Digital Signal Processing

Module 7: Stochastic Signal Processing and Quantization

Module Overview:

- Module 7.1: Stochastic signals
- Module 7.2: Quantization
- Module 7.2: A/D and D/A conversion

Digital Signal Processing

Module 7.1: Stochastic signal processing

Overview:

- A simple random signal
- Power spectral density
- Filtering a stochastic signal
- Noise

Overview:

- A simple random signal
- Power spectral density
- Filtering a stochastic signal
- Noise

Overview:

- A simple random signal
- Power spectral density
- Filtering a stochastic signal
- Noise

Overview:

- A simple random signal
- Power spectral density
- Filtering a stochastic signal
- Noise

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!
- will not develop stochastic signal processing rigorously but give enough intuition to deal with things such as "noise"

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!
- will not develop stochastic signal processing rigorously but give enough intuition to deal with things such as "noise"

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!
- will not develop stochastic signal processing rigorously but give enough intuition to deal with things such as "noise"

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!
- will not develop stochastic signal processing rigorously but give enough intuition to deal with things such as "noise"

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!
- will not develop stochastic signal processing rigorously but give enough intuition to deal with things such as "noise"

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!
- will not develop stochastic signal processing rigorously but give enough intuition to deal with things such as "noise"

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

For each new sample, toss a fair coin:

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

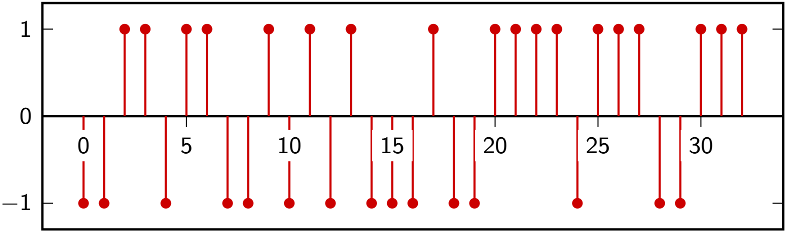
- each sample is independent from all others
- each sample value has a 50% probability

A simple discrete-time random signal generator

- every time we turn on the generator we obtain a different *realization* of the signal
- we know the “mechanism” behind each instance
- but how can we analyze a random signal?

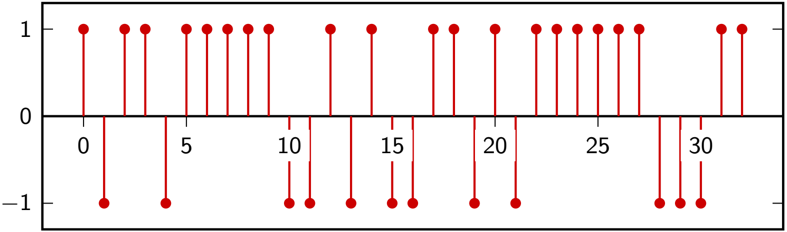
A simple discrete-time random signal generator

- 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?



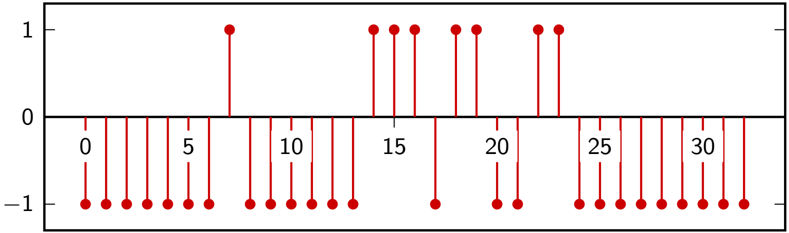
A simple discrete-time random signal generator

- 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?



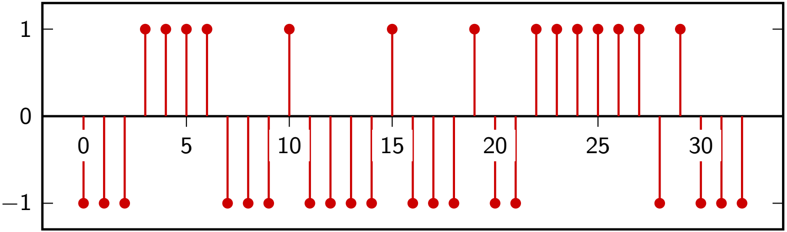
A simple discrete-time random signal generator

- 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?



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?

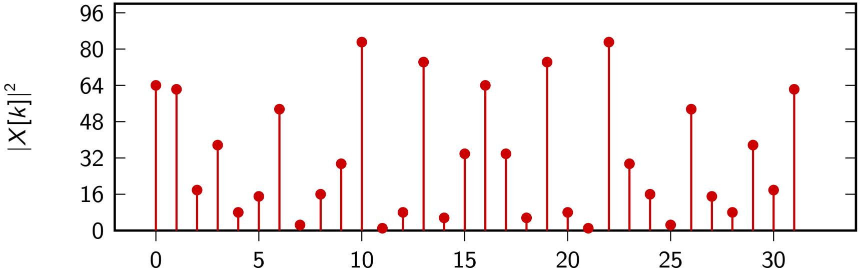


Spectral properties?

- let's try with the DFT of a finite set of random samples
- every time it's different; maybe with more data?
- no clear pattern... we need a new strategy

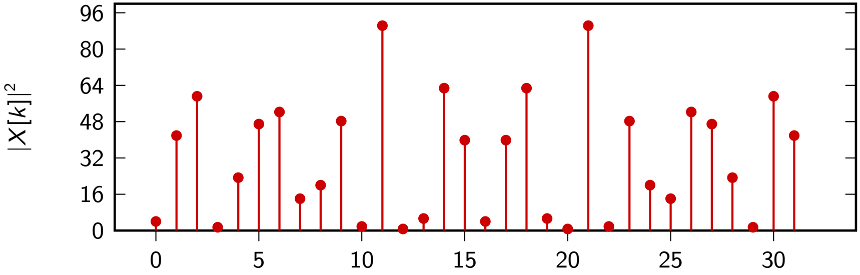
Spectral properties?

- let's try with the DFT of a finite set of random samples
- every time it's different; maybe with more data?
- no clear pattern... we need a new strategy



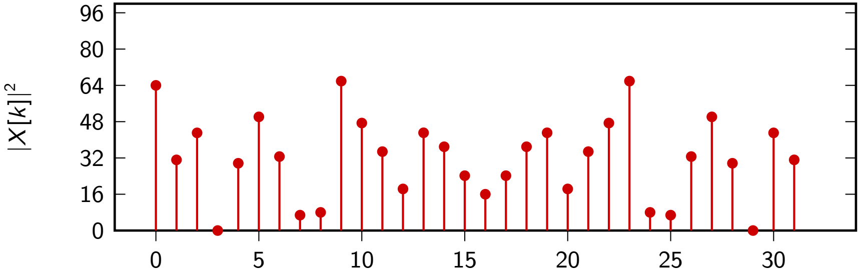
Spectral properties?

