Encoding media signals

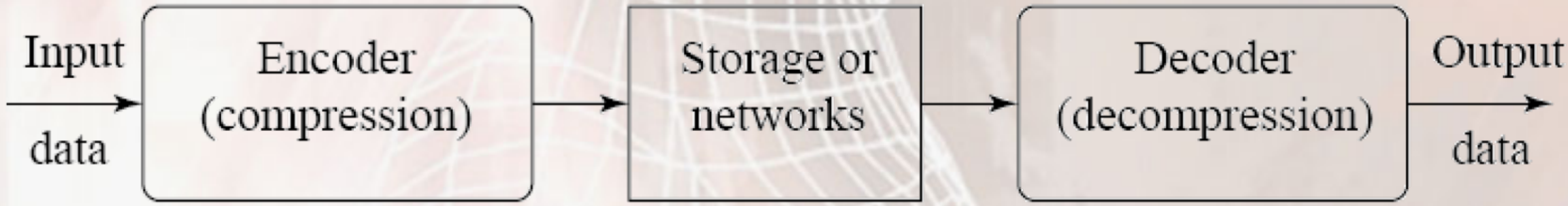
Prof.dr.sc. Davor Petrinović

Encoding media signals

- Most natural signals, as well as various media signals, show the property of statistical dependence of samples

Encoding media signals

- In general, ... **coding** is defined as the representation of information by a string of bits.
- **Data compression** is :
  - coding that aims to reduce the number of bits that represent certain information.



Encoding media signals

- The term coding is often used as a synonym for compression procedures, which will be the case in our materials as well.
- Compression is also called the process of removing redundancy, i.e., **redundancy removal**.
- The task of compression is:
  - transform the media format of nominal accuracy into the “equivalent” form of a digital representation in which the redundancy of information is reduced or removed.

Encoding media signals

- Coding in a broader sense can be divided into three subcategories:
  - ***source coding***,
  - ***entropy coding*** and
  - ***channel coding***.
- The first two are used in the context of compression, while ...
  - channel coding actually **increases** the number of bits of the message in order to protect the information during transmission.

Encoding media signals

- In the MS course, we will deal primarily with source coding and entropy coding.
- We also divide coding into two types:
  - ***Lossless coding*** *and*
  - ***Lossy coding***.
- Perfect reconstruction of the signal in the format of nominal accuracy is possible only in the case of lossless coding!

Encoding media signals

- What is the difference between source coding and entropy coding?
  - source coding is based on the properties of the signal to which it applies and exploits the redundancies related to the structure of a particular signal, while ...
  - entropy coding "attacks" a digital message without knowing its true source while utilizing the known or estimated statistical properties of that digital set of data samples.

Information and entropy

prof.dr.sc. Davor Petrinović
Material preparation:
Ivan Dokmanić, dipl.ing.

Information

- Information (amount of information) can be defined as the **level of surprise due to an event**, or as the improbability of that event.
- Example - guessing a woman's name :
  - If someone tells us that the first letter of the name is M (common first letter in a name), it doesn't mean much to us, but if they tell us that the name starts with Ž, the amount of information is much larger, and we can guess more easily which name it is.

Information measure

- Suppose a memoryless source of information that can generate discrete symbols from the set : $S = \{s_1, s_2, s_3, ..., s_n\}$
- If the probability of generating the i-th symbol is p_i, the amount of information due to the observation of the i-th symbol is : $\log_2 \frac{1}{p_i}$
- Taking the logarithm of base 2 gives the result in bits - (minimum) number of bits needed to represent the symbol s_i

