COM303: Digital Signal Processing

Lecture 15: Stochastic and adaptive signal processing

Update the book!

Please download the new and improved Chapter 8 from the website!

overview

- random variables, random processes and stationarity
- spectral representation of random processes
- adaptive signal processing

from random variables to stationary random processes

Deterministic vs. stochastic

- deterministic signals are known in advance: $x[n] = \sin(0.2\,n)$
- interesting signals are *not* known in advance: $s[n] =$ what I'm going to say next
- we usually know something, though: $s[n]$ is a speech signal
- stochastic signals can be described probabilistically
- can we do signal processing with random signals? Yes!

Deterministic vs. stochastic

- deterministic signals are known in advance: $x[n] = \sin(0.2\,n)$
- interesting signals are *not* known in advance: $s[n] =$ what I'm going to say next
- we usually know something, though: $s[n]$ is a speech signal
- stochastic signals can be described probabilistically
- can we do signal processing with random signals? Yes!

Deterministic vs. stochastic

- deterministic signals are known in advance: $x[n] = \sin(0.2\,n)$
- interesting signals are *not* known in advance: $s[n] =$ what I'm going to say next
- we usually know something, though: $s[n]$ is a speech signal
- stochastic signals can be described probabilistically
- can we do signal processing with random signals? Yes!

Deterministic vs. stochastic

- deterministic signals are known in advance: $x[n] = \sin(0.2\, n)$
- interesting signals are *not* known in advance: $s[n] =$ what I'm going to say next
- we usually know something, though: $s[n]$ is a speech signal
- stochastic signals can be described probabilistically
- can we do signal processing with random signals? Yes!

Deterministic vs. stochastic

- deterministic signals are known in advance: $x[n] = \sin(0.2\, n)$
- interesting signals are *not* known in advance: $s[n] =$ what I'm going to say next
- we usually know something, though: $s[n]$ is a speech signal
- stochastic signals can be described probabilistically
- can we do signal processing with random signals? Yes!

Random variable

a *mapping* from a random event to a value $x \in \mathbb{R}$

Examples:

- tossing a coin: map heads to 0, tails to 1
- tossing a die: *discrete* r.v. is face value
- electric circuit: *continuous* r.v. is output voltage

Random variable

a *mapping* from a random event to a value $x \in \mathbb{R}$

Examples:

- tossing a coin: map heads to 0, tails to 1
- tossing a die: *discrete* r.v. is face value
- electric circuit: *continuous* r.v. is output voltage

Random variable

a *mapping* from a random event to a value $x \in \mathbb{R}$

Examples:

- tossing a coin: map heads to 0, tails to 1
- tossing a die: *discrete* r.v. is face value
- electric circuit: *continuous* r.v. is output voltage

Random variable

a *mapping* from a random event to a value $x \in \mathbb{R}$

Examples:

- tossing a coin: map heads to 0, tails to 1
- tossing a die: *discrete* r.v. is face value
- electric circuit: *continuous* r.v. is output voltage

Measuring probability

- cumulative distribution function (cdf):

$$F_x(\alpha) = P[x \leq \alpha], \quad \alpha \in \mathbb{R}$$

$$\lim_{\alpha \to \infty} F_x(\alpha) = 1$$

- probability density function (pdf):

$$f_x(\alpha) = \frac{dF_x(\alpha)}{d\alpha}, \quad \alpha \in \mathbb{R}$$

$$F_x(\alpha) = \int_{-\infty}^{\alpha} f_x(x)dx$$

Measuring probability

- cumulative distribution function (cdf):

$$F_x(\alpha) = P[x \leq \alpha], \quad \alpha \in \mathbb{R}$$

$$\lim_{\alpha \to \infty} F_x(\alpha) = 1$$

- probability density function (pdf):

$$f_x(\alpha) = \frac{dF_x(\alpha)}{d\alpha}, \quad \alpha \in \mathbb{R}$$

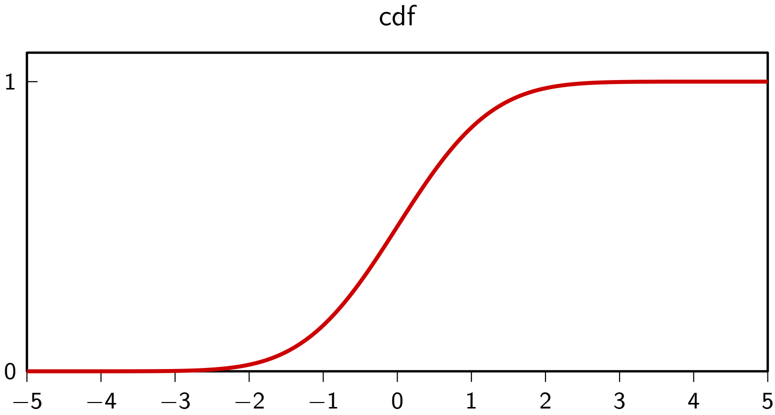
$$F_x(\alpha) = \int_{-\infty}^{\alpha} f_x(x)dx$$

Example

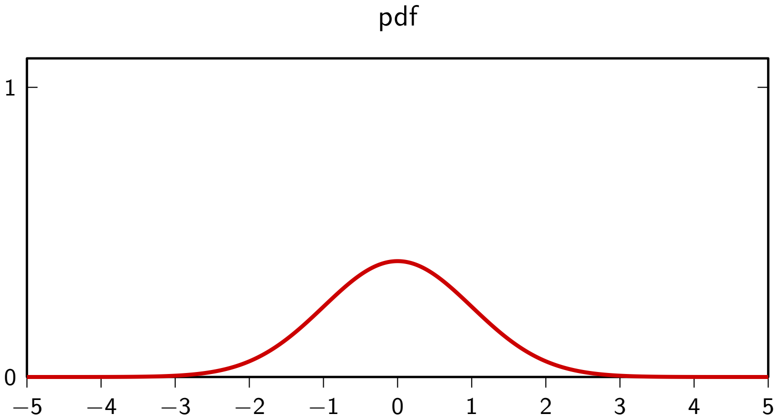
Measure repeatedly the temperature of melting ice:

- continuous random variable is the measured temperature
- should be zero Celsius
- changes in barometric pressure
- different mineral content in water
- inaccurate thermometer...

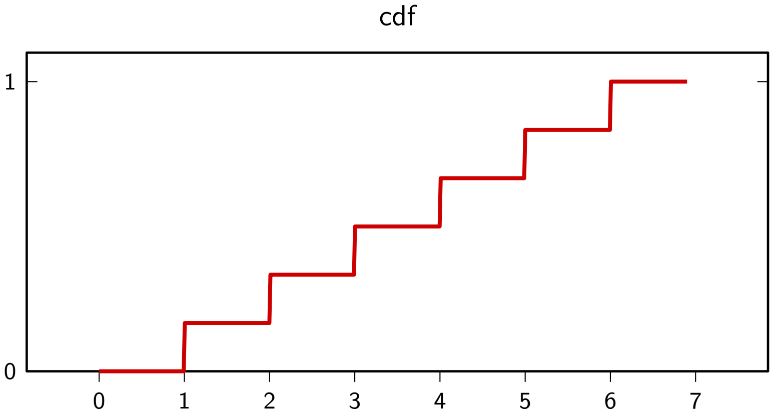
Continuous random variable: Gaussian $\mathcal{N}(0, 1)$



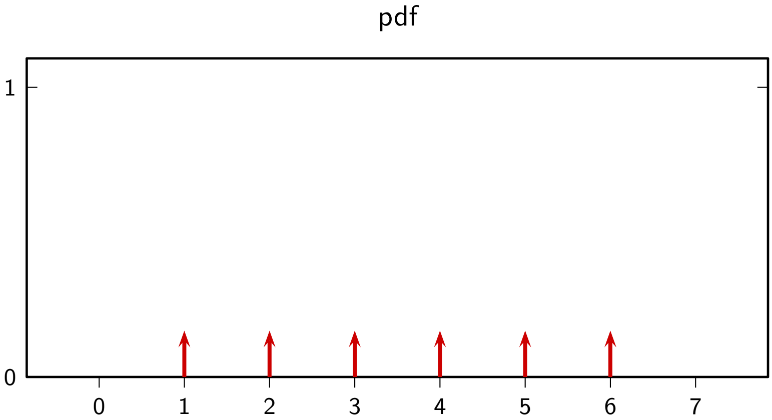
Continuous random variable: Gaussian $\mathcal{N}(0, 1)$



Discrete random variable: die toss



Discrete random variable: die toss



Expectation

$$\mathrm{E}\left[x\right] = \int_{-\infty}^{\infty} x\, f_x(x)\ dx$$

$$\mathrm{E}\left[g(x)\right] = \int_{-\infty}^{\infty} g(x) f_x(x)\ dx$$

Expectation

$$\mathrm{E}\left[x\right] = \int_{-\infty}^{\infty} x\, f_x(x)\; dx$$

$$\mathrm{E}\left[g(x)\right] = \int_{-\infty}^{\infty} g(x) f_x(x)\; dx$$

Moments

- raw moments: $\mathrm{E}\left[x^n\right] = \int_{-\infty}^{\infty} x^n f_x(x)\ dx$
- special case: mean: $m_x = \mathrm{E}\left[x\right] = \int_{-\infty}^{\infty} x\, f_x(x)\ dx$

- central moments: $\mathrm{E}\left[(x - m_x)^n\right] = \int_{-\infty}^{\infty} (x - m_x)^n f_x(x)\ dx$
- special case: variance: $\sigma_x^2 \ = \ \mathrm{E}\left[(x - m_x)^2\right]$

Gaussian Random Variable

$$f(x) = \mathcal{N}(m, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-m)^2}{2\sigma^2}}$$

- m is the mean
- σ^2 is the variance

Uniform Random Variable

$$f(x) = \mathcal{U}(A, B) = \frac{1}{B - A}$$

- $m = \dfrac{A + B}{2}$
- $\sigma^2 = \dfrac{(B - A)^2}{12}$

Discrete Uniform Random Variable

$$f(x) = \mathcal{U}\{A, B\} = \frac{1}{B - A + 1} \sum_{k=A}^{B} \delta(x - k)$$

- $m = \dfrac{A + B}{2}$
- $\sigma^2 = \dfrac{(B - A + 1)^2 - 1}{12}$

Relations between random variables

- cross-correlation: $R_{xy} = \mathrm{E}[x\,y]$.
- covariance: $C_{xy} = \mathrm{E}[(x - m_x)(y - m_y)]$.
- if zero-mean: $C_{xy} = R_{xy}$

to compute the covariance we need to know the *joint* pdf $f_{xy}(x, y)$:

$$\mathrm{E}[g(x, y)] = \int \int_{-\infty}^{\infty} g(x, y) \; \boxed{f_{xy}(x, y)} \; dxdy$$

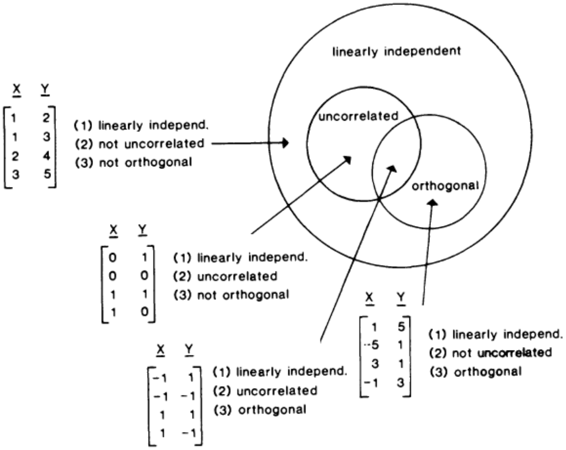
Special relations between random variables

- uncorrelated elements: $\mathrm{E}[xy] = \mathrm{E}[x]\,\mathrm{E}[y] = m_x\, m_y$
  (no linear relationship)
- independent elements: $f_{xy}(x, y) = f_X(x) f_y(y)$
  (no relationship)

Special relations between random variables

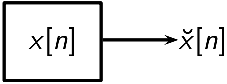
- uncorrelated elements: $\mathrm{E}[xy] = \mathrm{E}[x]\,\mathrm{E}[y] = m_x\, m_y$
  (no linear relationship)
- independent elements: $f_{xy}(x, y) = f_X(x) f_y(y)$
  (no relationship)

A handy map...



linearly independent
uncorrelated
orthogonal
X Y
1 2
1 3
2 4
3 5
(1) linearly independ.
(2) not uncorrelated
(3) not orthogonal
X Y
0 1
0 0
1 1
1 0
(1) linearly independ.
(2) uncorrelated
(3) not orthogonal
X Y
-1 1
-1 -1
1 1
1 -1
(1) linearly independ.
(2) uncorrelated
(3) orthogonal
X Y
1 5
-5 1
3 1
-1 3
(1) linearly independ.
(2) not uncorrelated
(3) orthogonal

Discrete-Time Random Processes



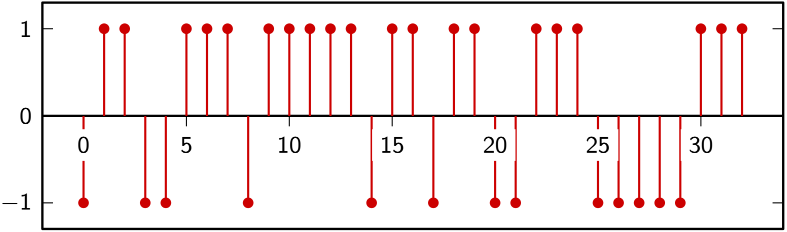
A simple discrete-time random signal generator

