(D[n] - X_n^T f)]$$
  $$= \mathbb{E}[D[n]^2] - 2f^T X_n + f^T X_n X_n^T f = \sigma_D^2 + f^T R_X f - 2f^T r_{DX}.$$

- We know that there is only one minimum that corresponds to the optimal filter $f^*$,

  $$f^* = R_X^{-1} r_{DX},$$

  and the minimum is

  $$J_{\text{min}} = \sigma_D^2 - r_{DX}^T R_X^{-1} r_{DX}.$$

We can rewrite the cost function as

$$J(f) = J_{\text{min}} + f^T R_X f - 2f^T r_{DX} + r_{DX}^T R_X^{-1} r_{DX}$$

$$= J_{\text{min}} + (f - f^*)^T R_X (f - f^*).$$

This is called a quadratic form
**What is the “shape” of our quadratic form?**

- $R_X$ is not an arbitrary matrix! The cost function has a certain shape.
- Remember that, since $R_X$ is a real symmetric matrix, it is always similar to a diagonal matrix, i.e.

$$R_X = V \Lambda V^T.$$  

- $\Lambda$ is a diagonal matrix which contains the eigenvalues of $R_X$

$$\Lambda = \text{diag} \lambda_k, \quad k = 0, \ldots, L - 1$$

- $V$ is an orthogonal matrix, i.e.

$$VV^T = V^TV = I.$$  

- The columns of $V$, say $v_k$, are the orthogonal eigenvectors of $R_X$, i.e.

$$R_X v_k = \lambda_k v_k, \quad \|v_k\| = 1.$$
What is the “shape” of our quadratic form?

- Since $R_X$ is positive semi-definite, the eigenvalues $\lambda_k$ are non negative:

$$v_k^T R_X v_k = v_k^T \lambda_k v_k = \lambda_k \geq 0$$

- We introduce the change of variable:

$$v = V^T (f - f^*)$$

This is a rototranslation in $L$ dimensions. The new reference system has the origin at the optimal solution $f^*$ and has the axes oriented along the eigenvectors of $R_X$.

The cost function in the new reference system...

$$J(f) = J_{\text{min}} + (f - f^*)^T R_X (f - f^*) = J_{\text{min}} + (f - f^*)^T V \Lambda V^T (f - f^*)$$

$$J(v) = J_{\text{min}} + v^T \Lambda v = J_{\text{min}} + \sum_{k=0}^{L-1} \lambda_k v[k]^2,$$

where $v[k]$ is the component $k$ of $v$. This representation shows that the cost function is a sum of parabolic surfaces along each variable.
Example:

Take $L = 2$ and with the following data

$$R_X = \begin{bmatrix}
\frac{5}{4} & -\frac{\sqrt{3}}{4} \\
-\frac{\sqrt{3}}{4} & \frac{7}{4}
\end{bmatrix} \quad r_{DX} = \begin{bmatrix}
\frac{10-\sqrt{3}}{8} \\
\frac{7-2\sqrt{3}}{8}
\end{bmatrix} \quad \sigma^2_D = \frac{27 - 4\sqrt{3}}{16}.$$

- We compute first the optimum filter

$$f^* = R_X^{-1}r_{DX} = \begin{bmatrix} 1 & \frac{1}{2} \end{bmatrix}^T,$$

- The minimum of the cost function is $J_{\text{min}} = \sigma^2_D - r_{DX}^T R_X^{-1} r_{DX} = 0$

- We compute the eigenvalues imposing that

$$\det(R_X - \lambda_k I) = 0, \quad k = 0, 1,$$

which gives $\lambda_0 = 2, \lambda_1 = 1$
Example

- The corresponding eigenvectors are computed by solving the two linear systems:

\[
(R - \lambda_k I)v_k = 0, \quad \|v_k\| = 1, \quad k = 0, 1.
\]

We find

\[
v_0 = \begin{bmatrix} \frac{\sqrt{3}}{2} & -\frac{1}{2} \end{bmatrix}^T, \quad v_1 = \begin{bmatrix} \frac{1}{2} & \frac{\sqrt{3}}{2} \end{bmatrix}^T.
\]

- The change of reference system is given by

\[
v = V^T(f - f^*) = \begin{bmatrix} \frac{\sqrt{3}}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{\sqrt{3}}{2} \end{bmatrix} \left( f - \begin{bmatrix} 1 \\ \frac{1}{2} \end{bmatrix} \right).
\]

- In the new reference system the equation of the cost function is

\[
J(v) = 2v[0]^2 + v[1]^2.
\]

The level curves, i.e the points corresponding to \(J(v) = K\) are the points that satisfy

\[
2v[0]^2 + v[1]^2 = K,
\]

which are ellipses of semi-axes \(\sqrt{K/2}\) and \(\sqrt{K}\).
Example:

Top view of the cost function (level curves for $K = 1, 2, 3$)
**Example:**

3D view of the cost function
Adaptive Signal Processing: Part II

Today:
- How to choose step-size and direction with steepest-descent
- How fast does steepest-descent converge?
- Stochastic gradient methods

Next week (May 7th) MIDTERM
- Lecture will take place as usual!
- Midterm will take place during the EXERCISE SESSION
- Exam is CLOSED BOOK. Only an A4 sheet is allowed (prepared by the students). No microscopes allowed in the classroom :-)
- Be here on time!
Steepest-descent algorithms: search direction

- Let’s go back to our iteration $f^{(i+1)} = f^{(i)} + \mu p$

- How does the cost function changes after one step?

\[
J(f^{(i+1)}) = \sigma_D^2 + (f^{(i)} + \mu p)^T R_X (f^{(i)} + \mu p) - 2r_{DX}^T (f^{(i)} + \mu p)
\]

\[
= J(f^{(i)}) + 2\mu f^{(i)}^T R_X p + \mu^2 p^T R_X p - 2\mu r_{DX}^T p
\]

\[
= J(f^{(i)}) + 2\mu (f^{(i)}^T R_X - r_{DX}^T) p + \mu^2 p^T R_X p.
\]

- How do we choose the search direction?