- let's try with the DFT of a finite set of random samples
- every time it's different; maybe with more data?
- no clear pattern... we need a new strategy



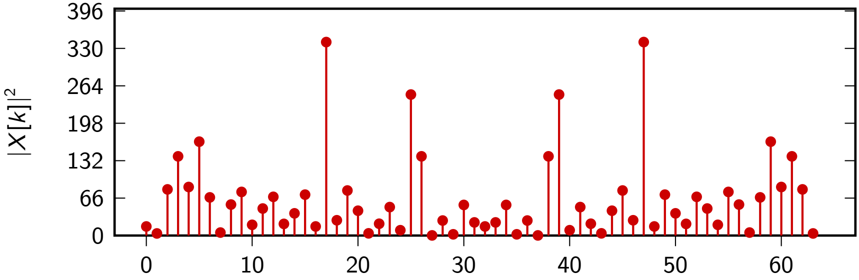
Spectral properties?

- let's try with the DFT of a finite set of random samples
- every time it's different; maybe with more data?
- no clear pattern... we need a new strategy



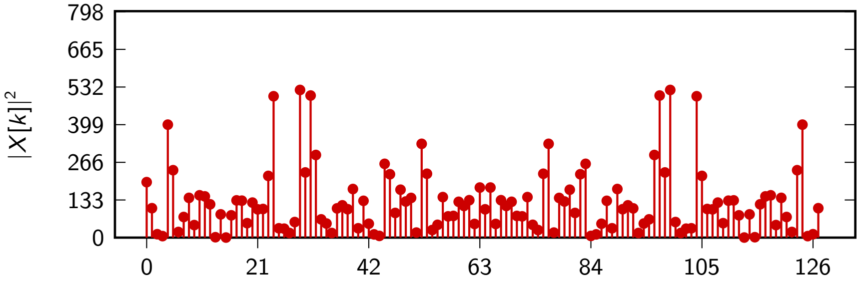
Spectral properties?

- let's try with the DFT of a finite set of random samples
- every time it's different; maybe with more data?
- no clear pattern... we need a new strategy



Spectral properties?

- let's try with the DFT of a finite set of random samples
- every time it's different; maybe with more data?
- no clear pattern... we need a new strategy



Averaging

- when faced with random data an intuitive response is to take "averages"
- in probability theory the average is across realizations and it's called *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

- when faced with random data an intuitive response is to take "averages"
- in probability theory the average is across realizations and it's called *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

- when faced with random data an intuitive response is to take "averages"
- in probability theory the average is across realizations and it's called *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

- when faced with random data an intuitive response is to take "averages"
- in probability theory the average is across realizations and it's called *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

- ... as a consequence, averaging the DFT will not work
- $\text{E}[X[k]] = 0$
- however the signal "moves", so its energy or power must be nonzero

Averaging the DFT

- ... as a consequence, averaging the DFT will not work
- $\mathrm{E}[X[k]] = 0$
- however the signal "moves", so its energy or power must be nonzero

Averaging the DFT

- … as a consequence, averaging the DFT will not work
- $\mathrm{E}[X[k]] = 0$
- however the signal “moves”, so its energy or power must be nonzero

Energy and power

- the coin-toss signal has infinite energy (see Module 2.1):

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

- however it has finite power over any interval:

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

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

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

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

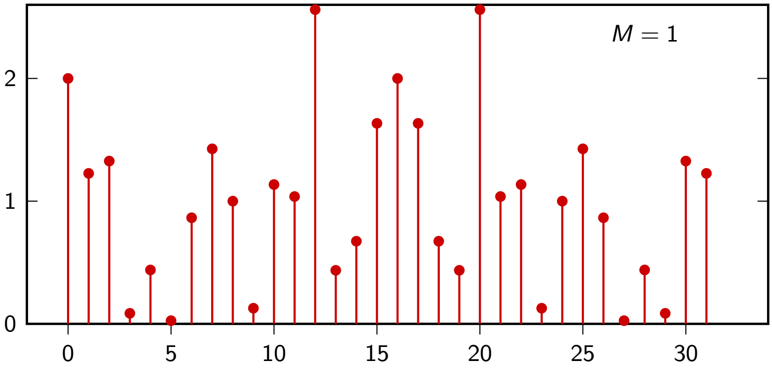
- pick an interval length N
- pick a number of iterations M
- run the signal generator M times and obtain M N-point realizations
- compute the DFT of each realization
- average their square magnitude divided by N

Averaging

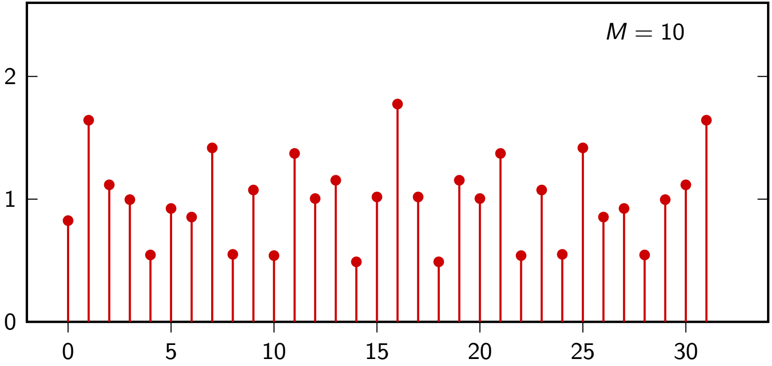
let's try to average the DFT's square magnitude, normalized:

- 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

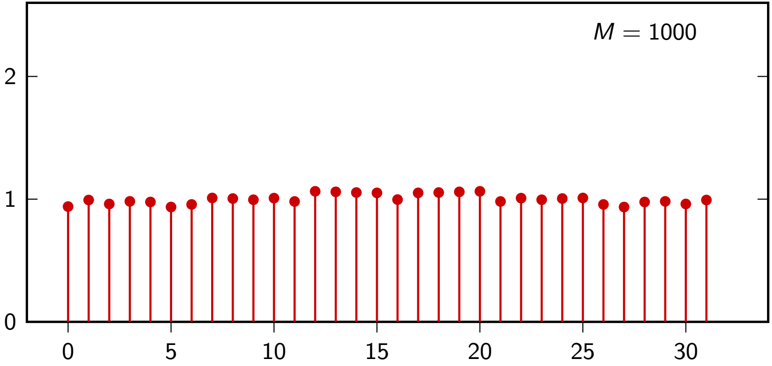
Averaged DFT square magnitude



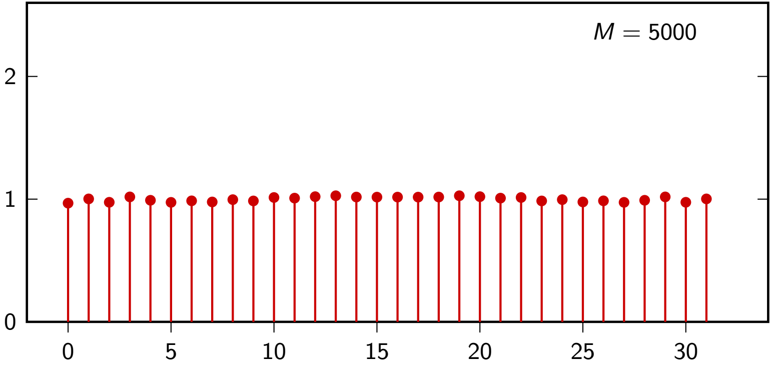
Averaged DFT square magnitude



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

- $P[k] = 1$ 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?

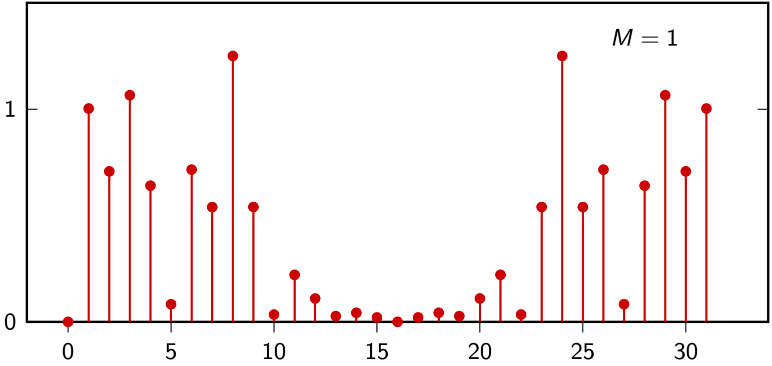
Filtering a random process

- 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?

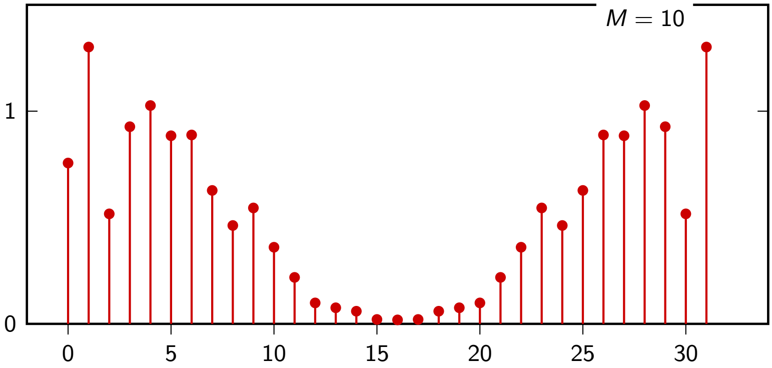
Filtering a random process

- 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?

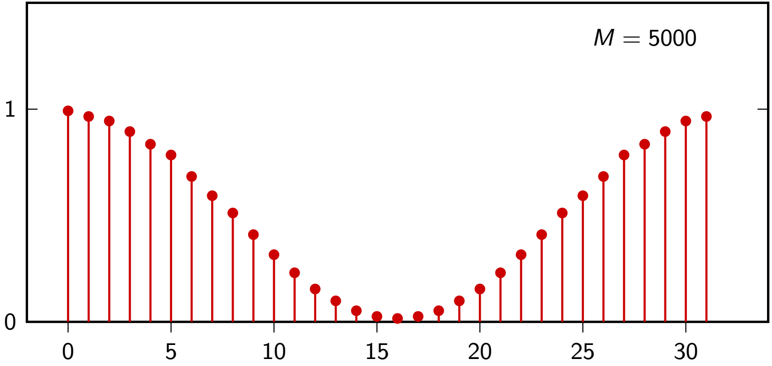
Averaged DFT magnitude of filtered process



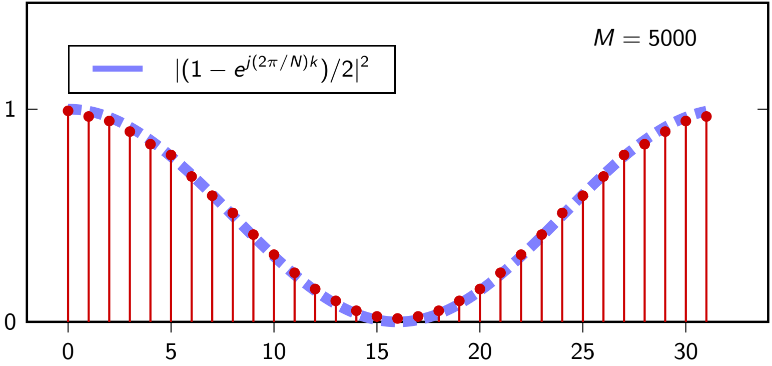
Averaged DFT magnitude of filtered process



Averaged DFT magnitude of filtered process



Averaged DFT magnitude of filtered process



- 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?

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?

Stochastic signal processing

- a stochastic process is characterized by its power spectral density (PSD)
- it can be shown (see the textbook) that the PSD is

$$P_x(e^{j\omega}) = \text{DTFT}\{r_x[n]\}$$

  where $r_x[n] = \text{E}[x[k]\,x[n+k]]$ is the autocorrelation of the process.

- for a filtered stochastic process $y[n] = \mathcal{H}\{x[n]\}$, it is:

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

Stochastic signal processing

- a stochastic process is characterized by its power spectral density (PSD)
- it can be shown (see the textbook) that the PSD is

$$P_x(e^{j\omega}) = \text{DTFT}\{r_x[n]\}$$

where $r_x[n] = \mathrm{E}[x[k]\,x[n+k]]$ is the autocorrelation of the process.