For each new sample, toss a fair coin:

$$\check{x}[n] = \begin{cases} +1 & \text{if the outcome of the } n\text{-th toss is head} \\ -1 & \text{if the outcome of the } n\text{-th toss is tail} \end{cases}$$

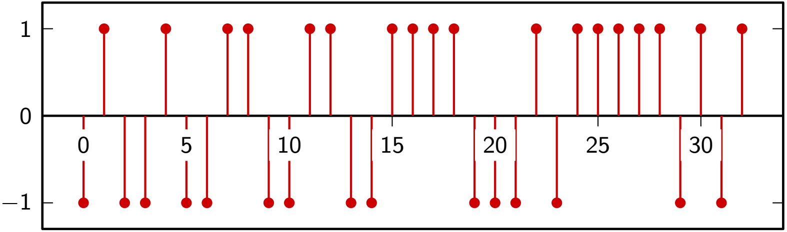
A simple discrete-time random signal generator

every time we turn on the generator we obtain a different *realization* of the signal



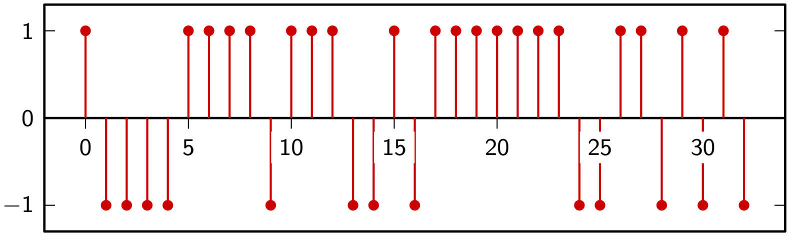
A simple discrete-time random signal generator

every time we turn on the generator we obtain a different *realization* of the signal



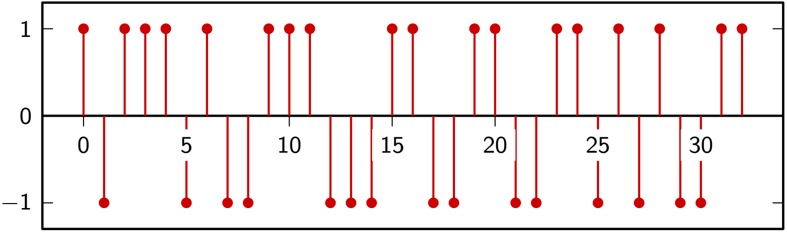
A simple discrete-time random signal generator

every time we turn on the generator we obtain a different *realization* of the signal



A simple discrete-time random signal generator

every time we turn on the generator we obtain a different *realization* of the signal



Discrete-Time Random Processes

DT random processes generate an infinite-length sequence of random values

- what is the distribution of each value ?
- what are the statistical relations between values?

Discrete-Time Random Processes

- infinite-length sequence of *interdependent* random variables
- a full characterization requires knowing

$$f_{x[n_0]x[n_1]\cdots x[n_{k-1}]}(x_0, x_1, \cdots, x_{k-1})$$

  for *all* possible sets of k indices $\{n_0, n_1, \cdots, n_{k-1}\}$ and for *all* $k \in \mathbb{N}$
- clearly too much to handle

k-th order descriptions

- first-order description:
  - $f_{X[n]}(x[n]) \longrightarrow$ time-varying mean $\bar{x}[n] = \mathrm{E}\left[x[n]\right]$
- second-order description:
  - $f_{X[n]}(x[n]) \longrightarrow$ time-varying mean $\bar{x}[n] = \mathrm{E}\left[x[n]\right]$
  - $f_{X[n]X[m]}(x[n], x[m]) \longrightarrow$ time-varying auto-correlation $r_x[n, m] = \mathrm{E}\left[x[n]x[m]\right]$
- third-order description:
  - time-varying mean
  - time-varying auto-correlation
  - $f_{X[n]X[m]X[p]}(x[n], x[m], x[p]) \longrightarrow$ time-varying third moment
- ...

Manageable random processes: 1 – Stationarity

for a stationary process, all partial-order descriptions are **time-invariant**:

$$f_{x[n_0]x[n_1]\cdots x[n_{k-1}]}(\cdots) = f_{x[n_0+M]x[n_1+M]\cdots x[n_{k-1}+M]}(\cdots)$$

Manageable random processes: 1 – Stationarity

For stationary random processes:

- mean is time-invariant: $\mathrm{E}\left[x[n]\right] = m_x$
- autocorrelation depends only on time lag: $\mathrm{E}\left[x[n]x[m]\right] = r_x[n-m]$
- (higher-order moments depend only on relative time differences, etc...)

Manageable random processes: 2 – Wide-Sense Stationarity

For WSS random processes we only care about the first two moments:

- $\mathrm{E}\left[x[n]\right] = m_x$
- $\mathrm{E}\left[x[n]x[m]\right] = r[n-m]$

Manageable random processes: 2 – Wide-Sense Stationarity

Why WSS?

- most stochastic SP techniques use quadratic “cost” functions
- algorithms require only the first and second moments
- quadratic optimization (Mean Square Error) mathematically well-behaved

White Processes (White Noise)

White noise process:

- zero-mean: $\mathrm{E}\left[x[n]\right] = 0$
- uncorrelated: $\mathrm{E}\left[x[n]x[m]\right] = \mathrm{E}\left[x[n]\right]\mathrm{E}\left[x[m]\right]$ for $m \neq n$
- autocorrelation $r_x[n] = \sigma_x^2 \delta[n]$

According to underlying distribution:

- Gaussian white noise
- uniform white noise
- ...

White Processes (White Noise)

White noise process:

- zero-mean: $\mathrm{E}\left[x[n]\right] = 0$
- uncorrelated: $\mathrm{E}\left[x[n]x[m]\right] = \mathrm{E}\left[x[n]\right]\mathrm{E}\left[x[m]\right]$ for $m \neq n$
- autocorrelation $r_x[n] = \sigma_x^2 \delta[n]$

According to underlying distribution:

- Gaussian white noise
- uniform white noise
- ...

The coin-toss process

For each new sample, toss a fair coin:

$$x[n] = \begin{cases} +1 & \text{if the outcome of the } n\text{-th toss is head} \\ -1 & \text{if the outcome of the } n\text{-th toss is tail} \end{cases}$$

- each sample is independent from all others
- each sample value has a 50% probability: $f_x(x) = \delta(x \pm 1)/2$

white noise process with $r_x[n] = \delta[n]$

Computing the moments: theory

With access to the theoretical univariate and bivariate pdfs of the WSS process:

$$m_x = \mathrm{E}\left[x[n]\right] = \int_{-\infty}^{\infty} a f_{x[0]}(a) da$$

$$r_x[k] = \mathrm{E}\left[x[0]x[k]\right] = \iint_{-\infty}^{\infty} ab\, f_{x[0]x[k]}(a, b)\, da\, db$$

Computing the moments: ensemble averages

With access to M realizations of the WSS process:

$$m_x \approx \frac{1}{M} \sum_{i=0}^{M-1} \check{x}_i[n]$$

$$r_x[k] \approx \frac{1}{M} \sum_{i=0}^{M-1} \check{x}_i[n]\check{x}_i[n+k]$$

Computing the moments: time averages

With access to M samples of a single realization of the WSS process:

$$m_x \approx \frac{1}{M} \sum_{n=0}^{M-1} \check{x}[n]$$

$$r_x[k] \approx \frac{1}{M} \sum_{n=0}^{M-|k|-1} \check{x}[n]\check{x}[n+|k|]$$

- processes for which this works are called *ergodic*
- at least $M > 4k_{\max}$

Orthogonality

Correlation (via ensemble average):

$$\mathrm{E}\left[xy\right] = \lim_{M\to\infty} \frac{1}{M} \sum_{i=0}^{M-1} \breve{x}_i \, \breve{y}_i .$$

Inner product in $\ell_2(\mathbb{Z})$:

$$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{n=-\infty}^{\infty} x_n y_n .$$

Orthogonality

Correlation (via ensemble average):

$$\mathrm{E}\left[xy\right] = \lim_{M\to\infty} \frac{1}{M} \sum_{i=0}^{M-1} \breve{x}_i \, \breve{y}_i .$$

Inner product in $\ell_2(\mathbb{Z})$:

$$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{n=-\infty}^{\infty} x_n y_n .$$

Orthogonality

$$\mathrm{E}[xy] = 0 \quad \Longrightarrow \quad \textit{orthogonal} \text{ random variables}$$

- if x, y zero mean: orthogonal $=$ uncorrelated
- no linear relationship between variables
- variables are maximally different

spectral representation of random processes

A simple discrete-time random signal generator

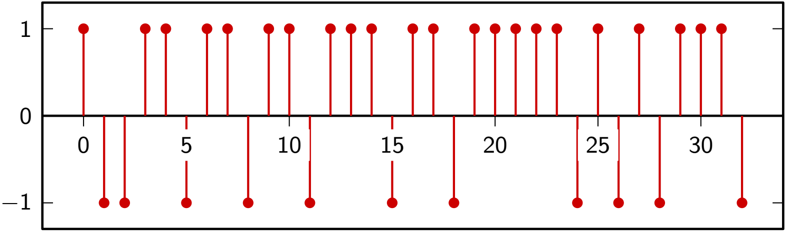
For each new sample, toss a fair coin:

$$x[n] = \begin{cases} +1 & \text{if the outcome of the } n\text{-th toss is head} \\ -1 & \text{if the outcome of the } n\text{-th toss is tail} \end{cases}$$

- each sample is independent from all others
- each sample value has a 50% probability

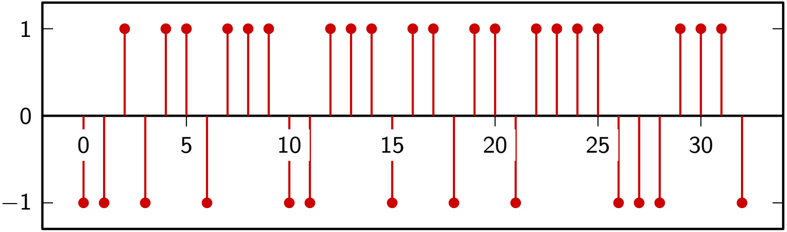
A simple discrete-time random signal generator

every time we turn on the generator we obtain a different *realization* of the signal



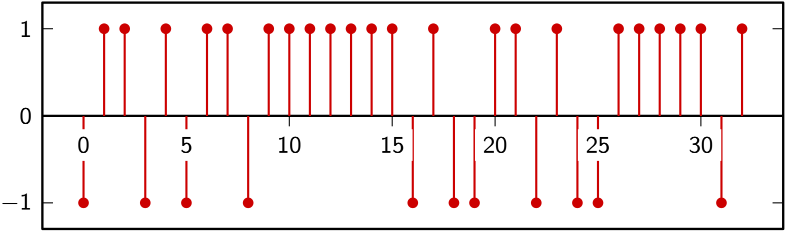
A simple discrete-time random signal generator

every time we turn on the generator we obtain a different *realization* of the signal



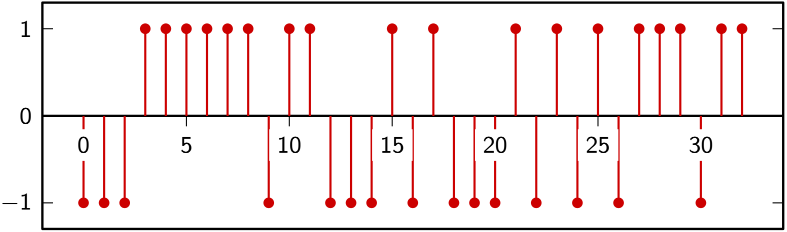
A simple discrete-time random signal generator

every time we turn on the generator we obtain a different *realization* of the signal



A simple discrete-time random signal generator

every time we turn on the generator we obtain a different *realization* of the signal

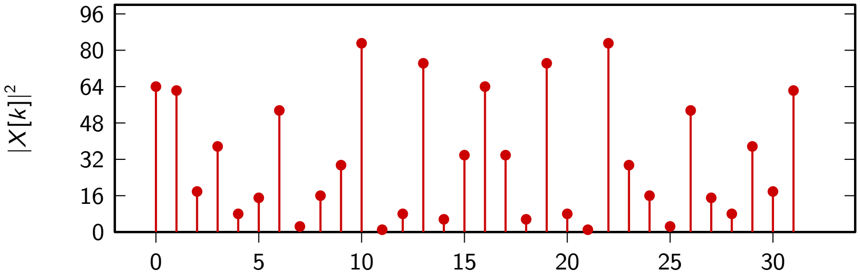


A simple discrete-time random signal generator

- every time we turn on the generator we obtain a different *realization* of the signal
- we know the "mechanism" behind each instance
- but how can we analyze a random signal? What about its frequency content?