Remark that the gradient of $J(f)$ is

\[
\nabla J(f) = 2(f^T R_X - r_{DX}^T)
\]

The iteration changes the cost function according to

\[
J(f^{(i+1)}) = J(f^{(i)}) + \mu \nabla J(f^{(i)}) p + \mu^2 p^T R_X p.
\]

Since $\mu^2 p^T R_X p \geq 0$ a necessary condition to have $J(f^{(i+1)}) < J(f^{(i)})$ is to require that the update direction $p$ satisfies

\[
\nabla J(f^{(i)}) p < 0
\]
Steepest-descent algorithms: search direction

- If $\nabla J(f^{(i)}) = 0$, then $R_X f^{(i)} = r_{DX}$ and $f^{(i)} = f^*$. In this case the recursion should be stopped.

- There are many choices of vectors $p$ that satisfies the condition. Namely, any $p$ of the form
  
  $$p = -B \nabla J(f^{(i)})^T,$$

  for any positive definite matrix $B$. In fact,
  
  $$\nabla J(f^{(i)})p = -\nabla J(f^{(i)})B \nabla J(f^{(i)})^T < 0$$

- A simple solution (low complexity) is to take $B = \frac{1}{2} I$

  This gives the update direction is

  $$p = -\frac{1}{2} \nabla J(f^{(i)})^T = r_{DX} - R_X f^{(i)}$$

  This choice for $p$ reduces the recursion to

  $$f^{(i+1)} = f^{(i)} + \mu (r_{DX} - R_X f^{(i)}), \quad i > 0, \quad f^{(0)} = \text{initial guess}$$
Steepest-descent algorithms: step-size

- How do we choose the step-size? It is easier to use the change of variable introduced for the cost function.
- The filter coefficients \( f^{(i)} \) at iteration \( i \) are mapped to the vector
  \[
  v^{(i)} = V^T(f^{(i)} - f^*)
  \]
- What is the iteration in the new reference system?
  \[
  v^{(i+1)} = V^T(f^{(i+1)} - f^*)
  = V^T(f^{(i)} - f^* + \mu(r_{DX} - R_X f^{(i)})).
  \]
  Taking into account that \( r_{DX} = R_X f^* \), we have
  \[
  v^{(i+1)} = V^T(f^{(i)} - f^* - \mu R_X (f^{(i)} - f^*))
  = v^{(i)} - \mu V^T R_X V v^{(i)},
  \]
  which gives,
  \[
  v^{(i+1)} = (I - \mu \Lambda) v^{(i)}, \quad i > 0, \quad v^{(0)} = V^T(f^{(0)} - f^*).\]
Steepest-descent algorithms: step-size

- The coefficient matrix for this difference equation is now diagonal and equal to \( (I - \mu \Lambda) \)
- The evolution of the individual entries are decoupled. For the entry \( j \) of \( v^{(i)} \)

\[
v^{(i)}[j] = (1 - \mu \lambda_j)^i v^{(0)}[j], \quad i \geq 0
\]

where \( v^{(0)}[j] \) is the \( j \)-th entry of the initial condition \( v^{(0)} \).
- The factor \( (1 - \mu \lambda_j) \) is the mode associated with \( v^{(i)}[j] \).
- In order to have that \( v^{(i)}[j] \) tends to zero, regardless of \( v^{(0)}[j] \), the mode \( (1 - \mu \lambda_j) \) must have less than unit magnitude. This condition is both necessary and sufficient.

\[
|1 - \mu \lambda_j| < 1, \quad \text{for all } j = 0, \ldots, L - 1
\]

- This condition is equivalent to

\[
0 < \mu < \frac{2}{\lambda_{\text{max}}},
\]

where \( \lambda_{\text{max}} \) denotes the largest eigenvalue of \( R_X \).
Steepest-descent algorithms: the solution!

Theorem: (Steepest-descent algorithm)

Consider a zero-mean random variable $D[n]$ with variance $\sigma_D^2$ and a zero-mean random vector $X_n$ with $R_X = \mathbb{E}[X_n X_n^T]$ positive definite. Let $\lambda_{\text{max}}$ denote the largest eigenvalue of $R_X$. The solution $f^*$ of the linear least-mean-squares estimation problem:

$$\min_f \mathbb{E}[|D[n] - f^T X_n|^2]$$

can be obtained recursively as follows. Start with any initial guess $f^{(0)}$, chose any step-size $\mu$ that satisfies $0 < \mu < 2/\lambda_{\text{max}}$, and iterate for $i \geq 0$:

$$f^{(i+1)} = f^{(i)} + \mu(r_{DX} - R_X f^{(i)})$$

then $f^{(i)} \rightarrow f^*$, as $i \rightarrow \infty$. 
Transient behavior

- OK, it converges, but how fast?
  - The form of the exponential decay of each entry $v^{(i)}[j]$ to zero depends on the value of the mode $1 - \mu \lambda_j$
  - The sign of $1 - \mu \lambda_j$ determines whether the convergence of $v^{(i)}[j]$ occurs with or without oscillations
    - When $0 \leq 1 - \mu \lambda_j < 1$, the decay of $v^{(i)}[j]$ to zero is monotonic.
    - When $-1 < 1 - \mu \lambda_j < 0$, the decay of $v^{(i)}[j]$ to zero is oscillatory.
Transient behavior: example