- for a filtered stochastic process $y[n] = \mathcal{H}\{x[n]\}$, it is:

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

- a stochastic process is characterized by its power spectral density (PSD)
- it can be shown (see the textbook) that the PSD is

$$P_x(e^{j\omega}) = \mathrm{DTFT}\left\{r_x[n]\right\}$$

  where $r_x[n] = \mathrm{E}\left[x[k]\,x[n+k]\right]$ is the autocorrelation of the process.
- for a filtered stochastic process $y[n] = \mathcal{H}\{x[n]\}$, it is:

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

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

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

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

White noise

- "white" indicates uncorrelated samples
- $r_w[n] = \sigma^2 \delta[n]$
- $P_w(e^{j\omega}) = \sigma^2$

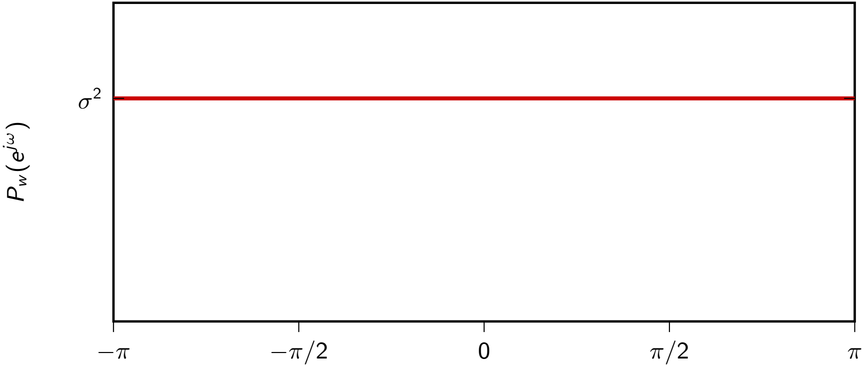
White noise

- “white” indicates uncorrelated samples
- $r_w[n] = \sigma^2 \delta[n]$
- $P_w(e^{j\omega}) = \sigma^2$

White noise

- "white" indicates uncorrelated samples
- $r_w[n] = \sigma^2 \delta[n]$
- $P_w(e^{j\omega}) = \sigma^2$

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

END OF MODULE 7.1

Digital Signal Processing

Module 7.2: Quantization

Overview:

- Quantization
- Uniform quantization and error analysis
- Clipping, saturation, companding

Overview:

- Quantization
- Uniform quantization and error analysis
- Clipping, saturation, companding

Overview:

- Quantization
- Uniform quantization and error analysis
- Clipping, saturation, companding

Quantization

- digital devices can only deal with integers (b bits per sample)
- we need to map the range of a signal onto a finite set of values
- irreversible loss of information $\rightarrow$ quantization noise

- digital devices can only deal with integers (b bits per sample)
- we need to map the range of a signal onto a finite set of values
- irreversible loss of information $\rightarrow$ quantization noise

Quantization

- digital devices can only deal with integers (b bits per sample)
- we need to map the range of a signal onto a finite set of values
- irreversible loss of information $\rightarrow$ quantization noise

Quantization schemes

$$x[n] \longrightarrow \boxed{\mathcal{Q}\{\cdot\}} \longrightarrow \hat{x}[n]$$

Several factors at play:

- storage budget (bits per sample)
- storage scheme (fixed point, floating point)
- properties of the input
  - range
  - probability distribution

Quantization schemes

$$x[n] \longrightarrow \boxed{\mathcal{Q}\{\cdot\}} \longrightarrow \hat{x}[n]$$

Several factors at play:

- storage budget (bits per sample)
- storage scheme (fixed point, floating point)
- properties of the input
  - range
  - probability distribution

Quantization schemes

$$x[n] \longrightarrow \boxed{\mathcal{Q}\{\cdot\}} \longrightarrow \hat{x}[n]$$

Several factors at play:

- storage budget (bits per sample)
- storage scheme (fixed point, floating point)
- properties of the input
  - range
  - probability distribution

Quantization schemes

$$x[n] \longrightarrow \boxed{\mathcal{Q}\{\cdot\}} \longrightarrow \hat{x}[n]$$

Several factors at play:

- storage budget (bits per sample)
- storage scheme (fixed point, floating point)
- properties of the input
  - range
  - probability distribution

Quantization schemes

$$x[n] \longrightarrow \boxed{\mathcal{Q}\{\cdot\}} \longrightarrow \hat{x}[n]$$

Several factors at play:

- storage budget (bits per sample)
- storage scheme (fixed point, floating point)
- properties of the input
  - range
  - probability distribution

Scalar quantization

$$x[n] \longrightarrow \boxed{\mathcal{Q}\{\cdot\}} \longrightarrow \hat{x}[n]$$

The simplest quantizer:

- each sample is encoded individually (hence *scalar*)
- each sample is quantized independently (memoryless quantization)
- each sample is encoded using R bits

Scalar quantization

$$x[n] \longrightarrow \boxed{\mathcal{Q}\{\cdot\}} \longrightarrow \hat{x}[n]$$

The simplest quantizer:

- each sample is encoded individually (hence *scalar*)
- each sample is quantized independently (memoryless quantization)
- each sample is encoded using R bits

Scalar quantization

$$x[n] \longrightarrow \boxed{\mathcal{Q}\{\cdot\}} \longrightarrow \hat{x}[n]$$

R bps

The simplest quantizer:

- each sample is encoded individually (hence *scalar*)
- each sample is quantized independently (memoryless quantization)
- each sample is encoded using R bits

Scalar quantization

Assume input signal bounded: $A \leq x[n] \leq B$ for all n:

- each sample quantized over 2^R possible values $\Rightarrow$ 2^R intervals.
- each interval associated to a quantization value

Scalar quantization

Assume input signal bounded: $A \leq x[n] \leq B$ for all n:

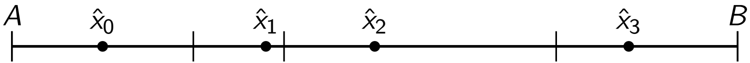
- each sample quantized over 2^R possible values $\Rightarrow$ 2^R intervals.

- each interval associated to a quantization value

Scalar quantization

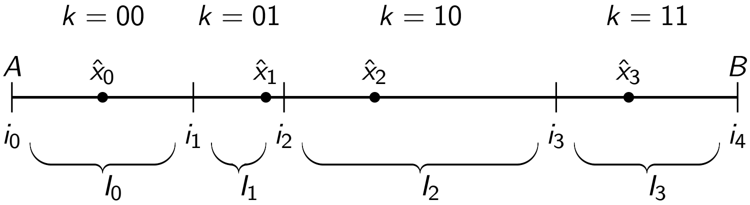
Assume input signal bounded: $A \leq x[n] \leq B$ for all n:

- each sample quantized over 2^R possible values $\Rightarrow 2^R$ intervals.
- each interval associated to a quantization value



Scalar quantization

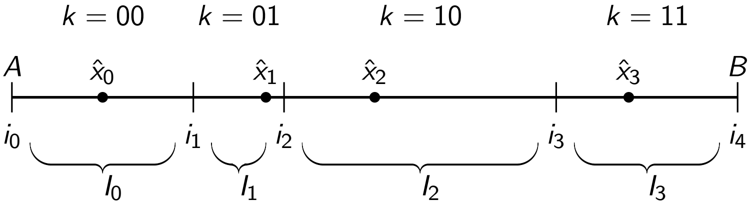
Example for $R = 2$:



- what are the optimal interval boundaries i_k?
- what are the optimal quantization values $\hat{x}_k$?