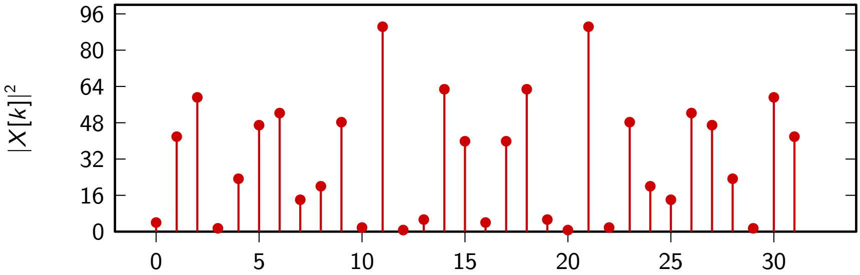
Spectral properties?

let's try with the DFT of a finite set of random samples



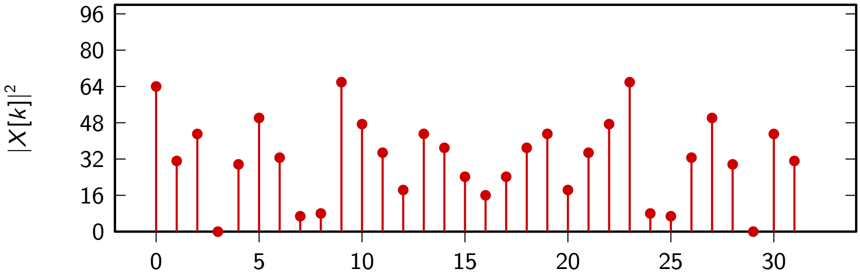
Spectral properties?

let's try with the DFT of a finite set of random samples



Spectral properties?

let's try with the DFT of a finite set of random samples

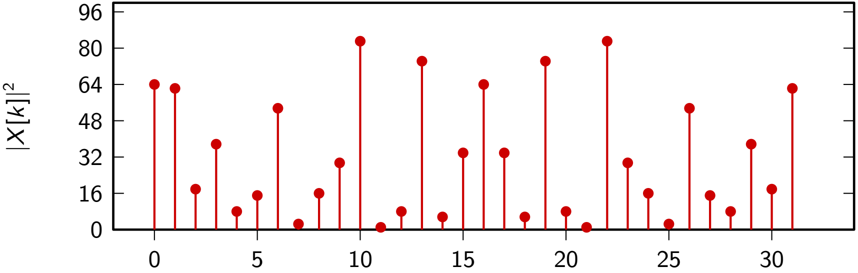


Spectral properties?

every time it's different; maybe with more data?

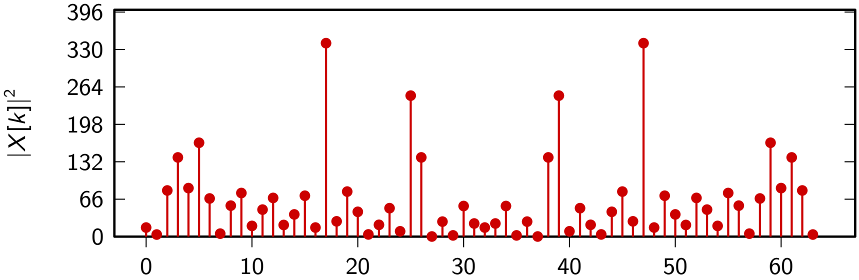
Spectral properties?

DFT of an increasing number of samples

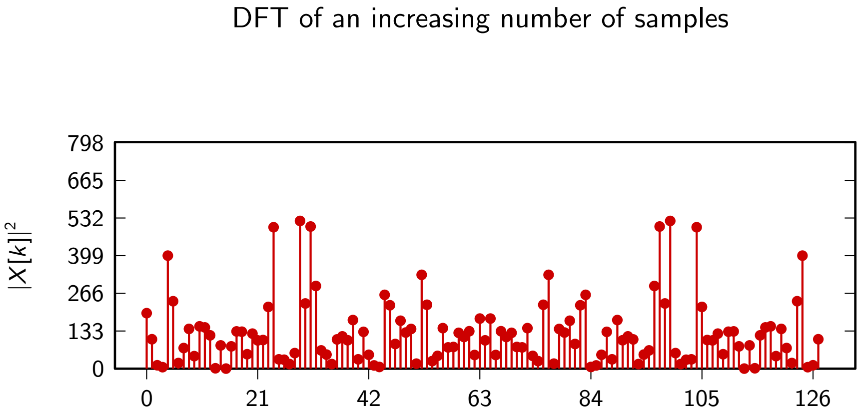


Spectral properties?

DFT of an increasing number of samples



Spectral properties?



Averaging

- DFTs of realizations show no clear pattern... we need a new strategy
- when faced with random data an intuitive response is to take "averages" (i.e. expectation)
- for the coin-toss signal:

$$\mathrm{E}\left[x[n]\right] = -1 \cdot P[\text{n-th toss is tail}] + 1 \cdot P[\text{n-th toss is head}] = 0$$

- so the average value for each sample is zero...

Averaging

- DFTs of realizations show no clear pattern... we need a new strategy
- when faced with random data an intuitive response is to take "averages" (i.e. expectation)
- for the coin-toss signal:

$$\mathrm{E}\left[x[n]\right] = -1 \cdot P[\text{n-th toss is tail}] + 1 \cdot P[\text{n-th toss is head}] = 0$$

- so the average value for each sample is zero...

Averaging

- DFTs of realizations show no clear pattern… we need a new strategy
- when faced with random data an intuitive response is to take "averages" (i.e. expectation)
- for the coin-toss signal:

$$\mathrm{E}\left[x[n]\right] = -1 \cdot P[\text{n-th toss is tail}] + 1 \cdot P[\text{n-th toss is head}] = 0$$

- so the average value for each sample is zero…

Averaging

- DFTs of realizations show no clear pattern... we need a new strategy
- when faced with random data an intuitive response is to take "averages" (i.e. expectation)
- for the coin-toss signal:

$$\mathrm{E}\left[x[n]\right] = -1 \cdot P[\text{n-th toss is tail}] + 1 \cdot P[\text{n-th toss is head}] = 0$$

- so the average value for each sample is zero...

Averaging the DFT

- but the DFT is linear so $\mathrm{E}[\mathrm{DFT}\{x[n]\}] = \mathrm{DFT}\{\mathrm{E}[x[n]]\} = 0$
- however the signal "moves", so its energy or power must be nonzero

Averaging the DFT

- but the DFT is linear so $\mathrm{E}\left[\mathrm{DFT}\left\{x[n]\right\}\right] = \mathrm{DFT}\left\{\mathrm{E}\left[x[n]\right]\right\} = 0$
- however the signal "moves", so its energy or power must be nonzero

Energy and power

- the coin-toss process produces realizations with infinite energy:

$$E_x = \lim_{N\to\infty} \sum_{n=-N}^{N} |\check{x}[n]|^2 = \lim_{N\to\infty} (2N+1) = \infty$$

- which, however, have has finite *power*:

$$P_x = \lim_{N\to\infty} \frac{1}{2N+1} \sum_{n=-N}^{N} |\check{x}[n]|^2 = 1$$

Averaging

let's try to average the DFT's square magnitude, normalized:

- pick an interval length N
- pick a number of iterations M
- run the signal generator M times and obtain M N-point realizations
- compute the DFT of each realization
- average their square magnitude divided by N

Averaging

let's try to average the DFT's square magnitude, normalized:

- pick an interval length N
- pick a number of iterations M
- run the signal generator M times and obtain M N-point realizations
- compute the DFT of each realization
- average their square magnitude divided by N

Averaging

let's try to average the DFT's square magnitude, normalized:

- pick an interval length N
- pick a number of iterations M
- run the signal generator M times and obtain M N-point realizations
- compute the DFT of each realization
- average their square magnitude divided by N

Averaging

let's try to average the DFT's square magnitude, normalized:

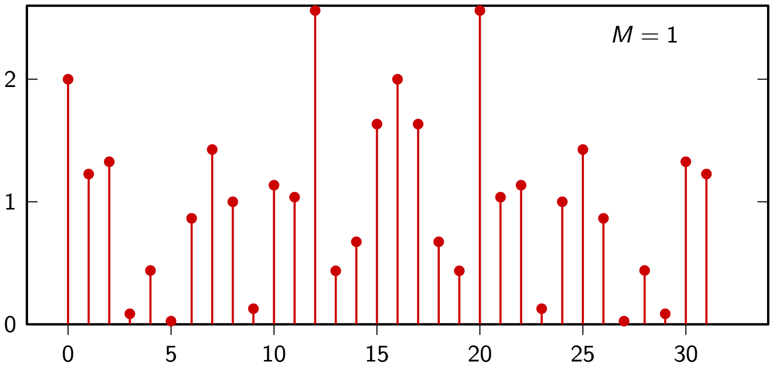
- pick an interval length N
- pick a number of iterations M
- run the signal generator M times and obtain M N-point realizations
- compute the DFT of each realization
- average their square magnitude divided by N

Averaging

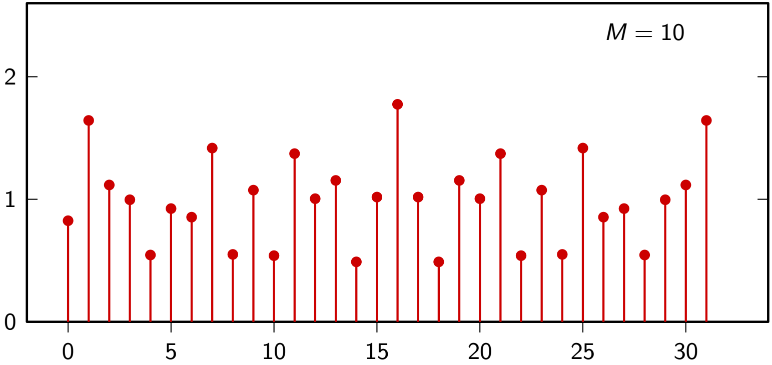
let's try to average the DFT's square magnitude, normalized:

- pick an interval length N
- pick a number of iterations M
- run the signal generator M times and obtain M N-point realizations
- compute the DFT of each realization
- average their square magnitude divided by N

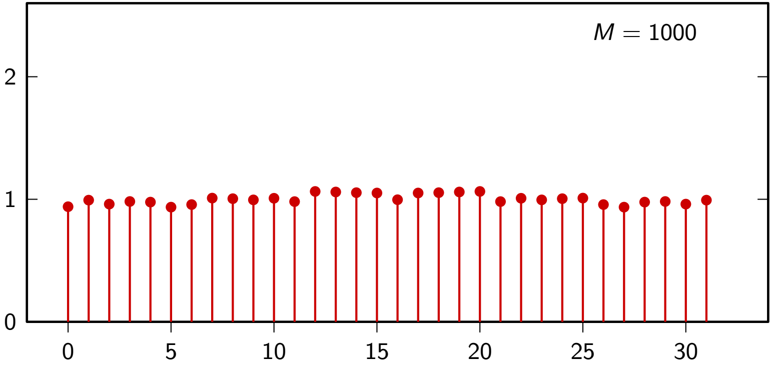
Averaged DFT square magnitude



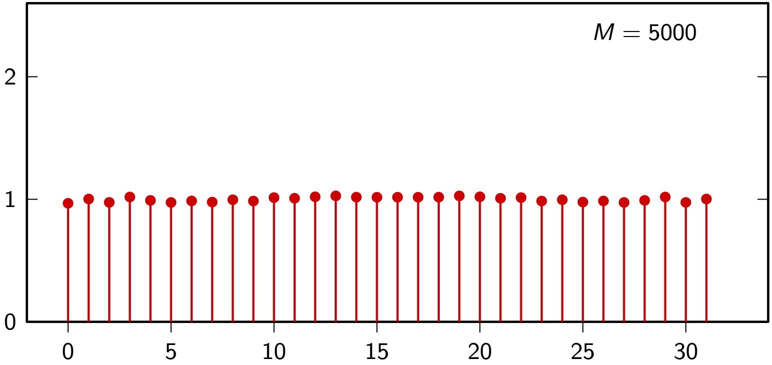
Averaged DFT square magnitude



Averaged DFT square magnitude



Averaged DFT square magnitude



Power spectral density

$$P[k] = \mathrm{E}\left[|X_N[k]|^2/N\right]$$

- it looks very much as if $P[k] = 1$
- if $|X_N[k]|^2$ tends to the *energy* distribution in frequency...
- ...$|X_N[k]|^2/N$ tends to the *power* distribution (aka *density*) in frequency
- the frequency-domain representation for stochastic processes is the power spectral density

Power spectral density

$$P[k] = \mathrm{E}\left[|X_N[k]|^2/N\right]$$

- it looks very much as if $P[k] = 1$
- if $|X_N[k]|^2$ tends to the *energy* distribution in frequency...
- ...$|X_N[k]|^2/N$ tends to the *power* distribution (aka *density*) in frequency
- the frequency-domain representation for stochastic processes is the power spectral density

Power spectral density

$$P[k] = \mathrm{E}\left[|X_N[k]|^2/N\right]$$

- it looks very much as if $P[k] = 1$
- if $|X_N[k]|^2$ tends to the *energy* distribution in frequency...
- ...$|X_N[k]|^2/N$ tends to the *power* distribution (aka *density*) in frequency
- the frequency-domain representation for stochastic processes is the power spectral density

Power spectral density

$$P[k] = \mathrm{E}\left[|X_N[k]|^2/N\right]$$

- it looks very much as if $P[k] = 1$
- if $|X_N[k]|^2$ tends to the *energy* distribution in frequency...
- ...$|X_N[k]|^2/N$ tends to the *power* distribution (aka *density*) in frequency
- the frequency-domain representation for stochastic processes is the power spectral density

Power spectral density: intuition

- $P[k] = 1$ means that the power is equally distributed over all frequencies
- i.e., we cannot predict if the signal moves "slowly" or "super-fast"
- this is because each sample is independent of each other: we could have a realization of all ones or a realization in which the sign changes every other sample or anything in between

Power spectral density: intuition

- $P[k] = 1$ means that the power is equally distributed over all frequencies
- i.e., we cannot predict if the signal moves "slowly" or "super-fast"
- this is because each sample is independent of each other: we could have a realization of all ones or a realization in which the sign changes every other sample or anything in between

Power spectral density: intuition

- $P[k] = 1$ means that the power is equally distributed over all frequencies
- i.e., we cannot predict if the signal moves "slowly" or "super-fast"
- this is because each sample is independent of each other: we could have a realization of all ones or a realization in which the sign changes every other sample or anything in between

Filtering a random process

- let's filter the random process with a 2-point Moving Average filter
- $y[n] = (x[n] + x[n-1])/2$
- what is the power spectral density?