- Take $L = 2$ and with the following data

  $$R_X = \begin{bmatrix} \frac{5}{4} & -\frac{\sqrt{3}}{4} \\ -\frac{\sqrt{3}}{4} & \frac{7}{4} \end{bmatrix} \quad r_{DX} = \begin{bmatrix} \frac{10-\sqrt{3}}{8} \\ \frac{7-2\sqrt{3}}{8} \end{bmatrix} \quad \sigma_D^2 = \frac{27 - 4\sqrt{3}}{16}.$$

- We found $\lambda_0 = 2$, $\lambda_1 = 1$. Hence,

  $$0 < \mu < \frac{2}{\lambda_{\text{max}}} = 1$$

- Assume that we start from the initial condition $f^{(0)} = \begin{bmatrix} 0 & 0 \end{bmatrix}^T$. The initial condition in the modified reference system is

  $$v^{(0)} = V^T(f^{(0)} - f^*) = \begin{bmatrix} \frac{1-2\sqrt{3}}{4} & \frac{-2-\sqrt{3}}{4} \end{bmatrix}^T.$$
**Transient behavior: example**

- Suppose that we take $\mu = 2/3$, the modes of the system are

\[
1 - \mu \lambda_0 = -\frac{1}{3} < 0, \quad 1 - \mu \lambda_1 = \frac{1}{3} > 0
\]
Optimal step-size

- For each value of the step-size $\mu$ there are (in general) $L$ modes $\{1 - \mu\lambda_j, j = 0, \ldots, L - 1\}$

- One of the modes have maximum magnitude, i.e. slowest rate of convergence. This mode will ultimately determine the convergence rate of the algorithm

- Different choices of $\mu$ give different slowest modes, we can select $\mu$ optimally by minimizing the value of the slowest mode:

$$\min_{\mu} \max_{j=0,\ldots,L-1} |1 - \mu\lambda_j|$$

with the constraint that each of the modes is stable, i.e. $|1 - \mu\lambda| < 1$.

- If $\mu = 0$, all the modes are one. As $\mu$ increases, the modes associated to the highest values of $\lambda_j$ decreases faster than modes corresponding the lowest values.
**Optimal step-size**

- Graphically:
**Optimal step-size**

- The optimal step-size $\mu^*$ corresponds to the point of intersection of $|1 - \mu \lambda_{\text{max}}|$ and $|1 - \mu \lambda_{\text{min}}|$, i.e.

$$1 - \mu^* \lambda_{\text{min}} = -(1 - \mu^* \lambda_{\text{max}}) \quad \Rightarrow \quad \mu^* = \frac{2}{\lambda_{\text{min}} + \lambda_{\text{max}}}$$

- By choosing $\mu = \mu^*$, there are (at least) two optimal slowest modes corresponding to $(1 - \mu^* \lambda_{\text{max}})$ and $(1 - \mu^* \lambda_{\text{min}})$. The first mode is negative, the second is positive.

- The magnitude of the slowest modes is

$$\max_j |1 - \mu^* \lambda_j| = \frac{\lambda_{\text{max}} - \lambda_{\text{min}}}{\lambda_{\text{max}} + \lambda_{\text{min}}}.$$ 

- We define the eigenvalue spread of the covariance matrix $R_X$ as

$$\rho = \lambda_{\text{max}} / \lambda_{\text{min}},$$

then

$$\max_j |1 - \mu^* \lambda_j| = \frac{\rho - 1}{\rho + 1}. $$
The optimal slowest modes that we can obtain depend on the eigenvalue spread of $R_X$. The larger the eigenvalue spread is, the slower will the convergence of the steepest-descent algorithm be.

Example:

- Case 1: $\lambda_0 = 2$, $\lambda_1 = 1$
- Case 2: $\lambda_0 = 10$, $\lambda_1 = 1$

In both cases, $\mu = \mu^*$. Same initial condition


Eigenvalue spread

- Case 1 (left), Case 2 (right)

- The eigenvalue spread determines the slowest mode in the optimal case and the speed of convergence of the steepest-descent algorithm
Learning curve

- We wanted to find $f$ such that $J(f) = \mathbb{E}[|D[n] - f^T X_n|^2]$ is minimized
- The steepest-descent recursion provides $f^{(i)}$, with cost values

$$J(f^{(i)}) = \mathbb{E}[|D[n] - f^{(i)^T} X_n|^2].$$
- The evolution of $J(f^{(i)})$ as a function of $i$ is called the learning curve. It provides useful information about the learning behavior of the steepest-descent algorithm
- By choosing the step size $0 < \mu < 2/\lambda_{\text{max}}$, we guarantee that $f^{(i)} \to f^*$ and $J(f^{(i)}) \to J_{\text{min}}$:

$$J(f^{(i)}) = J_{\text{min}} + \sum_{j=0}^{L-1} \lambda_j |v[j]^{(i)}|^2$$

$$= J_{\text{min}} + \sum_{j=0}^{L-1} \lambda_j (1 - \mu \lambda_j)^{2i} |v[j]^{(0)}|^2 \to J_{\text{min}}$$
Learning curve: example

- Case $\lambda_0 = 10$, $\lambda_1 = 1$ ($\mu^* = 2/11$)

- Learning curves for $\mu = \mu^*$, $\mu = \mu^*/2$ and $\mu = \mu^*/4$
Other cost functions

• We have a procedure for solving the least-mean-squares estimation problem

\[ \min_{f} \mathbb{E}[|D[n] - f^T X_n|^2]. \]

• We can apply steepest-descent to more general optimization problems of the form

\[ \min_{f} J(f) \]

with cost functions \( J(f) \) that are not necessarily quadratic in \( f \)