Scalar quantization

Example for $R = 2$:



- what are the optimal interval boundaries i_k?
- what are the optimal quantization values $\hat{x}_k$?

Quantization Error

$$e[n] = \mathcal{Q}\{x[n]\} - x[n] = \hat{x}[n] - x[n]$$

- model $x[n]$ as a stochastic process
- model error as a white noise sequence:
  - error samples are uncorrelated
  - all error samples have the same distribution
- we need statistics of the input to study the error

Quantization Error

$$e[n] = \mathcal{Q}\{x[n]\} - x[n] = \hat{x}[n] - x[n]$$

- model $x[n]$ as a stochastic process
- model error as a white noise sequence:
  - error samples are uncorrelated
  - all error samples have the same distribution
- we need statistics of the input to study the error

Quantization Error

$$e[n] = \mathcal{Q}\{x[n]\} - x[n] = \hat{x}[n] - x[n]$$

- model $x[n]$ as a stochastic process
- model error as a white noise sequence:
  - error samples are uncorrelated
  - all error samples have the same distribution
- we need statistics of the input to study the error

Quantization Error

$$e[n] = \mathcal{Q}\{x[n]\} - x[n] = \hat{x}[n] - x[n]$$

- model $x[n]$ as a stochastic process
- model error as a white noise sequence:
  - error samples are uncorrelated
  - all error samples have the same distribution
- we need statistics of the input to study the error

Quantization Error

$$e[n] = \mathcal{Q}\{x[n]\} - x[n] = \hat{x}[n] - x[n]$$

- model $x[n]$ as a stochastic process
- model error as a white noise sequence:
  - error samples are uncorrelated
  - all error samples have the same distribution
- we need statistics of the input to study the error

Uniform quantization

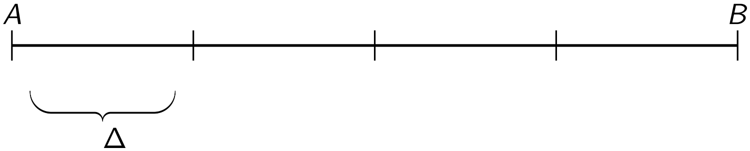
- simple but very general case
- range is split into 2^R *equal* intervals of width $\Delta = (B - A)2^{-R}$

Uniform quantization

- simple but very general case
- range is split into 2^R *equal* intervals of width $\Delta = (B - A)2^{-R}$

Uniform quantization

- simple but very general case
- range is split into 2^R *equal* intervals of width $\Delta = (B-A)2^{-R}$



Uniform quantization

Mean Square Error is the second moment of the error signal:

$$\sigma_e^2 = \mathrm{E}\left[|\mathcal{Q}\{x[n]\} - x[n]|^2\right]$$

$$= \int_A^B f_x(\tau)(\mathcal{Q}\{\tau\} - \tau)^2 \, d\tau$$

$$= \sum_{k=0}^{2^R-1} \int_{I_k} f_x(\tau)(\hat{x}_k - \tau)^2 \, d\tau$$

error depends on the probability distribution of the input

Uniform quantization

Mean Square Error is the second moment of the error signal:

$$
\begin{aligned}
\sigma_e^2 &= \mathrm{E}\left[|\mathcal{Q}\{x[n]\} - x[n]|^2\right] \\
&= \int_A^B f_x(\tau)(\mathcal{Q}\{\tau\} - \tau)^2 \, d\tau \\
&= \sum_{k=0}^{2^R-1} \int_{I_k} f_x(\tau)(\hat{x}_k - \tau)^2 \, d\tau
\end{aligned}
$$

error depends on the probability distribution of the input

Uniform quantization

Mean Square Error is the second moment of the error signal:

$$\begin{aligned}
\sigma_e^2 &= \mathrm{E}\left[|\mathcal{Q}\{x[n]\} - x[n]|^2 \right] \\
&= \int_A^B f_x(\tau)(\mathcal{Q}\{\tau\} - \tau)^2 \, d\tau \\
&= \sum_{k=0}^{2^R-1} \int_{I_k} f_x(\tau)(\hat{x}_k - \tau)^2 \, d\tau
\end{aligned}$$

error depends on the probability distribution of the input

Mean Square Error is the second moment of the error signal:

$$\begin{aligned}
\sigma_e^2 &= \mathrm{E}\left[|\mathcal{Q}\{x[n]\} - x[n]|^2\right] \\
&= \int_A^B f_x(\tau)(\mathcal{Q}\{\tau\} - \tau)^2 \, d\tau \\
&= \sum_{k=0}^{2^R-1} \int_{I_k} f_x(\tau)(\hat{x}_k - \tau)^2 \, d\tau
\end{aligned}$$

error depends on the probability distribution of the input

Uniform quantization of uniform input

Uniform-input hypothesis:

$$f_x(\tau) = \frac{1}{B - A}$$

$$\sigma_e^2 = \sum_{k=0}^{2^R-1} \int_{I_k} \frac{(\hat{x}_k - \tau)^2}{B - A} \, d\tau$$

Let's find the optimal quantization point by minimizing the error

$$\frac{\partial \sigma_e^2}{\partial \hat{x}_m} = \frac{\partial}{\partial \hat{x}_m} \sum_{k=0}^{2^R-1} \int_{I_k} \frac{(\hat{x}_k - \tau)^2}{B-A} \, d\tau$$

$$= \int_{I_m} \frac{2(\hat{x}_m - \tau)}{B-A} \, d\tau$$

$$= \left. \frac{(\hat{x}_m - \tau)^2}{B-A} \right|_{A+m\Delta}^{A+m\Delta+\Delta}$$

Uniform quantization of uniform input

Let's find the optimal quantization point by minimizing the error

$$\begin{aligned}\frac{\partial \sigma_e^2}{\partial \hat{x}_m} &= \frac{\partial}{\partial \hat{x}_m} \sum_{k=0}^{2^R-1} \int_{I_k} \frac{(\hat{x}_k - \tau)^2}{B-A}\, d\tau \\ &= \int_{I_m} \frac{2(\hat{x}_m - \tau)}{B-A}\, d\tau \\ &= \left. \frac{(\hat{x}_m - \tau)^2}{B-A} \right|_{A+m\Delta}^{A+m\Delta+\Delta}\end{aligned}$$