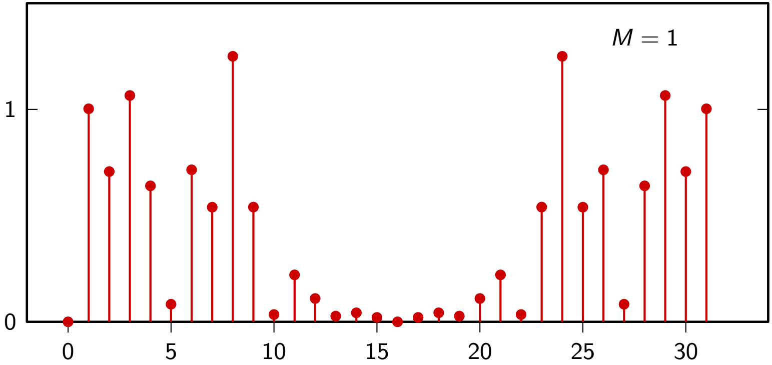
Filtering a random process

- let's filter the random process with a 2-point Moving Average filter
- $y[n] = (x[n] + x[n-1])/2$
- what is the power spectral density?

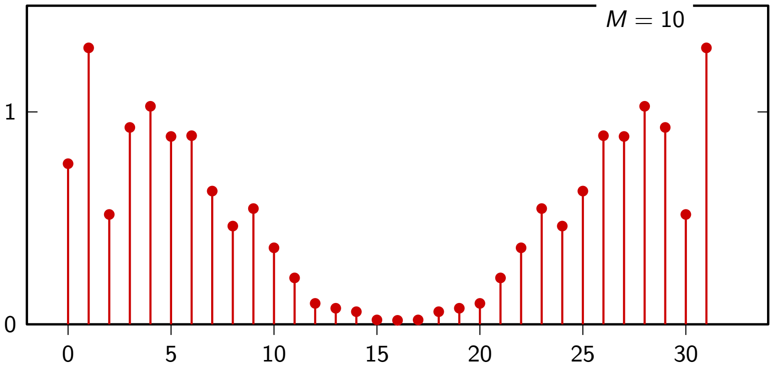
Filtering a random process

- let's filter the random process with a 2-point Moving Average filter
- $y[n] = (x[n] + x[n-1])/2$
- what is the power spectral density?

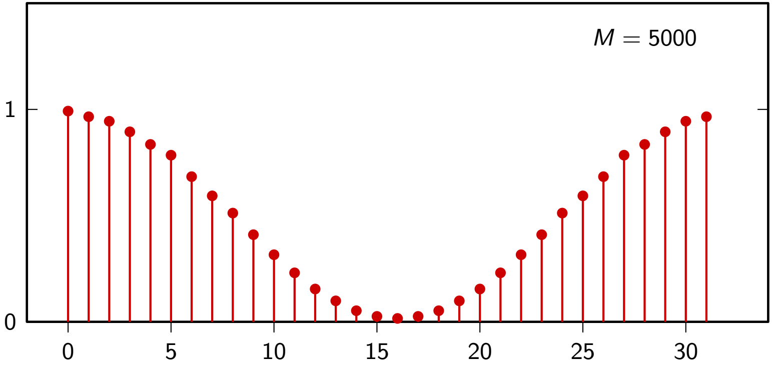
Averaged DFT magnitude of filtered process



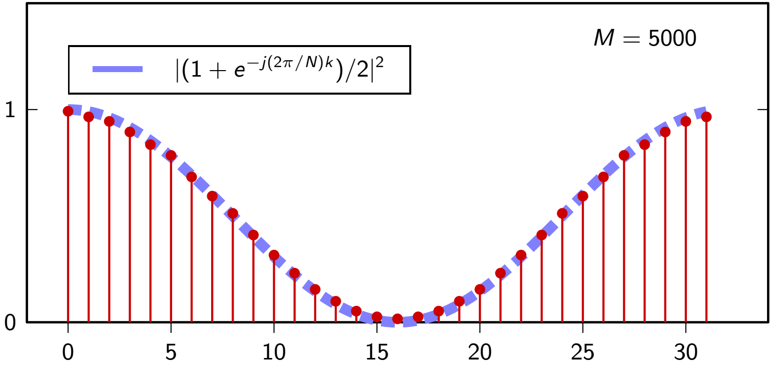
Averaged DFT magnitude of filtered process



Averaged DFT magnitude of filtered process



Averaged DFT magnitude of filtered process



Filtering a random process

- it looks like $P_y[k] = P_x[k] \, |H[k]|^2$, where $H[k] = \mathrm{DFT}\{h[n]\}$
- can we generalize these results beyond a finite set of samples?

Filtering a random process

- it looks like $P_y[k] = P_x[k]\,|H[k]|^2$, where $H[k] = \text{DFT}\,\{h[n]\}$
- can we generalize these results beyond a finite set of samples?

Energy and Power Signals

- energy signals: $\sum_{n=-\infty}^{\infty} |x[n]|^2 < \infty$
- power signals: $\lim_{N\to\infty} \frac{1}{2N+1} \sum_{n=-N}^{N} |x[n]|^2 < \infty$

Energy Signals

- finite support, sinc(n), $\alpha^n u[n]$ for $|\alpha| < 1$, ...
- DTFT is well defined
- DTFT square magnitude is *energy* distribution in frequency

Power Signals

- $x[n] = 1,\ u[n],\ e^{j\omega n},\ \sin, \cos,\ \ldots$
- DTFT uses the Dirac delta formalism
- "DTFT square magnitude" doesn't make sense!

Power Spectral Density

Consider a truncated DTFT

$$X_N(e^{j\omega}) = \sum_{n=-N}^{N} x[n]e^{-j\omega n}$$

define the power spectral density of a signal as:

$$P(e^{j\omega}) = \lim_{N\to\infty} \frac{1}{2N+1} |X_N(e^{j\omega})|^2$$

Power Spectral Density

Consider a truncated DTFT

$$X_N(e^{j\omega}) = \sum_{n=-N}^{N} x[n]e^{-j\omega n}$$

define the power spectral density of a signal as:

$$P(e^{j\omega}) = \lim_{N\to\infty} \frac{1}{2N+1}|X_N(e^{j\omega})|^2$$

Power Spectral Density

Examples:

- $x[n] = a$, $P_x(e^{j\omega}) = a^2\tilde{\delta}(\omega)$
- $x[n] = ae^{j\sigma n}$, $P_x(e^{j\omega}) = a^2\tilde{\delta}(\omega - \sigma)$
- $x[n] = au[n]$, $P_x(e^{j\omega}) = (a^2/2)\tilde{\delta}(\omega)$

Power Spectral Density

Examples:

- $x[n] = a$, $P_x(e^{j\omega}) = a^2\tilde{\delta}(\omega)$
- $x[n] = ae^{j\sigma n}$, $P_x(e^{j\omega}) = a^2\tilde{\delta}(\omega - \sigma)$
- $x[n] = au[n]$, $P_x(e^{j\omega}) = (a^2/2)\tilde{\delta}(\omega)$

Power Spectral Density

Examples:

- $x[n] = a$, $P_x(e^{j\omega}) = a^2\tilde{\delta}(\omega)$
- $x[n] = ae^{j\sigma n}$, $P_x(e^{j\omega}) = a^2\tilde{\delta}(\omega - \sigma)$
- $x[n] = au[n]$, $P_x(e^{j\omega}) = (a^2/2)\tilde{\delta}(\omega)$

Power Spectral Density for WSS Processes

For a random process

$$P_x(e^{j\omega}) = \lim_{N\to\infty} \frac{1}{2N+1} \mathrm{E}\left[|X_N(e^{j\omega})|^2 \right]$$

Power Spectral Density for WSS Processes

$$\mathrm{E}\left[\left|\sum_{n=-N}^{N} x[n]e^{-j\omega n}\right|^2\right] = \mathrm{E}\left[\sum_{n=-N}^{N} x[n]e^{j\omega n} \sum_{m=-N}^{N} x[m]e^{-j\omega m}\right]$$

Power Spectral Density for WSS Processes

$$\mathrm{E}\left[\left|\sum_{n=-N}^{N} x[n]e^{-j\omega n}\right|^2\right] = \mathrm{E}\left[\sum_{n=-N}^{N} x[n]e^{j\omega n} \sum_{m=-N}^{N} x[m]e^{-j\omega m}\right]$$

$$= \sum_{n=-N}^{N} \sum_{m=-N}^{N} \mathrm{E}\left[x[n]x[m]\right] e^{-j\omega(m-n)}$$

Power Spectral Density for WSS Processes

$$
\begin{aligned}
\mathrm{E}\left[\left|\sum_{n=-N}^{N} x[n]e^{-j\omega n}\right|^2\right] &= \mathrm{E}\left[\sum_{n=-N}^{N} x[n]e^{j\omega n} \sum_{m=-N}^{N} x[m]e^{-j\omega m}\right] \\
&= \sum_{n=-N}^{N} \sum_{m=-N}^{N} \mathrm{E}\left[x[n]x[m]\right] e^{-j\omega(m-n)} \\
&= \sum_{n=-N}^{N} \sum_{m=-N}^{N} \underbrace{\boxed{r_x[m-n]}}_{\text{WSS}} e^{-j\omega(m-n)}
\end{aligned}
$$

A clever manipulation

$$S = \sum_{m=-N}^{N} \sum_{n=-N}^{N} f(m-n)$$

$$-2N \leq (m-n) \leq 2N$$

$$S = \sum_{k=-2N}^{2N} c(k) f(k)$$

A clever manipulation

$$S = \sum_{m=-N}^{N} \sum_{n=-N}^{N} f(m-n)$$

$$-2N \leq (m-n) \leq 2N$$

$$S = \sum_{k=-2N}^{2N} c(k) f(k)$$

A clever manipulation

$$S = \sum_{m=-N}^{N} \sum_{n=-N}^{N} f(m-n)$$

$$-2N \leq (m-n) \leq 2N$$

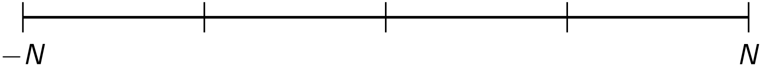
$$S = \sum_{k=-2N}^{2N} c(k) f(k)$$

A clever manipulation

$c(k)$: number of ways we can pick n, m in $[-N, N]$ so that $(m - n) = k$

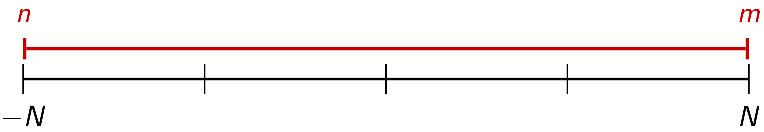
A clever manipulation

geometrically: $c(k) =$ number of ways we can fit a segment of length k over $[-N, N]$:



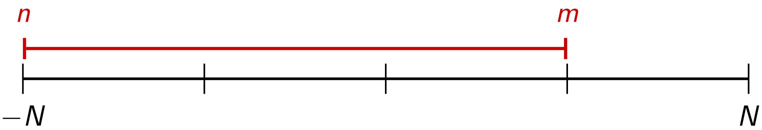
A clever manipulation

geometrically: $c(k) =$ number of ways we can fit a segment of length k over $[-N, N]$:



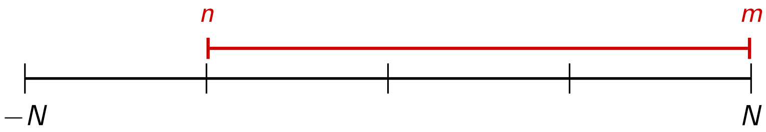
A clever manipulation

geometrically: $c(k) =$ number of ways we can fit a segment of length k over $[-N, N]$:



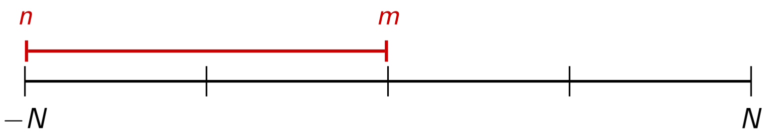
A clever manipulation

geometrically: $c(k)$ = number of ways we can fit a segment of length k over $[-N, N]$:



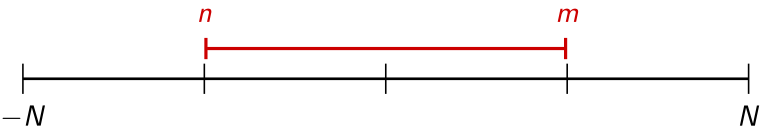
A clever manipulation

geometrically: $c(k) =$ number of ways we can fit a segment of length k over $[-N, N]$:



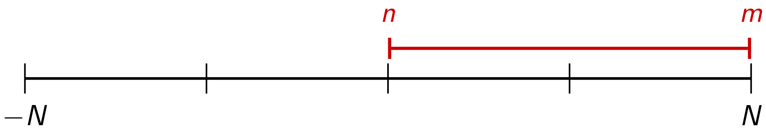
A clever manipulation

geometrically: $c(k)$ = number of ways we can fit a segment of length k over $[-N, N]$:



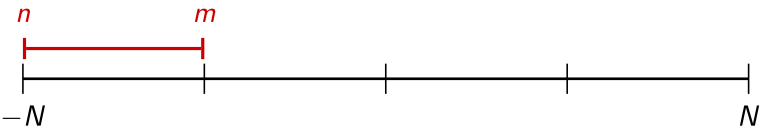
A clever manipulation

geometrically: $c(k) =$ number of ways we can fit a segment of length k over $[-N, N]$:



A clever manipulation

geometrically: $c(k) =$ number of ways we can fit a segment of length k over $[-N, N]$:



A clever manipulation

$c(k)$: number of ways we can pick n, m in $[-N, N]$ so that $(m - n) = k$

$$c(k) = 2N + 1 - |k|$$

Power Spectral Density for WSS Processes

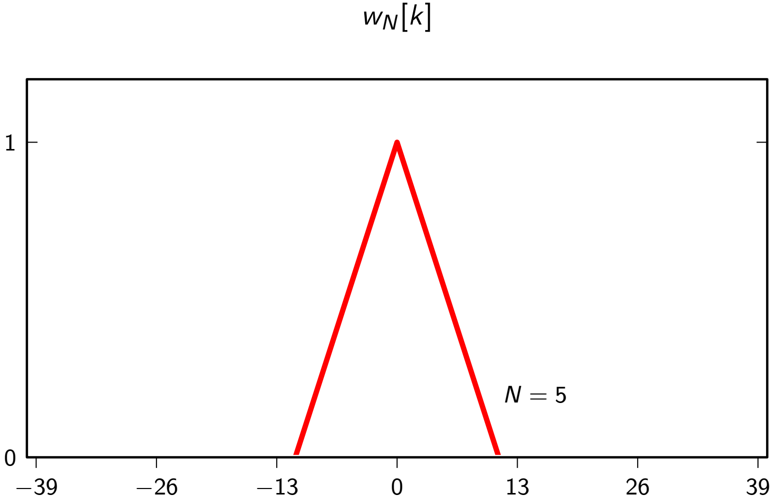
$$\mathrm{E}\left[\left|\sum_{n=-N}^{N} x[n]e^{-j\omega n}\right|^2\right] = \sum_{k=-2N}^{2N} (2N+1-|k|)\; r_x[k]e^{-j\omega k}$$

Power Spectral Density for WSS Processes

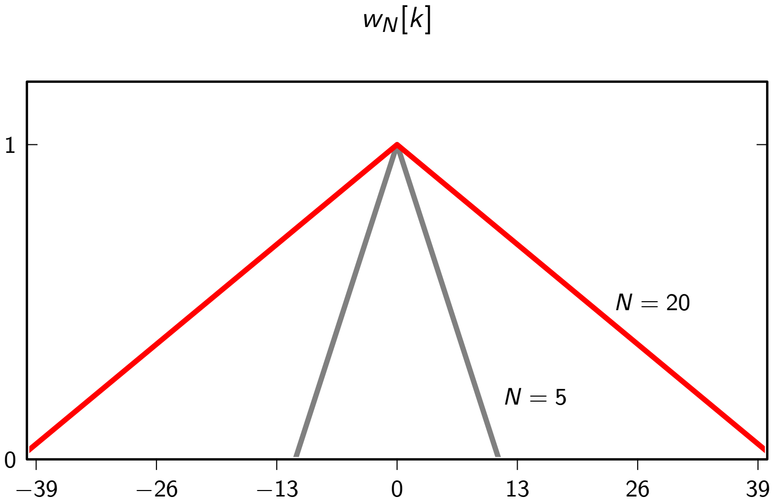
$$P_x(e^{j\omega}) = \lim_{N\to\infty} \sum_{k=-2N}^{2N} \left(\frac{2N+1-|k|}{2N+1} \right) \left(r_X[k] e^{-j\omega k} \right)$$

$$= \lim_{N\to\infty} \sum_{k=-2N}^{2N} w_N[k] r_x[k] e^{-j\omega k}$$

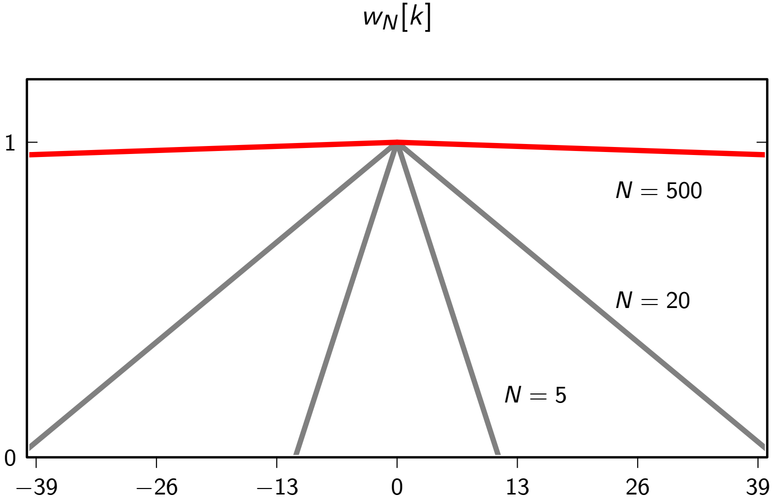
Power Spectral Density for WSS Processes



Power Spectral Density for WSS Processes



Power Spectral Density for WSS Processes



Power Spectral Density for WSS Processes

$$\lim_{N\to\infty} w_N[k] = 1$$

Power Spectral Density for WSS Processes

$$
\begin{aligned}
P_x(e^{j\omega}) &= \lim_{N\to\infty} \sum_{k=-2N}^{2N} w_N[k] r_x[k] e^{-j\omega k} \\
&= \sum_{k=-\infty}^{\infty} r_X[k] e^{-j\omega k} \\
&= \mathrm{DTFT}\{r_x[k]\}
\end{aligned}
$$

Noise

- noise is everywhere:
  - thermal noise
  - sum of extraneous interferences
  - quantization and numerical errors
  - ...
- we can model noise as a stochastic signal
- the most important noise is white noise

Noise

- noise is everywhere:
  - thermal noise
  - sum of extraneous interferences
  - quantization and numerical errors
  - ...
- we can model noise as a stochastic signal
- the most important noise is white noise

Noise

- noise is everywhere:
  - thermal noise
  - sum of extraneous interferences
  - quantization and numerical errors
  - ...
- we can model noise as a stochastic signal
- the most important noise is white noise

Noise

- noise is everywhere:
  - thermal noise
  - sum of extraneous interferences
  - quantization and numerical errors
  - ...
- we can model noise as a stochastic signal
- the most important noise is white noise

Noise

- noise is everywhere:
  - thermal noise
  - sum of extraneous interferences
  - quantization and numerical errors
  - ...
- we can model noise as a stochastic signal
- the most important noise is white noise

Noise

- noise is everywhere:
  - thermal noise
  - sum of extraneous interferences
  - quantization and numerical errors
  - ...
- we can model noise as a stochastic signal
- the most important noise is white noise

Noise

- noise is everywhere:
  - thermal noise
  - sum of extraneous interferences
  - quantization and numerical errors
  - ...
- we can model noise as a stochastic signal
- the most important noise is white noise

PSD of white noise

White noise:

- $m = 0$
- $r[k] = \sigma^2 \delta[k]$

$$P(e^{j\omega}) = \sigma^2$$

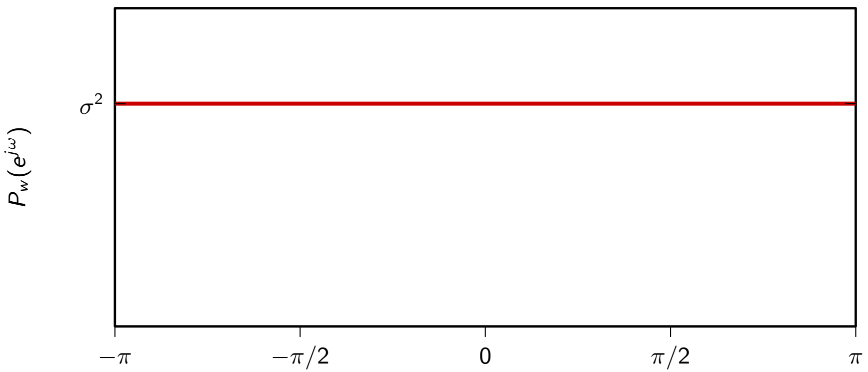
PSD of white noise

White noise:

- $m = 0$
- $r[k] = \sigma^2 \delta[k]$

$$P(e^{j\omega}) = \sigma^2$$

PSD of white noise



White noise

- the PSD is independent of the probability distribution of the single samples (depends only on the variance)
- distribution is important to estimate bounds for the signal
- very often a Gaussian distribution models the experimental data the best
- AWGN: additive white Gaussian noise

White noise

- the PSD is independent of the probability distribution of the single samples (depends only on the variance)
- distribution is important to estimate bounds for the signal
- very often a Gaussian distribution models the experimental data the best
- AWGN: additive white Gaussian noise

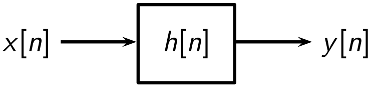
White noise

- the PSD is independent of the probability distribution of the single samples (depends only on the variance)
- distribution is important to estimate bounds for the signal
- very often a Gaussian distribution models the experimental data the best
- AWGN: additive white Gaussian noise

White noise

- the PSD is independent of the probability distribution of the single samples (depends only on the variance)
- distribution is important to estimate bounds for the signal
- very often a Gaussian distribution models the experimental data the best
- AWGN: additive white Gaussian noise

Filtering a Random Process



- is $y[n]$ a random process?
- if $x[n]$ WSS, is $y[n]$ WSS?
- what are m_y and $r_y[n]$?

Filtering a Random Process

$$x[n] \longrightarrow \boxed{h[n]} \longrightarrow y[n]$$

- is $y[n]$ a random process?
- if $x[n]$ WSS, is $y[n]$ WSS?
- what are m_y and $r_y[n]$?

Filtering a Random Process

$$x[n] \longrightarrow \boxed{h[n]} \longrightarrow y[n]$$

- is $y[n]$ a random process?
- if $x[n]$ WSS, is $y[n]$ WSS?
- what are m_y and $r_y[n]$?

Mean of the Filtered Process

$$
\begin{aligned}
m_{y[n]} &= \mathrm{E}\left[y[n]\right] = \mathrm{E}\left[\sum_{k=-\infty}^{\infty} h[k]x[n-k]\right] \\
&= \sum_{k=-\infty}^{\infty} h[k]\mathrm{E}\left[x[n-k]\right] \\
&= \sum_{k=-\infty}^{\infty} h[k]m_x \qquad (x[n] \text{ is WSS}) \\
&= m_x \sum_{k=-\infty}^{\infty} h[k] \\
&= m_x H(e^{j0})
\end{aligned}
$$

Autocorrelation of the Filtered Process

$$\begin{aligned}
\mathrm{E}\left[y[n]y[m]\right] &= \mathrm{E}\left[\sum_{k=-\infty}^{\infty} h[k]x[n-k] \sum_{i=-\infty}^{\infty} h[i]x[m-i]\right] \\
&= \sum_{k=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} h[k]h[i]\mathrm{E}\left[x[n-k]x[m-i]\right] \\
&= \sum_{k=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} h[k]h[i]r_x[(n-m)-(k+i)]
\end{aligned}$$

output depends only on lag $(n-m) \longrightarrow y[n]$ is WSS

Fundamental Result

with a change of variable in the double sum:

$$r_y[n] = h[n] * h[-n] * r_x[n]$$

so that:

$$P_y(e^{j\omega}) \;=\; \left|H(e^{j\omega})\right|^2 P_x(e^{j\omega})$$

Deterministic filters can be used to shape the power distribution of WSS random processes

Stochastic signal processing

key points:

- filters designed for deterministic signals still work (in magnitude) in the stochastic case
- we lose the concept of phase since we don't know the shape of a realization in advance

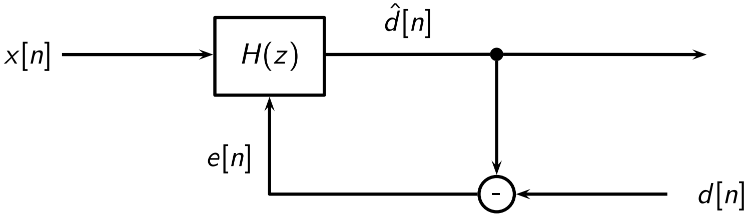
Stochastic signal processing

key points:

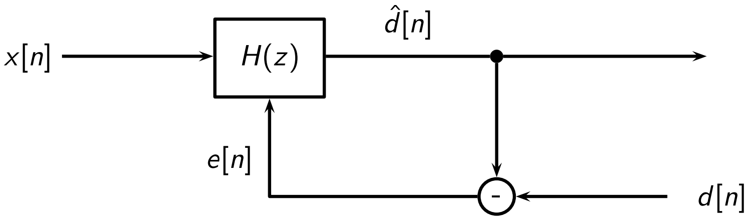
- filters designed for deterministic signals still work (in magnitude) in the stochastic case
- we lose the concept of phase since we don't know the shape of a realization in advance

adaptive signal processing

Adaptive signal processing

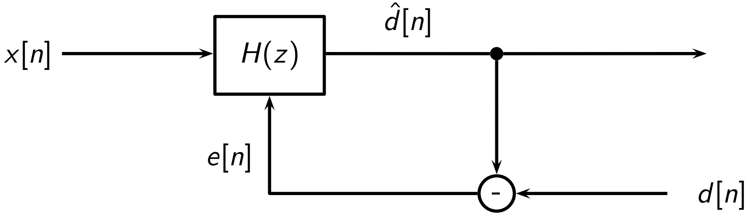


Adaptive signal processing



- $d[n]$: desired signal
- $\hat{d}[n]$: adaptive approximation
- $e[n]$: error signal

Adaptive signal processing



how do we find the filter's coefficients?