• The update recursion would continue be of the form

\[ f^{(i+1)} = f^{(i)} - \mu \nabla J(f^{(i)})^T \]

• Since \( J(f) \) may have both local and global minima, the successive iterates \( f^{(i)} \) may not approach a global minimum of \( J(f) \)

• It is difficult to predict if the iteration will converge to a local or a global minimum. The convergence behavior depends on the value of the step-size and on the location of the initial condition \( f^{(0)} \)
Steepest-descent over time

- We reintroduce the time index “\(n + 1\)”. Steepest-descent becomes

\[
f^{(i+1)}_{n+1} = f^{(i)}_{n+1} + \mu (r_{DX,n+1} - R_{X,n+1}f^{(i)}_{n+1}), \quad f^{(0)}_{n+1} = \text{initial guess}
\]

Ideally, we should iterate with respect to \(i\) for each temporal index \(n\)!

- Assume that \(R_{X,n+1}\) and \(r_{DX,n+1}\) are changing slowly. The optimal solution of the last index \(n\) is a good initial value for the index \(n + 1\), i.e. we set \(f^{(0)}_{n+1} = f^*_{n}\)

- we can suppose that the initial guess is very close to the optimal solution. To reduce the complexity of the algorithm, we perform a single iteration of the recursion, i.e. \(f^{(1)}_{n+1} \simeq f^*_{n+1}\).

The iteration becomes (neglecting \((1)\) and \(*\))

\[
f_{n+1} = f_n + \mu (r_{DX,n+1} - R_{X,n+1}f_n), \quad n > 1,
\]

\[
f_0 = \text{initial guess}
\]

- In the stationary case, the iteration is exactly the steepest-descent algorithm, where the role of \(f^{(i+1)}\) is played by \(f_{n+1}\)! This is the learning mechanism
**Steepest-descent over time**

- If we are not in the stationary case, but the variations are slow, hopefully the algorithm will still converge and will be able to track slow variations (adaptation/tracking mechanism).

- This is only an intuitive explanation since non-stationary processes are difficult to handle mathematically. The algorithm is studied assuming the stationary case and tested in the non-stationary case by means of simulations.

**Goal:** replace the gradient of the cost function by an approximation

**Stochastic-gradient algorithms**

- We will use a random quantity to replace a deterministic one (as the gradient of a steepest-descent method).

- The update direction becomes subject to random fluctuations that are referred to as gradient noise.
Stochastic-gradient algorithms: LMS

- We are interested in determining estimates at a very low computational cost. A common solution is to replace the statistic averages, i.e. the expectations, with a random quantity

\[ R_{X,n+1} = \mathbb{E}[X_{n+1}X_{n+1}^T] \approx X_{n+1}X_{n+1}^T \]

\[ r_{DX,n+1} = \mathbb{E}[D[n+1]X_{n+1}] \approx D[n+1]X_{n+1} \]

- Consider the cost function of steepest descent:

\[ J_{n+1}(f) = \mathbb{E}[(D[n+1] - f^TX_{n+1})^2] \]

with the approximation, the steepest-descent search direction becomes,

\[ p_n = -\frac{1}{2} \nabla J_{n+1}(f_n)^T = \mathbb{E}[D[n+1]X_{n+1} - X_{n+1}X_{n+1}^T f_n] \]

\[ p_n \approx X_{n+1}[D[n+1] - X_{n+1}^T f_n], \]

this gives the name stochastic-gradient to the method.
**Stochastic-gradient algorithms: LMS**

- The steepest-descent recursion is rewritten as

  \[ f_{n+1} = f_n + \mu X_{n+1}(D[n + 1] - X^T_{n+1} f_n), \quad n > 1, \quad f_0 = \text{initial guess}. \]

- Remark that the quantities \( p_n \) and \( f_n \) are now random vectors (we should use capital letters)

**Variation (This is what we call Least-Mean-Squares algorithm, or LMS)**

- To reduce even more the complexity, instead of the cost function \( J_{n+1} \), we consider \( J_n \) (this is valid if the statistics of the signal is changing slowly). The search direction becomes

  \[ p_n = -\frac{1}{2} \nabla J_n(f_n)^T = \mathbb{E}[D[n] X_n - X_n X_n^T f_n] \]

  \[ \simeq X_n [D[n] - X_n^T f_n] = X_n E[n], \]

  where we use the error process \( E[n] = D[n] - X_n^T f_n \)

- The recursion takes the simple form

  \[ f_{n+1} = f_n + \mu X_n E[n], \quad n > 1, \quad f_0 = \text{initial guess}. \]

- This iteration does not require the computation of the computation of quantity \( X^T_{n+1} f_n \)
**LMS**

Complexity:

\[ f_{n+1} = f_n + \mu X_n (D[n] - X_n^T f_n) \]

- Each step of the algorithm requires the following computations:
  1. Evaluation of \( X_n^T f_n \) (which is also the output of the filter), requires \( L \) multiplications and \( L - 1 \) additions
  2. Evaluation of the error \( E[n] = D[n] - Y[n] \) (one addition)
  3. Evaluation of the product \( \mu E[n] \), (one multiplication)
  4. Multiplication of the scalar \( \mu E[n] \) by the vector \( X_n \) (\( L \) multiplications)
  5. Addition of the two vectors \( f_n \) and \( \mu E[n] X_n \) (\( L \) additions)

- In summary the LMS algorithm needs \( 2L + 1 \) multiplications and \( 2L \) additions for each iteration, i.e. for each sample of the input signal.
Adaptive Signal Processing: Part III

Today:

- LMS: revision of last lecture
- LMS: convergence, learning curve, asymptotic cost function
- RLS (if enough time)
- More applications of adaptive filtering
Adaptive Signal Processing