Let's find the optimal quantization point by minimizing the error

$$
\begin{aligned}
\frac{\partial \sigma_e^2}{\partial \hat{x}_m} &= \frac{\partial}{\partial \hat{x}_m} \sum_{k=0}^{2^R-1} \int_{I_k} \frac{(\hat{x}_k - \tau)^2}{B-A}\, d\tau \\
&= \int_{I_m} \frac{2(\hat{x}_m - \tau)}{B-A}\, d\tau \\
&= \left. \frac{(\hat{x}_m - \tau)^2}{B-A} \right|_{A+m\Delta}^{A+m\Delta+\Delta}
\end{aligned}
$$

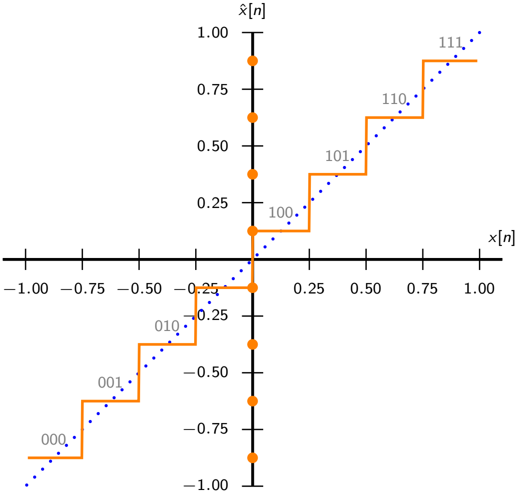
Uniform quantization of uniform input

Minimizing the error:

$$\frac{\partial \sigma_e^2}{\partial \hat{x}_m} = 0 \quad \text{for } \hat{x}_m = A + m\Delta + \frac{\Delta}{2}$$

optimal quantization point is the interval's midpoint, for all intervals

Uniform 3-Bit quantization function



Quantizer's mean square error:

$$\sigma_e^2 = \sum_{k=0}^{2^R-1} \int_{A+k\Delta}^{A+k\Delta+\Delta} \frac{(A + k\Delta + \Delta/2 - \tau)^2}{B - A}\, d\tau$$

$$= 2^R \int_0^{\Delta} \frac{(\Delta/2 - \tau)^2}{B - A}\, d\tau$$

$$= \frac{\Delta^2}{12}$$

Uniform quantization of uniform input

Quantizer's mean square error:

$$\sigma_e^2 = \sum_{k=0}^{2^R-1} \int_{A+k\Delta}^{A+k\Delta+\Delta} \frac{(A + k\Delta + \Delta/2 - \tau)^2}{B - A} \, d\tau$$

$$= 2^R \int_0^{\Delta} \frac{(\Delta/2 - \tau)^2}{B - A} \, d\tau$$

$$= \frac{\Delta^2}{12}$$

Quantizer's mean square error:

$$\sigma_e^2 = \sum_{k=0}^{2^R-1} \int_{A+k\Delta}^{A+k\Delta+\Delta} \frac{(A + k\Delta + \Delta/2 - \tau)^2}{B - A} \, d\tau$$

$$= 2^R \int_0^{\Delta} \frac{(\Delta/2 - \tau)^2}{B - A} \, d\tau$$

$$= \frac{\Delta^2}{12}$$

Error analysis

- second moment of the error (error's power)

$$\sigma_e^2 = \Delta^2/12, \qquad \Delta = (B - A)/2^R$$

- second moment of the signal (signal's power)

$$\sigma_x^2 = (B - A)^2/12$$

- signal to noise ratio

$$\mathrm{SNR} = 2^{2R}$$

- in dB

$$\mathrm{SNR}_{\mathrm{dB}} = 10 \log_{10} 2^{2R} \approx 6R \text{ dB}$$

Error analysis

- second moment of the error (error's power)

$$\sigma_e^2 = \Delta^2/12, \qquad \Delta = (B - A)/2^R$$

- second moment of the signal (signal's power)

$$\sigma_x^2 = (B - A)^2/12$$

- signal to noise ratio

$$\text{SNR} = 2^{2R}$$

- in dB

$$\text{SNR}_{\text{dB}} = 10 \log_{10} 2^{2R} \approx 6R \text{ dB}$$

Error analysis

- second moment of the error (error's power)

$$\sigma_e^2 = \Delta^2/12, \qquad \Delta = (B - A)/2^R$$

- second moment of the signal (signal's power)

$$\sigma_x^2 = (B - A)^2/12$$

- signal to noise ratio

$$\mathrm{SNR} = 2^{2R}$$

- in dB

$$\mathrm{SNR}_{\mathrm{dB}} = 10 \log_{10} 2^{2R} \approx 6R \text{ dB}$$

Error analysis

- second moment of the error (error's power)

$$\sigma_e^2 = \Delta^2/12, \qquad \Delta = (B - A)/2^R$$

- second moment of the signal (signal's power)

$$\sigma_x^2 = (B - A)^2/12$$

- signal to noise ratio

$$\mathrm{SNR} = 2^{2R}$$

- in dB

$$\mathrm{SNR}_{\mathrm{dB}} = 10 \log_{10} 2^{2R} \approx 6R \text{ dB}$$

The "6dB/bit" rule of thumb

- a compact disk has 16 bits/sample:

$$\max \text{SNR} = 96\text{dB}$$

- a DVD has 24 bits/sample:

$$\max \text{SNR} = 144\text{dB}$$

The "6dB/bit" rule of thumb

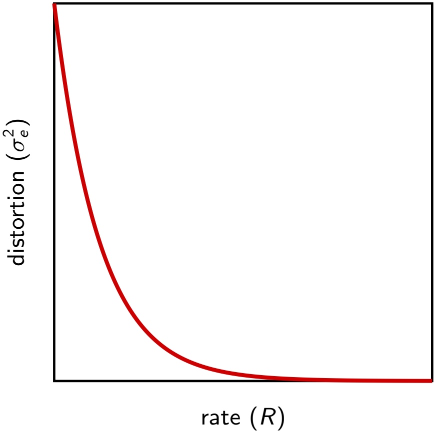
- a compact disk has 16 bits/sample:

$$\max \text{SNR} = 96\text{dB}$$

- a DVD has 24 bits/sample:

$$\max \text{SNR} = 144\text{dB}$$

Rate/Distortion Curve



Other quantization errors

If input is not bounded to $[A, B]$:

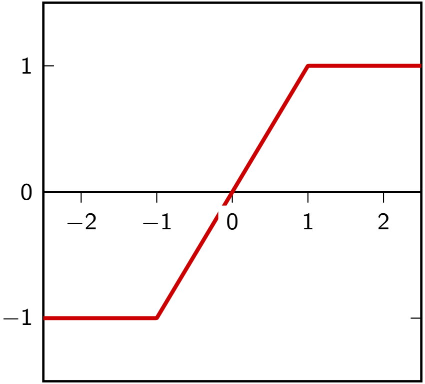
- clip samples to $[A, B]$: linear distortion (can be put to good use in guitar effects!)
- smoothly saturate input: this simulates the saturation curves of analog electronics

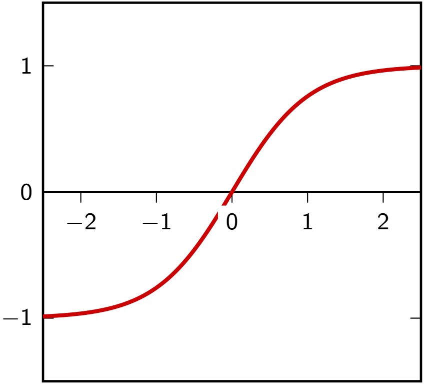
Other quantization errors

If input is not bounded to $[A, B]$:

- clip samples to $[A, B]$: linear distortion (can be put to good use in guitar effects!)
- smoothly saturate input: this simulates the saturation curves of analog electronics

Clipping vs saturation





Other quantization errors

If input is not uniform:

- use uniform quantizer and accept increased error.
  For instance, if input is Gaussian:

$$\sigma_e^2 = \frac{\sqrt{3}\pi}{2}\,\sigma^2\,\Delta^2$$

- design optimal quantizer for input distribution, if known (Lloyd-Max algorithm)
- use "companders"

Other quantization errors

If input is not uniform:

- use uniform quantizer and accept increased error.
  For instance, if input is Gaussian:

$$\sigma_e^2 = \frac{\sqrt{3}\pi}{2}\,\sigma^2\,\Delta^2$$

- design optimal quantizer for input distribution, if known (Lloyd-Max algorithm)

- use "companders"

Other quantization errors

If input is not uniform:

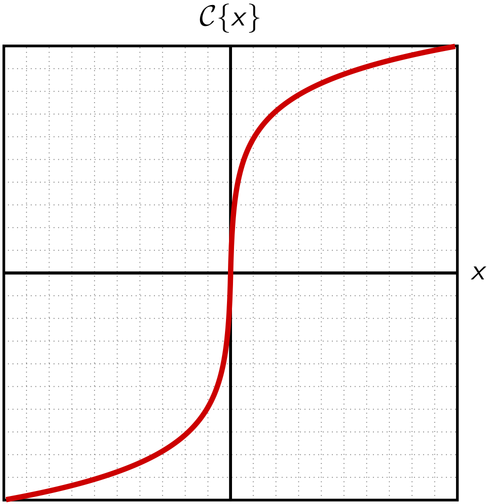
- use uniform quantizer and accept increased error.
  For instance, if input is Gaussian:

$$\sigma_e^2 = \frac{\sqrt{3}\pi}{2}\,\sigma^2\,\Delta^2$$

- design optimal quantizer for input distribution, if known (Lloyd-Max algorithm)
- use "companders"

μ-law compander

$$\mathcal{C}\{x[n]\} = \operatorname{sgn}(x[n]) \frac{\ln(1 + \mu |x[n]|)}{\ln(1 + \mu)}$$



END OF MODULE 7.2

Digital Signal Processing

Module 7.3: A/D and D/A Conversion

Overview:

- Analog-to-digital (A/D) conversion
- Digital-to-analog (D/A) conversion

Overview:

- Analog-to-digital (A/D) conversion
- Digital-to-analog (D/A) conversion

From analog to digital

- sampling discretizes time
- quantization discretized amplitude
- how is it done in practice?

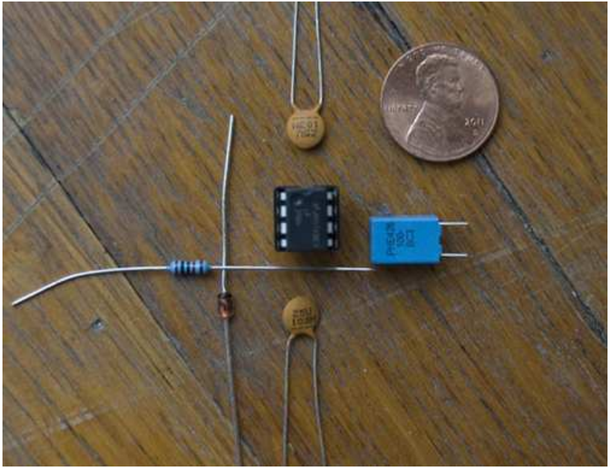
From analog to digital

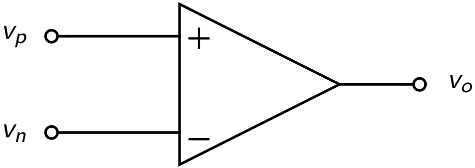
- sampling discretizes time
- quantization discretized amplitude
- how is it done in practice?

From analog to digital

- sampling discretizes time
- quantization discretized amplitude
- how is it done in practice?