Optimal adaptive filter

optimal filter $H(z)$ *minimizes* the Mean Square Error

$$H(z) = \underset{H(z)}{\arg\min} \left\{ \mathrm{E}\left[|e[n]|^2 \right] \right\}$$

Optimal adaptive filter

$$H(z) = \arg\min_{H(z)} \left\{ \mathrm{E}\left[|e[n]|^2 \right] \right\}$$

Advantages of a squared error measure:

- minimum always exist
- error easily differentiable
- output will be orthogonal to error
- only need second moments!

Just FIR adaptive filters for us

Will only consider FIR adaptive filters:

$$\hat{d}[n] = \sum_{k=0}^{N-1} h[k]x[n-k]$$

Finding the minimum squared error

Two cases:

- for WSS signals, one-shot solution: Optimal Least Squares
- for "almost" WSS signals, iterative solutions: stochastic gradient descent or LMS

Optimal Least Squares

$$e[n] = d[n] - \sum_{k=0}^{N-1} h[k]x[n-k]$$

Minimum is found by setting all partial derivatives to zero

$$\frac{\partial \mathrm{E}\left[e^2[n]\right]}{\partial h[i]} = 2\mathrm{E}\left[e[n]\,\frac{\partial e[n]}{\partial h[i]}\right]$$
$$= -2\mathrm{E}\left[e[n]\,x[n-i]\right] = 0$$

Optimal Least Squares

$$e[n] = d[n] - \sum_{k=0}^{N-1} h[k]x[n-k]$$

Minimum is found by setting all partial derivatives to zero

$$\frac{\partial \mathrm{E}\left[e^2[n]\right]}{\partial h[i]} = 2\mathrm{E}\left[e[n]\,\frac{\partial e[n]}{\partial h[i]}\right]$$

$$= -2\mathrm{E}\left[e[n]\,x[n-i]\right] = 0$$

Optimal Least Squares

$$e[n] = d[n] - \sum_{k=0}^{N-1} h[k]x[n-k]$$

Minimum is found by setting all partial derivatives to zero

$$\begin{aligned}\frac{\partial \mathrm{E}\left[e^2[n]\right]}{\partial h[i]} &= 2\mathrm{E}\left[e[n]\,\frac{\partial e[n]}{\partial h[i]}\right] \\ &= -2\mathrm{E}\left[e[n]\,x[n-i]\right] = 0\end{aligned}$$

Orthogonality principle

$$\mathrm{E}\left[e[n]\, x[n-i]\right] = 0$$

error is orthogonal to all input values we used:
all useful information has been extracted!

Optimal Least Squares

$$e[n] = d[n] - \sum_{k=0}^{N-1} h[k]x[n-k]$$

$$\begin{aligned}
\frac{1}{2}\frac{\partial \mathrm{E}\left[e^2[n]\right]}{\partial h[i]} &= -\mathrm{E}\left[e[n]\,x[n-i]\right] \\
&= \mathrm{E}\left[\sum_{k=0}^{N-1} h[k]x[n-k]x[n-i]\right] - \mathrm{E}\left[d[n]x[n-i]\right] \\
&= \sum_{k=0}^{N-1} h[k]r_x[i-k] - r_{dx}[i] \qquad \textit{(WSS signals)}
\end{aligned}$$

Optimal Least Squares

$$e[n] = d[n] - \sum_{k=0}^{N-1} h[k]x[n-k]$$

$$\begin{aligned}
\frac{1}{2}\frac{\partial \mathrm{E}\left[e^2[n]\right]}{\partial h[i]} &= -\mathrm{E}\left[e[n]\,x[n-i]\right] \\
&= \mathrm{E}\left[\sum_{k=0}^{N-1} h[k]x[n-k]x[n-i]\right] - \mathrm{E}\left[d[n]x[n-i]\right] \\
&= \sum_{k=0}^{N-1} h[k]r_x[i-k] - r_{dx}[i] \qquad \textit{(WSS signals)}
\end{aligned}$$

Optimal Least Squares

$$e[n] = d[n] - \sum_{k=0}^{N-1} h[k]x[n-k]$$

$$\begin{aligned}
\frac{1}{2}\frac{\partial \mathrm{E}\left[e^2[n]\right]}{\partial h[i]} &= -\mathrm{E}\left[e[n]\,x[n-i]\right] \\
&= \mathrm{E}\left[\sum_{k=0}^{N-1} h[k]x[n-k]x[n-i]\right] - \mathrm{E}\left[d[n]x[n-i]\right] \\
&= \sum_{k=0}^{N-1} h[k]r_x[i-k] - r_{dx}[i] \qquad \textit{(WSS signals)}
\end{aligned}$$

Optimal Least Squares

$$e[n] = d[n] - \sum_{k=0}^{N-1} h[k]x[n-k]$$

$$\begin{aligned}
\frac{1}{2}\frac{\partial \mathrm{E}\left[e^2[n]\right]}{\partial h[i]} &= -\mathrm{E}\left[e[n]\,x[n-i]\right] \\
&= \mathrm{E}\left[\sum_{k=0}^{N-1} h[k]x[n-k]x[n-i]\right] - \mathrm{E}\left[d[n]x[n-i]\right] \\
&= \sum_{k=0}^{N-1} h[k]r_x[i-k] - r_{dx}[i] \qquad \textit{(WSS signals)}
\end{aligned}$$

Optimal Least Squares

setting all partial derivatives to zero:

$$\sum_{k=0}^{N-1} h[k] r_x[i-k] = r_{dx}[i]$$

in matrix form:

$$\mathbf{Rh} = \mathbf{g}$$

Optimal Least Squares

$$\mathbf{h} = \begin{bmatrix} h[0] & h[1] & h[2] & \dots & h[N-1] \end{bmatrix}^T$$

$$\mathbf{R} = \begin{bmatrix} r_x[0] & r_x[1] & r_x[2] & \dots & r_x[N-1] \\ r_x[1] & r_x[0] & r_x[1] & \dots & r_x[N-2] \\ r_x[2] & r_x[1] & r_x[0] & \dots & r_x[N-3] \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_x[N-1] & r_x[N-2] & \dots & \dots & r_x[0] \end{bmatrix}$$

$$\mathbf{g} = \begin{bmatrix} r_{dx}[0] & r_{dx}[1] & r_{dx}[2] & \dots & r_{dx}[N-1] \end{bmatrix}^T$$

Error surface ($N = 2$)

$$
\begin{aligned}
J &= \mathrm{E}\left[e^2[n]\right] \\
&= \mathrm{E}\left[\left(d[n] - \hat{d}[n]\right)^2\right] \\
&= \mathrm{E}\left[(d[n] - (h_0 x[n] + h_1 x[n-1]))^2\right] \\
&= \sigma_d^2 + r_x[0]h_0^2 + r_x[0]h_1^2 + 2r_x[1]h_0 h_1 - 2r_{dx}[0]h_0 - 2r_{dx}[1]h_1 \\
&= \sigma_d^2 + \begin{bmatrix} h_0 & h_1 \end{bmatrix} \begin{bmatrix} r_x[0] & r_x[1] \\ r_x[1] & r_x[0] \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \end{bmatrix} - 2 \begin{bmatrix} h_0 & h_1 \end{bmatrix} \begin{bmatrix} r_{dx}[0] \\ r_{dx}[1] \end{bmatrix}
\end{aligned}
$$

Error surface ($N = 2$)

$$
\begin{aligned}
J &= \mathrm{E}\left[e^2[n]\right] \\
&= \mathrm{E}\left[\left(d[n] - \hat{d}[n]\right)^2\right] \\
&= \mathrm{E}\left[(d[n] - (h_0 x[n] + h_1 x[n-1]))^2\right] \\
&= \sigma_d^2 + r_x[0]h_0^2 + r_x[0]h_1^2 + 2r_x[1]h_0 h_1 - 2r_{dx}[0]h_0 - 2r_{dx}[1]h_1 \\
&= \sigma_d^2 + \begin{bmatrix} h_0 & h_1 \end{bmatrix} \begin{bmatrix} r_x[0] & r_x[1] \\ r_x[1] & r_x[0] \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \end{bmatrix} - 2 \begin{bmatrix} h_0 & h_1 \end{bmatrix} \begin{bmatrix} r_{dx}[0] \\ r_{dx}[1] \end{bmatrix}
\end{aligned}
$$

Error surface ($N = 2$)

$$
\begin{aligned}
J &= \mathrm{E}\left[e^2[n]\right] \\
&= \mathrm{E}\left[\left(d[n] - \hat{d}[n]\right)^2\right] \\
&= \mathrm{E}\left[(d[n] - (h_0 x[n] + h_1 x[n-1]))^2\right] \\
&= \sigma_d^2 + r_x[0]h_0^2 + r_x[0]h_1^2 + 2r_x[1]h_0h_1 - 2r_{dx}[0]h_0 - 2r_{dx}[1]h_1 \\
&= \sigma_d^2 + \begin{bmatrix} h_0 & h_1 \end{bmatrix} \begin{bmatrix} r_x[0] & r_x[1] \\ r_x[1] & r_x[0] \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \end{bmatrix} - 2 \begin{bmatrix} h_0 & h_1 \end{bmatrix} \begin{bmatrix} r_{dx}[0] \\ r_{dx}[1] \end{bmatrix}
\end{aligned}
$$

Error surface ($N = 2$)

$$
\begin{aligned}
J &= \mathrm{E}\left[e^2[n]\right] \\
&= \mathrm{E}\left[\left(d[n] - \hat{d}[n]\right)^2\right] \\
&= \mathrm{E}\left[(d[n] - (h_0 x[n] + h_1 x[n-1]))^2\right] \\
&= \sigma_d^2 + r_x[0]h_0^2 + r_x[0]h_1^2 + 2r_x[1]h_0 h_1 - 2r_{dx}[0]h_0 - 2r_{dx}[1]h_1 \\
&= \sigma_d^2 + \begin{bmatrix} h_0 & h_1 \end{bmatrix} \begin{bmatrix} r_x[0] & r_x[1] \\ r_x[1] & r_x[0] \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \end{bmatrix} - 2 \begin{bmatrix} h_0 & h_1 \end{bmatrix} \begin{bmatrix} r_{dx}[0] \\ r_{dx}[1] \end{bmatrix}
\end{aligned}
$$

Error surface ($N = 2$)

$$
\begin{aligned}
J &= \mathrm{E}\left[e^2[n]\right] \\
&= \mathrm{E}\left[\left(d[n] - \hat{d}[n]\right)^2\right] \\
&= \mathrm{E}\left[\left(d[n] - (h_0 x[n] + h_1 x[n-1])\right)^2\right] \\
&= \sigma_d^2 + r_x[0] h_0^2 + r_x[0] h_1^2 + 2 r_x[1] h_0 h_1 - 2 r_{dx}[0] h_0 - 2 r_{dx}[1] h_1 \\
&= \sigma_d^2 + \begin{bmatrix} h_0 & h_1 \end{bmatrix} \begin{bmatrix} r_x[0] & r_x[1] \\ r_x[1] & r_x[0] \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \end{bmatrix} - 2 \begin{bmatrix} h_0 & h_1 \end{bmatrix} \begin{bmatrix} r_{dx}[0] \\ r_{dx}[1] \end{bmatrix}
\end{aligned}
$$

Error surface ($N = 2$)

$$J = \sigma_d^2 + \mathbf{h}^T \mathbf{R} \mathbf{h} - 2\mathbf{h}^T \mathbf{g}$$

Error surface ($N = 2$)

minimum achievable MSE

$$J = \boxed{\sigma_d^2} + \mathbf{h}^T\mathbf{R}\mathbf{h} - 2\mathbf{h}^T\mathbf{g}$$

Error surface ($N = 2$)

translation term (minimum is not in origin)

$$J = \sigma_d^2 + \mathbf{h}^T \mathbf{R} \mathbf{h} - \boxed{2\mathbf{h}^T \mathbf{g}}$$

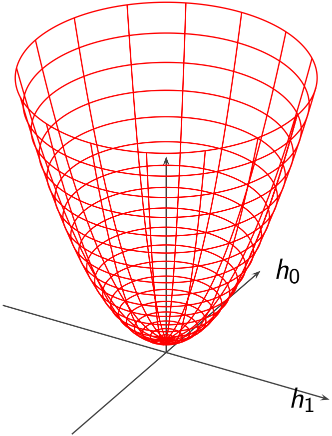
Error surface ($N = 2$)

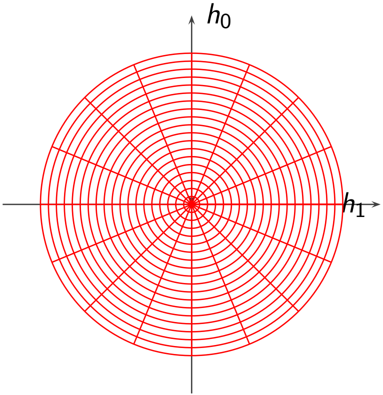
$$J = \sigma_d^2 + \mathbf{h}^T \mathbf{R} \mathbf{h}$$

error surface is an elliptic paraboloid:

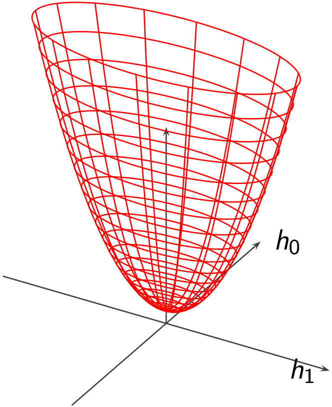
- major and minor axes are proportional to $1/\sqrt{\lambda_{0,1}}$
- signal's autocorrelation determines the shape of the error surface