**LMS convergence**

- LMS is inspired by the steepest-descent (for which we have a condition for convergence). However,
  - the filter obtained by the LMS algorithm is a random quantity since it is computed from the processes $X[n]$ and $D[n]$
  - The filter is a random multivariate process which, for appropriate values of $\mu$ should converge (in some sense) to the optimal filter
  - Even after convergence, the filter coefficients remain “noisy” (the error $E[n]$ is on average zero but it may locally be significant and the recursion will alter the filter coefficients)

- The step-size $\mu$ will control, in addition to the speed of convergence, also the level of noise of the filter coefficients. We have,
  - If $\mu$ is small, the filter $f_n$ converges slowly, but the final estimate will be only slightly noisy.
  - If $\mu$ is large, the filter will converge in less iterations, but the final value will be very noisy because the filter overreacts to the errors.
  - If $\mu$ is even larger, as in the case of steepest-descent the algorithm is unstable and the filter does not converge.
**LMS convergence. Simplified analysis**

- We consider only the stationary case for the input processes. We discuss the convergence in mean of the filter $f_n$:
  \[
  \lim_{n \to \infty} \mathbb{E}[f_n] = f^*.
  \]

- We use the “independence assumption”, i.e. we assume that $f_n$ is independent of $X_n$. This is clearly false, since $f_n$ depends on $X_{n-1}$ and $\mathbb{E}[X_n X_{n-1}^T] = 0$ is in general not true.

- We rewrite the iteration as
  \[
  f_{n+1} = f_n + \mu X_n (D[n] - X_n^T f_n)
  = f_n + \mu (D[n]X_n - X_n X_n^T f_n).
  \]
  We use the independence assumption to write
  \[
  \mathbb{E}[f_{n+1}] = \mathbb{E}[f_n] + \mu (r_{DX} - R_X \mathbb{E}[f_n]).
  \]

- Same iteration of steepest-descent (except that $(i)$ is replaced by $n$). We have the same results derived for the steepest-descent algorithm.
LMS convergence. Simplified analysis

- Results from steepest-descent:
  - $\mathbb{E}[f_n]$ converges to $f^*$ if the step-size satisfies
    \[ 0 < \mu < \frac{2}{\lambda_{\text{max}}}, \]
    where $\lambda_{\text{max}}$ is the maximum eigenvalue of $R_X$
  - All the results concerning the modes of the system, the optimal step-size and the time constants remain valid on average
  - The modes of convergence of the filter coefficients are exponentials. Each mode is perturbed by a random process whose amplitude increases with the step-size $\mu$
**LMS convergence. Simplified analysis**

- In practice, a more conservative condition is to take
  \[ 0 < \mu < \frac{2}{L\sigma_X^2}, \]
  where \( \sigma_X^2 \) is the variance of the process \( X[n] \). This is easier to verify since \( \sigma_X^2 \) can be estimated from the samples of the input process.

- Another (tighter) bound is
  \[ 0 < \mu < \frac{2}{LS_{X,\text{max}}}, \]
  where \( S_{X,\text{max}} = \max S_X(\omega) \). Experiments show that this is a necessary condition for the stability of LMS (for large values of the filter size \( L \)).

- A rigorous derivation of a necessary and sufficient stability bound on the step-size parameter \( \mu \) remains an open problem.
**LMS convergence. Eigenvalue spread**

- As in the case of steepest-descent, the eigenvalue spread is related to the number of iterations needed for convergence. It can be proved that,

\[
\rho = \frac{\lambda_{\text{max}}}{\lambda_{\text{min}}} \geq \frac{S_{X,\text{max}}}{S_{X,\text{min}}},
\]

where \( S_{X,\text{max}} \) and \( S_{X,\text{min}} \) are the maximum and the minimum values of the power spectral density \( S_X(\omega) \).

- This allows the estimation of the eigenvalue spread from the observation of the process \( X[n] \) and deduce the time of convergence of the LMS algorithm.

- The eigenvalue spread increases with the filter size \( L \): a longer filter allows the reduction of the residual error but at the cost of a higher number of iterations to reach convergence.

If we assume the stationary case, the maximum eigenvalue of the correlation matrix \( R_X^{(L)} \) of size \( L \times L \) is

\[
\lambda_{\text{max}}^{(L)} = \max_{\|v^{(L)}\|=1} v^{(L)T} R_X^{(L)} v^{(L)},
\]

In the same way,

\[
\lambda_{\text{max}}^{(L+1)} = \max_{\|v^{(L+1)}\|=1} v^{(L+1)T} R_X^{(L+1)} v^{(L+1)}.
\]
LMS convergence. Eigenvalue spread

- Clearly, \( \lambda_{\text{max}}^{(L+1)} \geq \lambda_{\text{max}}^{(L)} \).

- In the same way,
  \[
  \lambda_{\text{min}}^{(L+1)} \leq \lambda_{\text{min}}^{(L)}.
  \]

Thus,
\[
\frac{\lambda_{\text{max}}^{(L+1)}}{\lambda_{\text{min}}^{(L+1)}} \geq \frac{\lambda_{\text{max}}^{(L)}}{\lambda_{\text{min}}^{(L)}}.
\]

- The eigenvalue spread increases with the size \( L \)

\[
\lim_{L \to \infty} \frac{\lambda_{\text{max}}^{(L)}}{\lambda_{\text{min}}^{(L)}} = \frac{S_{X,\text{max}}}{S_{X,\text{min}}}
\]

- For large values of \( L \), the eigenvalue spread is the ratio between the maximum and minimum value of the spectral density (a flat spectral density gives a low eigenvalue spread, an irregular spectral density with peaks and valleys will correspond to a high eigenvalue spread)
**LMS ensemble-average learning curves**