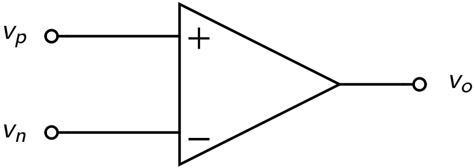
From analog to digital





$$v_o = G(v_p - v_n)$$

A tiny bit of electronics: the op-amp



$$v_o = G(v_p - v_n)$$

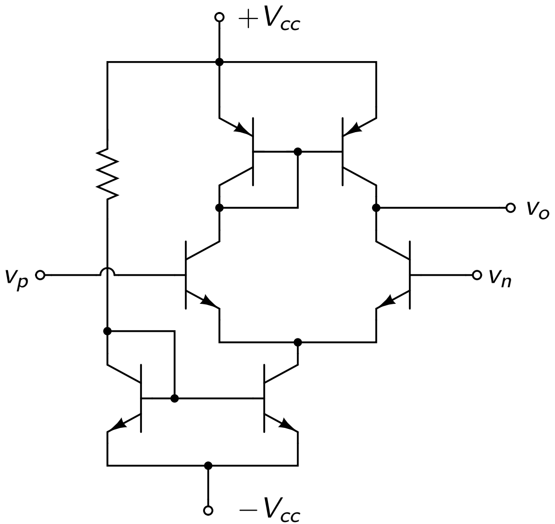
The two key properties

- infinite input gain ($G \approx \infty$)
- zero input current

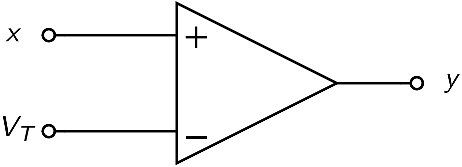
The two key properties

- infinite input gain ($G \approx \infty$)
- zero input current

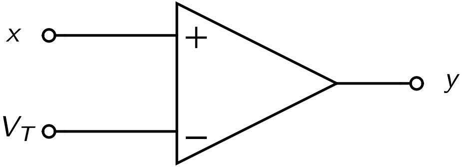
Inside the box



The op-amp in open loop: comparator

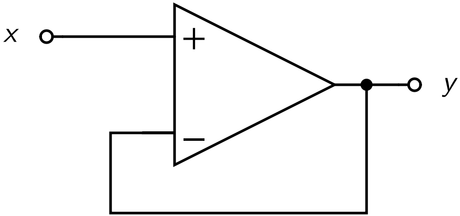


$$y = \begin{cases} +V_{cc} & \text{if } x > V_T \\ -V_{cc} & \text{if } x < V_T \end{cases}$$



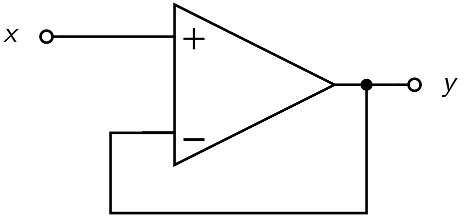
$$y = \begin{cases} +V_{cc} & \text{if } x > V_T \\ -V_{cc} & \text{if } x < V_T \end{cases}$$

The op-amp in closed loop: buffer



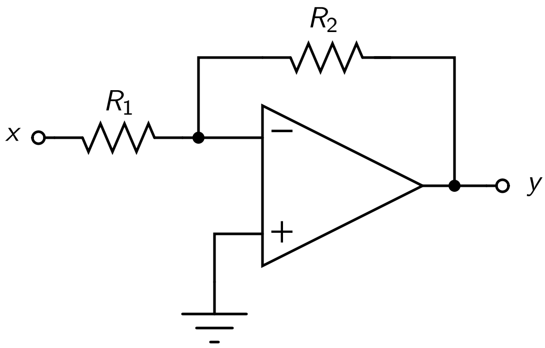
$$y = x$$

The op-amp in closed loop: buffer



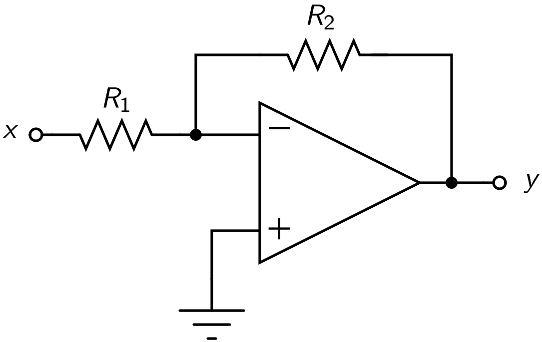
$$y = x$$

The op-amp in closed loop: inverting amplifier



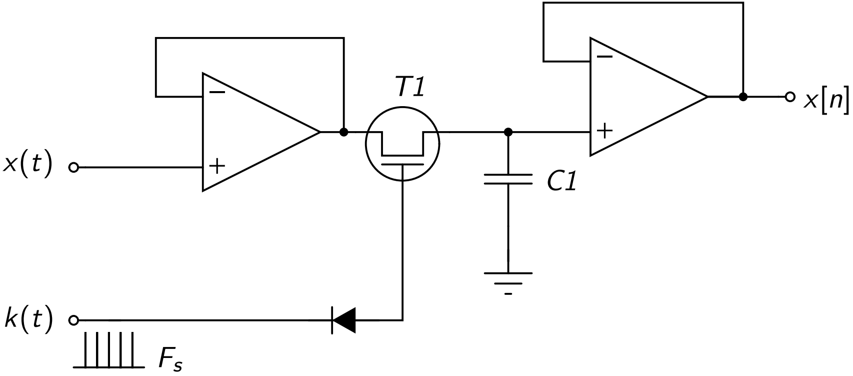
$y = -(R_2/R_1)x$

The op-amp in closed loop: inverting amplifier

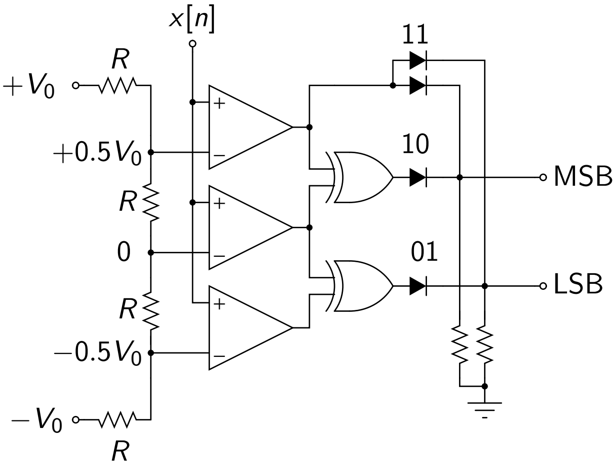


$$y = -(R_2/R_1)x$$

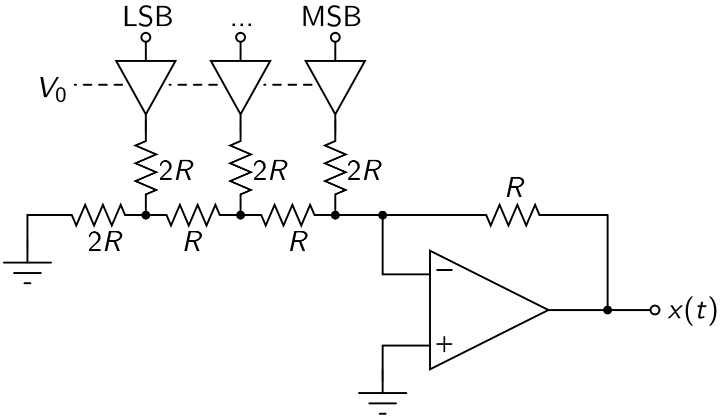
A/D Converter: Sample & Hold



A/D Converter: 2-Bit Quantizer



D/A Converter



END OF MODULE 7.3

END OF MODULE 7