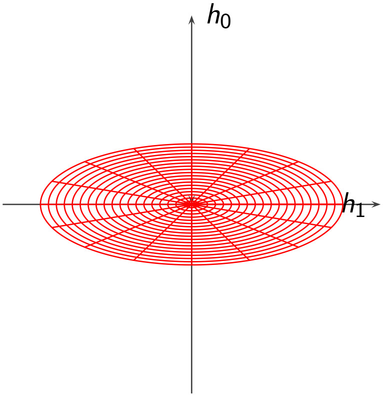
Error surface for white input



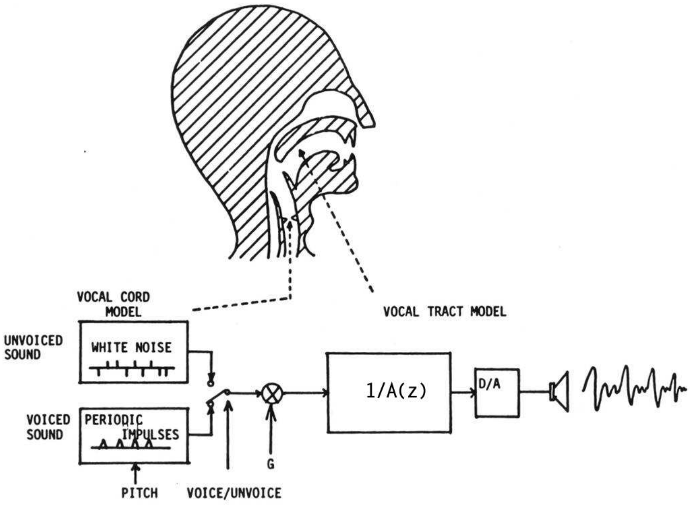


Error surface for correlated input





Example: linear prediction coding of speech

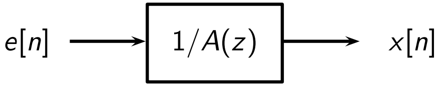


All-pole models

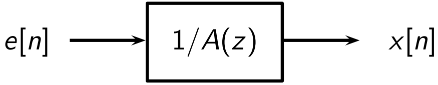
$$H(z) = \frac{1}{A(z)} = \frac{1}{1 - a_1 z^{-1} - a_2 z^{-2} - \ldots - a_N z^{-N}};$$

- poles model natural resonances of physical systems
- model is also called autoregressive (output is purely recursive)

Estimating an all-pole model



Estimating an all-pole model



- $e[n]$: unknown excitation
- $x[n]$: observable signal
- can we determine $A(z)$?

Linear Prediction

$$X(z) = E(z)/A(z)$$

$$E(z) = X(z)A(z)$$

$$e[n] = x[n] - \sum_{k=1}^{N} a_k x[n-k]$$

Linear Prediction

$$X(z) = E(z)/A(z)$$

$$E(z) = X(z)A(z)$$

$$e[n] = x[n] - \sum_{k=1}^{N} a_k x[n-k]$$

Linear Prediction

$$X(z) = E(z)/A(z)$$
$$E(z) = X(z)A(z)$$

$$e[n] = x[n] - \sum_{k=1}^{N} a_k x[n-k]$$

Remember the optimal Least Squares solution...

$$e[n] = d[n] - \sum_{k=0}^{N-1} h[k]x[n-k]$$

Linear Prediction

$$e[n] = x[n] - \sum_{k=1}^{N} a_k x[n-k]$$

- we shouldn't be able to predict excitation $e[n]$
- excitation and prediction should be orthogonal
- Least Squares solution is *the* solution

Linear Prediction

by setting $\partial \mathrm{E}\left[e^2[n]\right]/\partial a_i$ to zero...

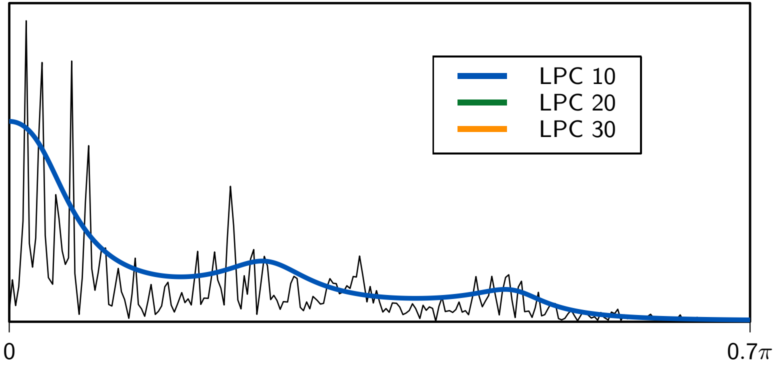
$$\mathbf{R}\hat{\mathbf{a}} = \mathbf{r}$$

$$\begin{bmatrix} r_x[0] & r_x[1] & \dots & r_x[N-1] \\ r_x[1] & r_x[0] & \dots & r_x[N-2] \\ \vdots & \vdots & \ddots & \vdots \\ r_x[N-1] & r_x[N-2] & \dots & r_x[0] \end{bmatrix} \begin{bmatrix} \hat{a}_1 \\ \hat{a}_2 \\ \vdots \\ \hat{a}_N \end{bmatrix} = \begin{bmatrix} r_x[1] \\ r_x[2] \\ \vdots \\ r_x[N] \end{bmatrix}.$$

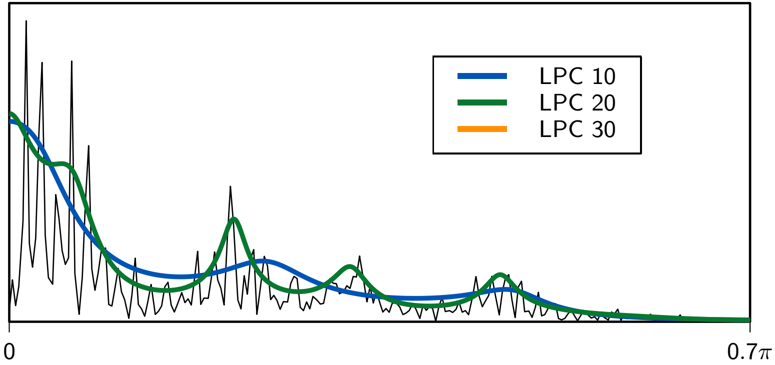
LPC speech coding

- segment speech in 20ms chunks (approx. stationary)
- find the coefficients for an all-pole model
- inverse filter and find the residual
- classify the residual excitation as voiced/unvoiced

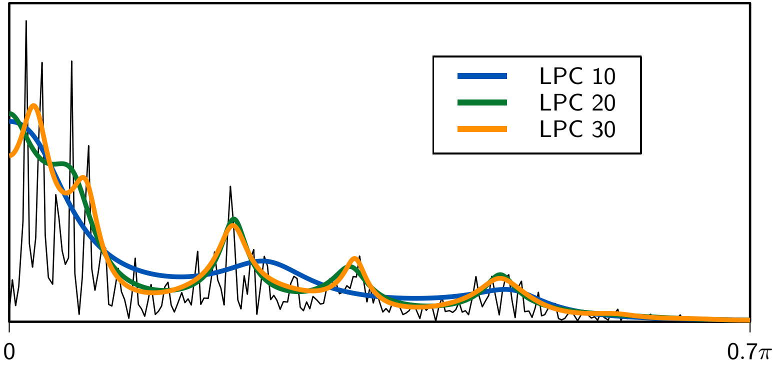
LPC order selection



LPC order selection



LPC order selection



LPC speech coding

- normally $N = 20$
- average bitrate 4Kbit/sec (raw data: 48Kbit/sec)
- many improvements exist: CELP & Co

original

LPC-coded

Finding the minimum squared error

Two cases:

- for WSS signals, one-shot solution: Optimal Least Squares
- for “almost” WSS signals, iterative solutions: stochastic gradient descent or LMS

Iterative minimization

Steepest descent:

start with a guess $\mathbf{x}_0$ and then, iteratively,

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \alpha_n \nabla f(\mathbf{x}_n)$$

$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f(\mathbf{x})}{\partial x_0} & \frac{\partial f(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial f(\mathbf{x})}{\partial x_{N-1}} \end{bmatrix}^T$$

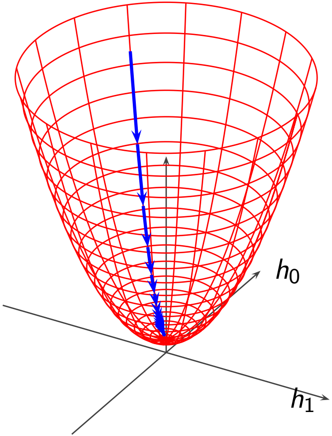
α_n: learning factor

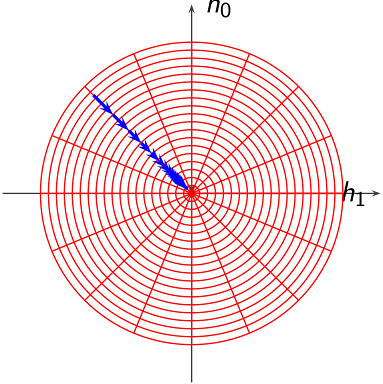
Iterative minimization

- for a quadratic error surface, minimum is always global
- gradient is easy to compute:

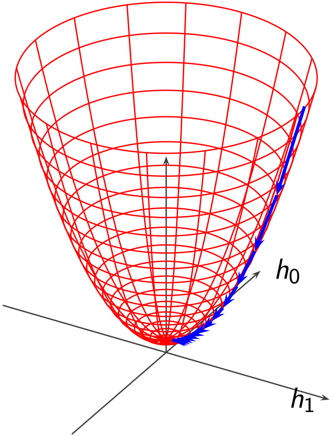
$$
\begin{aligned}
\nabla J(\mathbf{h}) &= \left[\frac{\partial \mathrm{E}\left[e^2[n]\right]}{\partial h[0]} \quad \frac{\partial \mathrm{E}\left[e^2[n]\right]}{\partial h[1]} \quad \ldots \quad \frac{\partial \mathrm{E}\left[e^2[n]\right]}{\partial h[N-1]} \right]^T \\
&= 2(\mathbf{R}\mathbf{h} - \mathbf{g})
\end{aligned}
$$

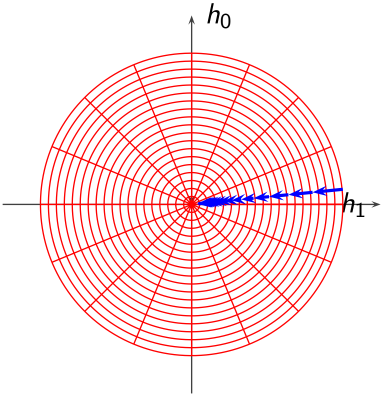
Steepest descent for white input



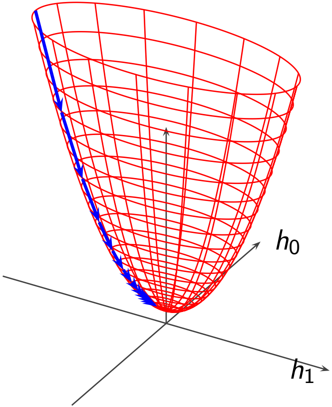


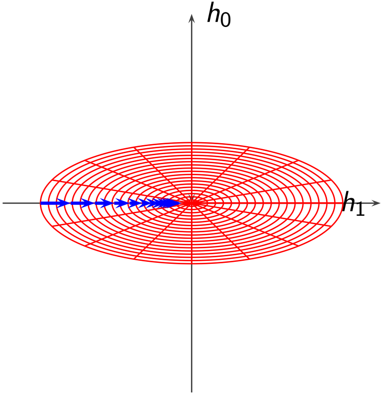
Steepest descent for white input



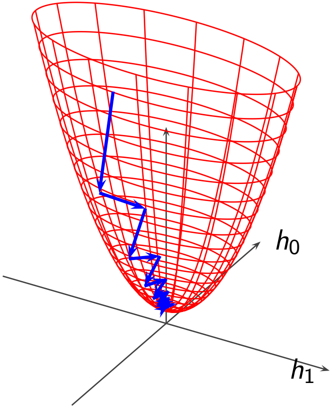


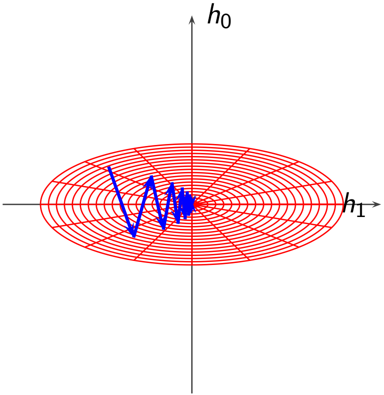
Error surface for correlated input: good guess



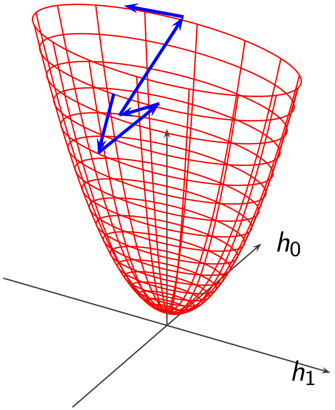


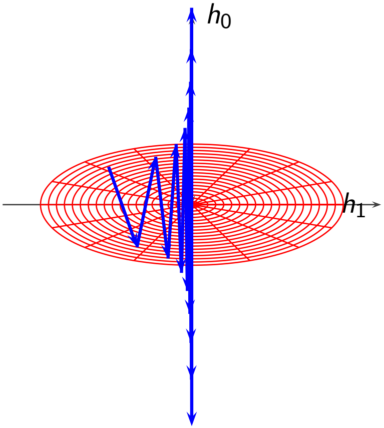
Error surface for correlated input: less good guess





Error surface for correlated input: learning factor too large!