- As in the case of steepest-descent, we define the learning curve as the sequence

\[
J(n) = \mathbb{E}[(|D[n] - X_n^T f_n|^2)] = \mathbb{E}[|E[n]|^2] = \,
\]

- At each iteration the estimated filter is a random quantity and the evaluation of the cost function is difficult

- In most cases the statistical description of the processes \( X[n], D[n] \) are not available but only their observations

- The learning curve can be estimated by observing some realizations of the error process \( E[n] \):
  - The algorithm is run for a certain number of iterations, e.g. for \( 0 \leq n \leq N - 1 \) (\( N \) is chosen large enough so that convergence is observed)
  - The experiment is repeated \( M \) times starting from the same initial condition \( f_0 \) and using data with the same statistical properties as in the first run
  - For each experiment, the error realization \( e[n]^{(j)}, 0 \leq j \leq M - 1 \), corresponding to \( E[n] \) is computed
LMS ensemble-average learning curves

- The ensemble-average learning curve, is defined as the average

\[ \hat{J}(n) = \frac{1}{M} \sum_{j=0}^{M-1} \left| e^{(j)}[n] \right|^2, \]

The average curve \( \hat{J}(n) \) represents an estimate of the true learning curve \( J(n) \).

- For the steepest-descent algorithm, the cost function approaches the quantity \( J_{\text{min}} \) at infinity. In the case of the LMS algorithm, it is difficult to compute the steady state error variance. An approximated study shows that

\[ \lim_{n \to \infty} J(n) \simeq J_{\infty} = J_{\text{min}} \left( 1 + \frac{\mu}{2} L \sigma_X^2 \right) , \]

where \( \sigma_X^2 \) is the variance of the input process \( X[n] \).

- The steady state error variance is larger than the minimum one, \( J_{\text{min}} \). The additive term is proportional to the step-size \( \mu \).
**LMS: example**

- Problem of identification:

  ![Diagram](image)

  - Given that:
    - The input process \( X[n] \) is a zero mean stationary process with correlation function
      
      \[
      R_X[n] = \begin{cases} 
      3/2 & \text{if } n = 0 \\
      -1/2 & \text{if } n = \pm 1 \\
      0 & \text{elsewhere}
      \end{cases}
      \]
    - The filter \( h \) has impulse response \( h[n] = \delta[n] + 1/2\delta[n - 1] \)
    - The process \( S[n] \) is a white process uncorrelated with \( X[n] \) and variance \( \sigma_S^2 = 1/25 \).
**LMS: example**

- We compute:
  - The correlation matrix $R_X$
    \[
    R_X = \begin{bmatrix}
    \frac{3}{2} & -\frac{1}{2} \\
    -\frac{1}{2} & \frac{3}{2}
    \end{bmatrix}
    \]
    and the eigenvalues are $\lambda_0 = 2$, $\lambda_1 = 1$.
  - According to the simplified analysis, the maximum value of the step-size is $2/\lambda_{\text{max}} = 1$.
    The more conservative condition gives
    \[0 < \mu < \frac{2}{3}.
    \]
  - We take $\mu = 1/9$ and we choose zero as initial guess of the filter coefficients. The LMS iteration becomes
    \[
    f_{n+1} = f_n + \frac{1}{9} X_n E[n] \quad n > 0,
    \quad f_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}^T.
    \]
LMS: example

- The trajectory of filter coefficients for $\mu = 1/9$ (left) and $\mu = 1/24$ (right)

- A reduction of the step-size increases the number of iterations needed for convergence, but reduces the fluctuations of filter coefficients at convergence
LMS: example

- Ensemble-average learning curves. $N = 60$, $M = 1000$

![Graphs showing ensemble-average learning curves for LMS example.](image)

- We have,

$$J_\infty = J_{\text{min}} \left( 1 + \frac{\mu}{2} L \sigma_X^2 \right),$$

with $J_{\text{min}} = \sigma_S^2 = 1/25$. The extra additive term is too small in this case.
Steepest-descent: Newton's method

- Consider again the steepest-descent method, we had:
  - The recursion
    \[ f^{(i+1)} = f^{(i)} + \mu p \]
  - The variation of the cost function:
    \[
    J(f^{(i+1)}) = J(f^{(i)}) + \mu \nabla J(f^{(i)}) p + \mu^2 p^T R_X p
    \]
  - The direction
    \[
    p = -\frac{1}{2} \nabla J(f^{(i)})^T = r_{DX} - R_X f^{(i)}
    \]
- Actually, we saw that any direction
  \[
  p = -B \nabla J(f^{(i)})^T,
  \]
  for any positive definite matrix \( B \) gives a reduction of the cost function (for an appropriate step-size \( \mu \))

Can we find a better search direction?
Steepest-descent: Newton’s method

- Let’s try with $B = (2R_X)^{-1}$ (this is a valid choice, since we take $R_X$ positive definite)
  - The iteration becomes $f^{(i+1)} = f^{(i)} + \mu R_X^{-1}(r_{DX} - R_X f^{(i)})$,
  - Remember that the optimal filter is $f^* = R_X^{-1}r_{DX}$
  - We obtain $f^{(i+1)} = (1 - \mu)f^{(i)} + \mu f^*$.
  - By choosing $\mu = 1$ the iteration converge in one step to the optimal solution

- What is so special in the new choice of the step direction? Recall that the Hessian matrix of a function is obtained by computing the second order partial derivatives, that is
  \[
  [H_J(f)]_{l,m} = \frac{\partial^2 J(f)}{\partial f[l] \partial f[m]}.\]

In our case

\[
H(f) = 2R_X.
\]

Therefore, $B = (H_J(f^{(i)}))^{-1}$
Steepest-descent: Newton’s method

- The Hessian matrix represents the shape of the cost function.
- The iteration

\[ \mathbf{f}^{(i+1)} = \mathbf{f}^{(i)} - \mu ( \mathbf{H}_f(\mathbf{f}^{(i)}))^{-1} \nabla J(\mathbf{f}^{(i)})^T, \quad i > 0, \quad \mathbf{f}^{(0)} = \text{initial guess}. \]

is known as the Newton’s recursion.

- For a quadratic cost function, the Newton’s recursion converges in one iteration. For other cost functions, we don’t know...

- For our problem, it doesn’t make sense to apply directly the Newton’s recursion since it needs the computation of the inverse of \( \mathbf{R}_X \)!
Newton’s iteration over time

- As we did for steepest-descent, we reintroduce the time index “$n + 1$” and we set the initial condition to the solution determined at time $n$ (if the time-variation is slow)

$$f_{n+1}^{(i+1)} = f_{n+1}^{(i)} + \mu R_{X,n+1}^{-1}(r_{DX,n+1} - R_{X,n+1} f_{n+1}^{(i)}), \quad f_{n+1}^{(0)} = f_{n}^{*}$$

- To reduce the complexity of the algorithm, we perform a single iteration of the recursion, i.e.

$$f_{n+1}^{(1)} \approx f_{n+1}^{*}.$$  

The iteration becomes (neglecting (1) and *)

$$f_{n+1} = f_{n} + \mu R_{X,n+1}^{-1}(r_{DX,n+1} - R_{X,n+1} f_{n}), \quad n > 1,$$

$$f_{0} = \text{initial guess}.$$
**Adaptive Signal Processing**

---

**RLS**

- We want to use a stochastic gradient approach to get rid of the correlations needed for Newton’s recursion. This method is called *Recursive-Least-Squares (RLS)*

- We already saw for LMS that

\[
\begin{align*}
\mathbf{p}_n &= -\frac{1}{2} \nabla J_n(\mathbf{f}_n)^T = \mathbb{E}[D[n] \mathbf{X}_n - \mathbf{X}_n \mathbf{X}_n^T \mathbf{f}_n] \\
&\simeq \mathbf{X}_n [D[n] - \mathbf{X}_n^T \mathbf{f}_n] = \mathbf{X}_n \mathbb{E}[n],
\end{align*}
\]

using the same for Newton’s recursion, we have

\[
\mathbf{f}_{n+1} = \mathbf{f}_n + \mu \hat{\mathbf{R}}_{X,n}^{-1} \mathbf{X}_n \mathbb{E}[n]
\]

\[
\mathbf{f}_0 = \text{initial guess}.
\]

- The matrix \( \hat{\mathbf{R}}_{X,n} \) is an approximation of the Hessian of the cost function, that is \( J_n \). We cannot approximate it with \( \mathbf{X}_n \mathbf{X}_n^T \), since it has to be invertible
To take into account time variation, we take

$$\hat{R}_{X,n} = \frac{1}{\alpha} \sum_{i=0}^{n} \lambda^{n-i} X_i X_i^T,$$

for some constant $0 < \lambda \leq 1$. The effect of $\lambda^{n-i}$ is to give a higher weight to the samples that are more recent and neglect progressively the older terms (that’s way $\lambda$ is called forgetting factor). The term $\alpha$ is used to renormalize the sum, i.e. $\alpha = \sum_{i=0}^{n} \lambda^{n-i}$. We combine $\alpha$ with the step size

We obtain the iteration,

$$f_{n+1} = f_n + \Phi_n^{-1} X_n E[n],$$

$$f_0 = \text{initial guess},$$

where $\Phi_n = \alpha \hat{R}_{X,n}$
RLS

- Problems:
  - $\Phi_n$ has to be invertible, for any $n$
  - Matrix inversion is complicated

We write $\Phi_n$ with the recursion

$$\Phi_n = \lambda \Phi_{n-1} + X_n X_n^T, \quad \Phi_{-1} = \varepsilon I,$$

where $\varepsilon$ is a small constant which has to be chosen according to the statistics of $X[n]$.

- To avoid inversion, we use the matrix inversion lemma: Let $A$ and $B$ be two positive definite $L \times L$ matrices related by

$$A = B^{-1} + CD^{-1}C^T,$$

where $D$ is a positive definite $N \times N$ matrix and $C$ is a $L \times N$ matrix. Then, the inverse of the matrix $A$ can be expressed as:

$$A^{-1} = B - BC(D + C^T BC)^{-1}C^T B.$$
**RLS**

- We apply the lemma to the recursion by taking:

\[
A = \Phi_n, \quad B = \lambda^{-1} \Phi^{-1}_{n-1}, \quad C = X_n, \quad D = 1.
\]

This gives,

\[
\Phi^{-1}_n = \lambda^{-1} \Phi^{-1}_{n-1} - \lambda^{-2} \frac{\Phi^{-1}_{n-1} X_n X_n^T \Phi^{-1}_{n-1}}{1 + \lambda^{-1} X_n^T \Phi^{-1}_{n-1} X_n}, \quad \Phi^{-1}_{-1} = \varepsilon^{-1} I,
\]

- The obtained RLS algorithm is:

1. Initialize \( P_{-1} = \varepsilon^{-1} I \) and \( f_0 \) to an initial guess (e.g. to zero).
2. At the arrival of samples \( X[n] \), build vector \( X_n \) and update matrix \( P_n \):

\[
P_n = \lambda^{-1} \left[ P_{n-1} - \lambda^{-1} \frac{P_{n-1} X_n X_n^T P_{n-1}}{1 + \lambda^{-1} X_n^T P_{n-1} X_n} \right].
\]