Iterative minimization

- for WSS signals, one-shot and iterative are the same
- for time-varying signals, we need to follow the changes: iterative solution
- computation of time-varying correlations is costly
- *stochastic* gradient descent:

$$\mathrm{E}\left[e^2[n]\right] \leftarrow e^2[n]$$

Iterative minimization

- for WSS signals, one-shot and iterative are the same
- for time-varying signals, we need to follow the changes: iterative solution
- computation of time-varying correlations is costly
- *stochastic* gradient descent:

$$\mathrm{E}\left[e^2[n]\right] \leftarrow e^2[n]$$

Iterative minimization

- for WSS signals, one-shot and iterative are the same
- for time-varying signals, we need to follow the changes: iterative solution
- computation of time-varying correlations is costly
- *stochastic* gradient descent:

$$\mathrm{E}\left[e^2[n]\right] \leftarrow e^2[n]$$

Iterative minimization

- for WSS signals, one-shot and iterative are the same
- for time-varying signals, we need to follow the changes: iterative solution
- computation of time-varying correlations is costly
- *stochastic* gradient descent:

$$\mathrm{E}\left[e^2[n]\right] \leftarrow e^2[n]$$

Stochastic gradient descent

$$\nabla J = \left[\frac{\partial \mathrm{E}\left[e^2[n]\right]}{\partial h[0]} \quad \frac{\partial \mathrm{E}\left[e^2[n]\right]}{\partial h[1]} \quad \cdots \quad \frac{\partial \mathrm{E}\left[e^2[n]\right]}{\partial h[N-1]} \right]^T$$

Stochastic gradient descent

$$\nabla J_n = \left[\frac{\partial e^2[n]}{\partial h[0]} \quad \frac{\partial e^2[n]}{\partial h[1]} \quad \cdots \quad \frac{\partial e^2[n]}{\partial h[N-1]} \right]^T$$

Stochastic gradient descent

$$e[n] = d[n] - \sum_{k=0}^{N-1} h[k]x[n-k]$$

$$\frac{\partial e^2[n]}{\partial h[i]} = -2e[n]\,x[n-i].$$

$$\nabla J_n = -2e[n]\,\mathbf{x_n}$$

$$\mathbf{x}_n = \begin{bmatrix} x[n] & x[n-1] & x[n-2] & \dots & x[n-N+1] \end{bmatrix}^T.$$

Stochastic gradient descent

$$e[n] = d[n] - \sum_{k=0}^{N-1} h[k]x[n-k]$$

$$\frac{\partial e^2[n]}{\partial h[i]} = -2e[n]\,x[n-i].$$

$$\nabla J_n = -2e[n]\,\mathbf{x_n}$$

$$\mathbf{x}_n = \begin{bmatrix} x[n] & x[n-1] & x[n-2] & \dots & x[n-N+1] \end{bmatrix}^T.$$

Stochastic gradient descent

$$e[n] = d[n] - \sum_{k=0}^{N-1} h[k]x[n-k]$$

$$\frac{\partial e^2[n]}{\partial h[i]} = -2e[n]\,x[n-i].$$

$$\nabla J_n = -2e[n]\,\mathbf{x_n}$$

$$\mathbf{x}_n = \begin{bmatrix} x[n] & x[n-1] & x[n-2] & \ldots & x[n-N+1] \end{bmatrix}^T.$$

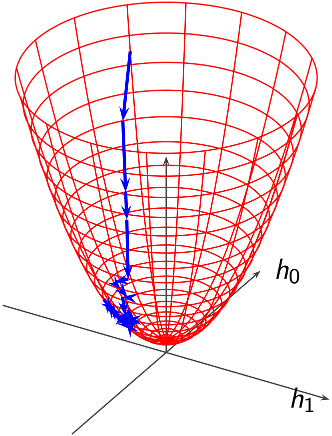
The LMS filter

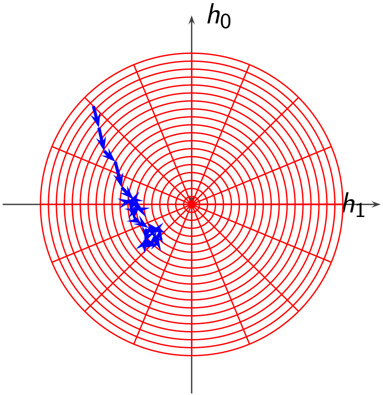
$$\mathbf{h}_0 = \begin{bmatrix} h_0[0] & h_0[1] & \dots & h_0[N-1] \end{bmatrix}^T \text{ initial guess}$$

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

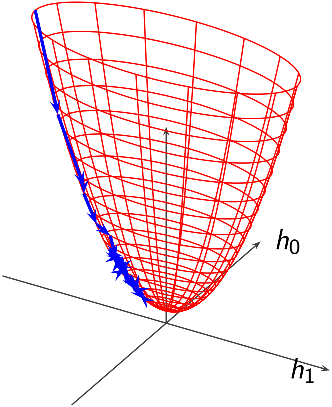
$$\mathbf{h}_{n+1} = \mathbf{h}_n + \alpha_n \, e[n] \, \mathbf{x}_n$$

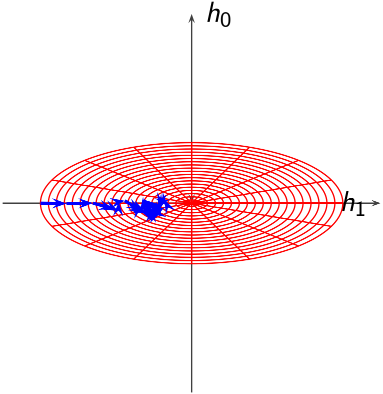
LMS for white input



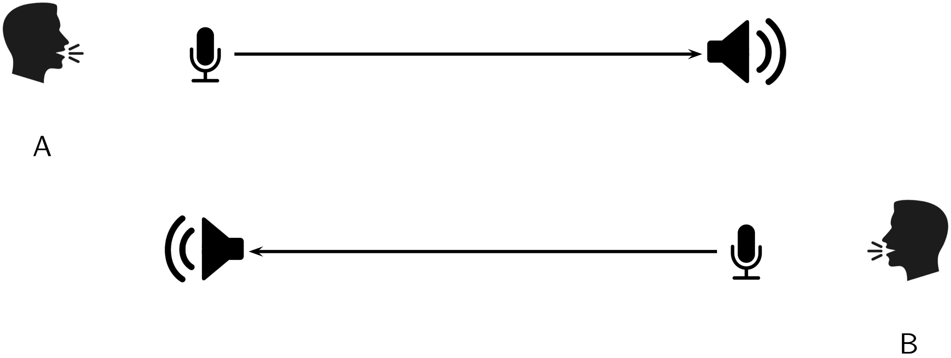


LMS for correlated input

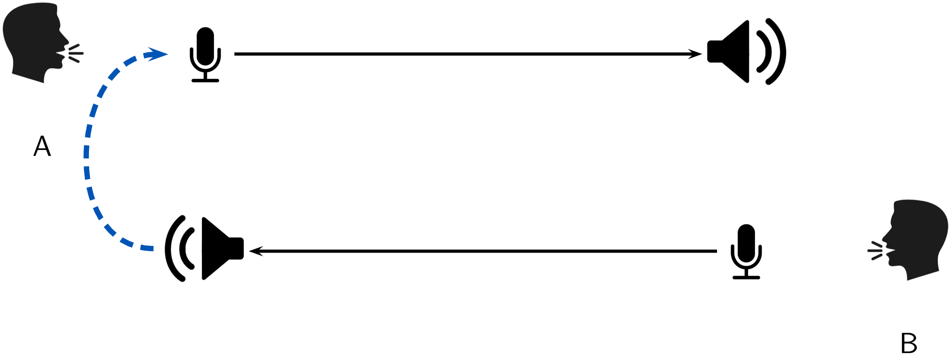




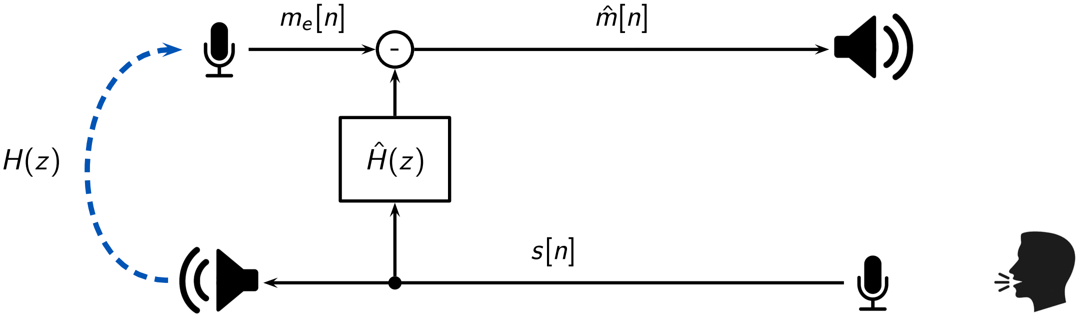
Example: adaptive echo cancellation



Example: adaptive echo cancellation



Example: adaptive echo cancellation



The echo-corrupted signal

Signal captured by the microphone:

$$m_e[n] = m[n] + h[n] * s[n]$$

The echo-corrupted signal

Signal captured by the microphone:

$$m_e[n] = \boxed{m[n]} + h[n] * s[n]$$

speaker A's voice

The echo-corrupted signal

Signal captured by the microphone:

$$m_e[n] = m[n] + \boxed{h[n]} * s[n]$$

echo transfer function

The echo-corrupted signal

Signal captured by the microphone:

$$m_e[n] = m[n] + h[n] * \underbrace{\boxed{s[n]}}_{\text{speaker B's voice}}$$

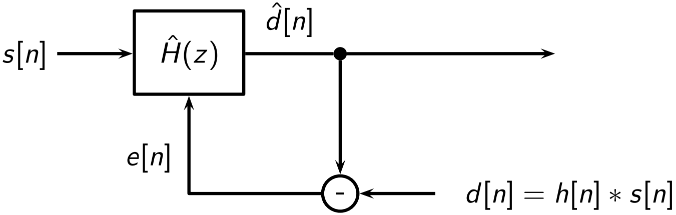
The echo-corrupted signal

Signal captured by the microphone:

$$m_e[n] = m[n] + h[n] * s[n]$$

we need to estimate $h[n]$ in order to *subtract* the unwanted echo

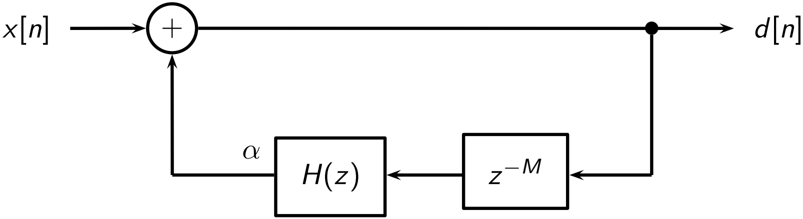
Echo cancellation as adaptive filtering



Training the filter

- "desired" signal is the echo (so we can subtract it)
- normally, only one person talks at a time: when B is speaking, $m_e[n] = h[n] * s[n]$
- people move, volume changes: $H(z)$ is time varying!
- use the LMS filter

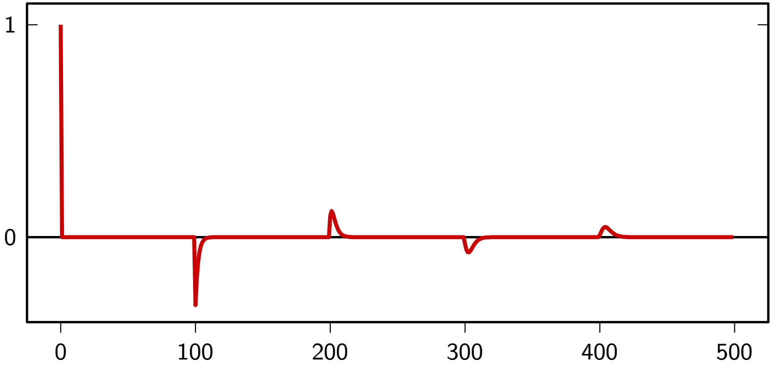
Example: simple echo model



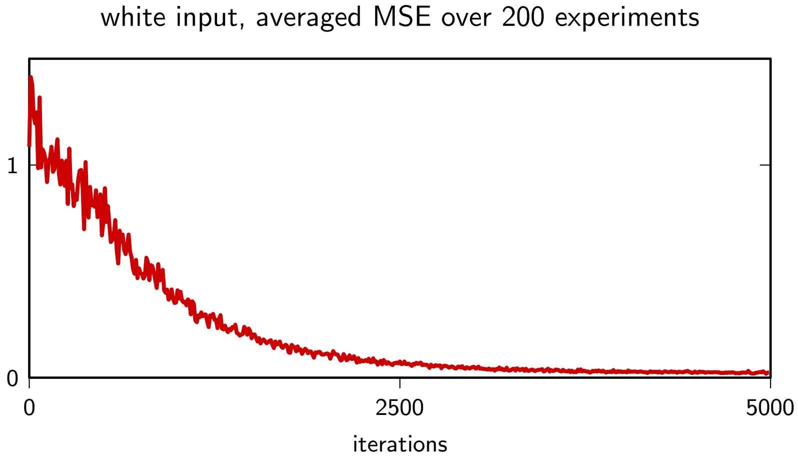
$$H(z) = (1-\lambda)/(1-\lambda z^{-1})$$

Echo impulse response

$$M = 100, \alpha = -0.8, \lambda = 0.6$$



Running the LMS adaptation



LMS can catch up with changes