3. Compute the filter output \( Y[n] = f_n^T X_n \) and the error \( E[n] = D[n] - Y[n] \).
4. Update the filter coefficients via the recursion: \( f_{n+1} = f_n + P_n X_n E[n] \).
5. Return to step 2.
**RLS**

- With respect to LMS, the RLS algorithm converges faster, provided that the parameters $\lambda$ and $\varepsilon$ are chosen appropriately.
- RLS is less sensitive to the problem of eigenvalue spread.
- The computational cost of RLS is $O(L^2)$, i.e. one order of magnitude higher than to LMS.
- The asymptotic cost function obtained by using RLS is approximatively

$$J(n) \simeq J_{\text{min}} \left(1 + \frac{L}{n}\right),$$

i.e. no additional error with respect to the optimal solution.
RLS: example

- Previous example of identification. LMS ($\mu = 1/9$) vs. RLS with $\varepsilon = \sigma_X^2 = 3/2$ and $\lambda = 0.99$
**RLS: example**

- Learning curve for RLS (continuous) vs. LMS (dotted)
Applications: **acoustic echo cancellation**

- We already saw this. The diagram is the following:

  ![Diagram](image)

- \( L \) can be large and performance of LMS degrades with the size (eigenvalue spread and additive noise)

- Input signal is voice (far from white noise) eigenvalue spread may be large
Applications: adaptive noise cancellation (ANC)

- The noise is represented by $X[n]$. It reaches the cockpit, giving the disturbing process $D[n]$.
- We assume that $D[n]$ is a filtered version of $X[n]$. The goal is to remove $D[n]$ without knowing the relation with the actual source $X[n]$.
- We emit a signal $-Y[n]$ so that it cancels the disturbing signal $D[n]$.
Applications: adaptive noise cancellation (ANC)

- The total sound pressure is
  \[ E[n] = D[n] - Y[n] + S[n], \]
  
  \( S[n] \) is neither correlated to \( X[n] \) nor to \( Y[n] \) (e.g. voice of the person in the car)

- As for echo cancellation, the expected variance of \( E[n] \) is,
  \[ \mathbb{E}[E^2[n]] = \mathbb{E}[S^2[n]] + \mathbb{E}[(D[n] - Y[n])^2] + 2\mathbb{E}[S[n](D[n] - Y[n])]
  \]
  \[ = \mathbb{E}[S^2[n]] + \mathbb{E}[(D[n] - Y[n])^2], \]
  
  if the filter minimizes the error variance \( \mathbb{E}[E^2[n]] \), we have that \( Y[n] \approx D[n] \) and the signal \( Y[n] \) cancels the disturbing signal \( D[n] \)

- \( S[n] \) is not perturbed by the noise cancellation filter
Applications: adaptive noise cancellation (ANC)

- Too good to be true
  - We assumed that the cockpit corresponds to a single physical point: the quantities $D[n]$, $S[n]$ and $-Y[n]$ add up to give the error signal $E[n]$...
  - In reality, sound propagates in the cockpit following the wave equation: the processes $D[n]$ and $-Y[n]$ are not generated in the same point. The adaptive filter generates a signal $-Y[n]$ so that the resulting process at the position of the microphone has minimum variance
  - In other points of the cockpit, the two sources will generate an interference which can be additive or subtractive, according to the distances from the sources and the wavelength of the sound wave
Applications: adaptive noise cancellation (ANC)

- Example: room $3 \times 3$, two sources at 1 m distance. Cancellation at 1 KHz and 120 Hz

- We can use ANC only for low frequencies (< 150 Hz)
Applications: equalization

- We consider a simplified scheme of a communication system:

  ![Diagram of communication system]

  - $S[n]$ is a binary sequence, i.e. assumes the values $\pm 1$ with probability $1/2$, $h$ represents the transmission channel and $V[n]$ is some additive noise (zero mean).
  - The received process $\hat{S}[n]$ is obtained by a decision element on the basis of $X[n]$:

    $$X[n] = \sum_{m=-\infty}^{\infty} h[n-m]S[m] + V[n].$$
Applications: equalization

- The received symbol is chosen according to

\[
\hat{S}[n] = \begin{cases} 
+1 & \text{if } X[n] \geq 0 \\
-1 & \text{if } X[n] < 0.
\end{cases}
\]

- We want to minimize the error probability, \( P[S[n] \neq \hat{S}[n]] \)

- Ideally, \( h[n] = \delta[n] \) and \( X[n] = S[n] + V[n] \). In practice, \( X[n] \) depends on \( S[m] \), with \( m \neq n \). This is called intersymbol interference and increases the probability of error of the communication system

- A solution is to add a filter at the receiver called equalizer. Neglecting the noise \( V[n] \), the equalizer \( f \) should be such that \( F(z) = H(z)^{-1} \)

- As usual, the channel impulse response is not known and may vary over time, e.g. mobile communications
Applications: equalization

- We use an adaptive filter to estimate the channel response and track it.

Where do we take the “desired” process? We need a training sequence know both at the emitter and the receiver.

- Update only for the parts of the input process that correspond to a training sequence.
- The repetition/length of the training sequence is the result of a trade-off between capacity and ability to track fast variations of the channel.
Applications: equalization

- Better solution, use the transmitted sequence to adapt (when we are close to the optimal solution). This is called decision directed equalization.
- Still need the training sequence (when we are very far from the optimum, e.g. initial phase).
End of adaptive signal processing part

- That’s all...
- Please, give your comments
- Good luck for the midterm!
  - Students having the family name starting with [A-K] should stay in INN118 (Luciano & Ayfer)
  - Students having the family name starting with [L-Z] should move to INM202 (Andrea & Ivana)