Entropy

- The entropy of a source of information is a **measure of the amount of information** that that source generates.
- It corresponds to the average (expected) amount of information per generated symbol :

$$H(S)=\sum_{i=1}^{n} p_i \log \frac{1}{p_i}$$

$$H(S)=-\sum_{i=1}^{n} p_i \log p_i$$

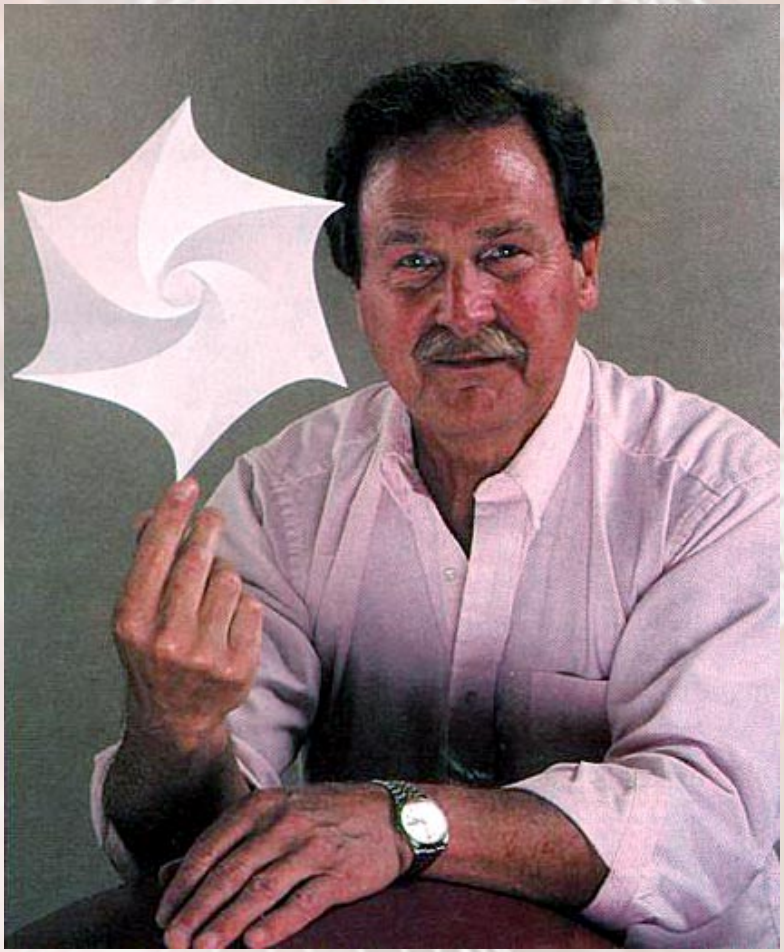
Entropy

- The entropy according to this definition corresponds to the smallest possible average length of the encoded message in bits.
- In doing so, we assume that the source is **memoryless**, i.e., that the probability of the observation of the current symbol is in no way related to the occurrence of any other symbol previously considered. Each output symbol is generated independently of previous symbols.

Variable length codes

- If the probabilities of occurrence of individual symbols are different, it is a good idea to assign **shorter code words to the more probable ones** (which occur more often) and thus reduce the expected length of the coded message.
- As codewords are of different lengths, **neither should be the beginning of another**, in order to achieve the possibility of **unique decoding**.
- Codes that have this property are called **prefix codes**.

Huffman codes



- David A. Huffman (1925-1999)

Huffman codes

- Satisfy the prefix condition!
- Algorithm for generating Huffman code:

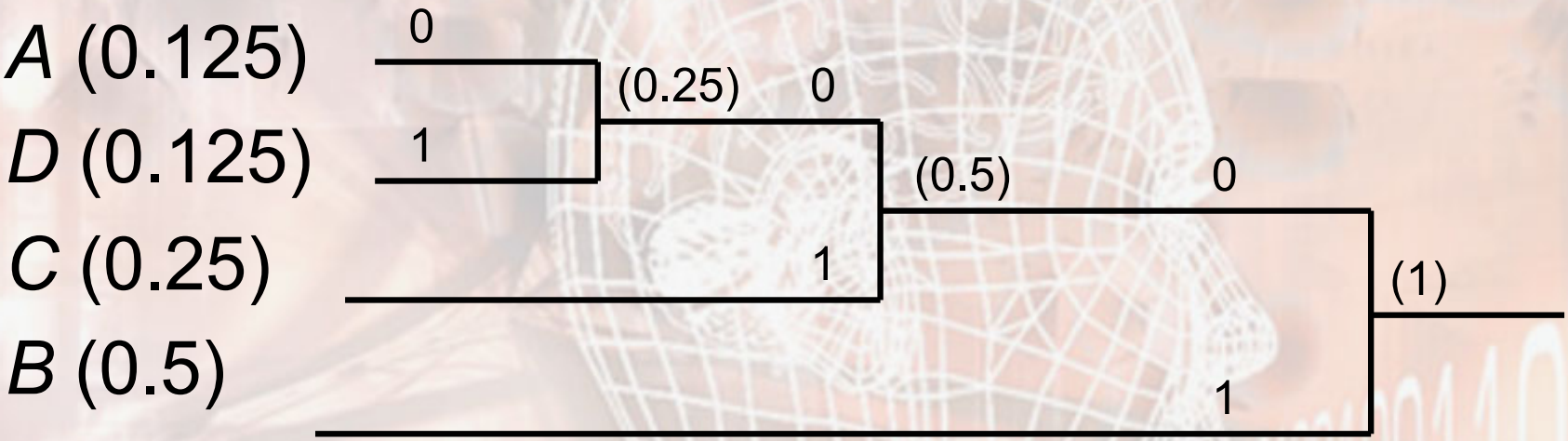
1. Sort symbols by probability,
2. Choose two symbols of least probability, make a Huffman sub-tree whose children are these symbols,
3. Assign the parent a probability that is the sum of the probabilities of the children and put it in the list of symbols so that it is still sorted, and remove the children from the list,
4. Repeat from 2 until you use all the symbols,
5. Assign codewords to each leaf of this tree (they correspond to the paths from the root of the tree to each leaf).

Huffman codes - example

- Take a source whose alphabet is $S = \{A, B, C, D\}$
- Let us construct a Huffman code for this source if the probabilities of the individual symbols are as in the table:

A	*B*	*C*	*D*
0.125	0.5	0.25	0.125

Huffman codes - example



We read code words from the root of the tree to the leaves …

A	*B*	*C*	*D*
000	1	01	001

Huffman codes - example

- Average codeword length is found as:

  0.125*3 + 0.125*3 + 0.25*2 + 0.5*1 = 1.75 bit
- This length **corresponds exactly** to the entropy of the source!
- The coded message **BADBBCBC** is: **1000001110110 1** and is 14 bits long. If we used equal codeword length for all symbols (2 bits), we would need a total of 16 bits for the same message.
- If the probability of each symbol was not an integer power of 0.5 (that is, if the amount of information per symbol was not an integer), the average length of the codeword would be at most 1 bit greater than the entropy!

Huffman codes - problem

- For natural processes, the amount of information per symbol does not really have to be an integer, and this method necessarily assigns an integer number of bits to each symbol.
- An extreme case is when the probability of a given symbol is very high (> 0.5), e.g., $p_i = 0.9$
- The amount of information of this symbol is 0.152 bits, but its codeword will be 1 bit long and thus the average code length will be significantly greater than the source entropy.

Arithmetic coding

- How to achieve fractional code length?
- Idea: encode the whole message at once – i.e., encode the whole message with one "long" number.

- Jorma J. Rissanen
- devised a procedure while working for IBM ...

Arithmetic coding

- Let the source be $S = \{A, B, C, D, E, F, \$\}$, with probabilities as in the table.
- Each symbol is assigned a subinterval of [0,1), the width of which corresponds exactly to the probability of the symbol :
  - frequent symbols - wide interval,
  - rare symbols - narrow interval.
- The order of the intervals is arbitrary.

A	B	C	D	E	F	$
0.2	**0.1**	**0.2**	**0.05**	**0.3**	**0.05**	**0.1**
[0, 0.2)	**[0.2,0.3)**	**[0.3,0.5)**	**[0.5,0.55)**	**[0.55,0.85)**	**[0.85,0.9)**	**[0.9,1)**

Arithmetic coding

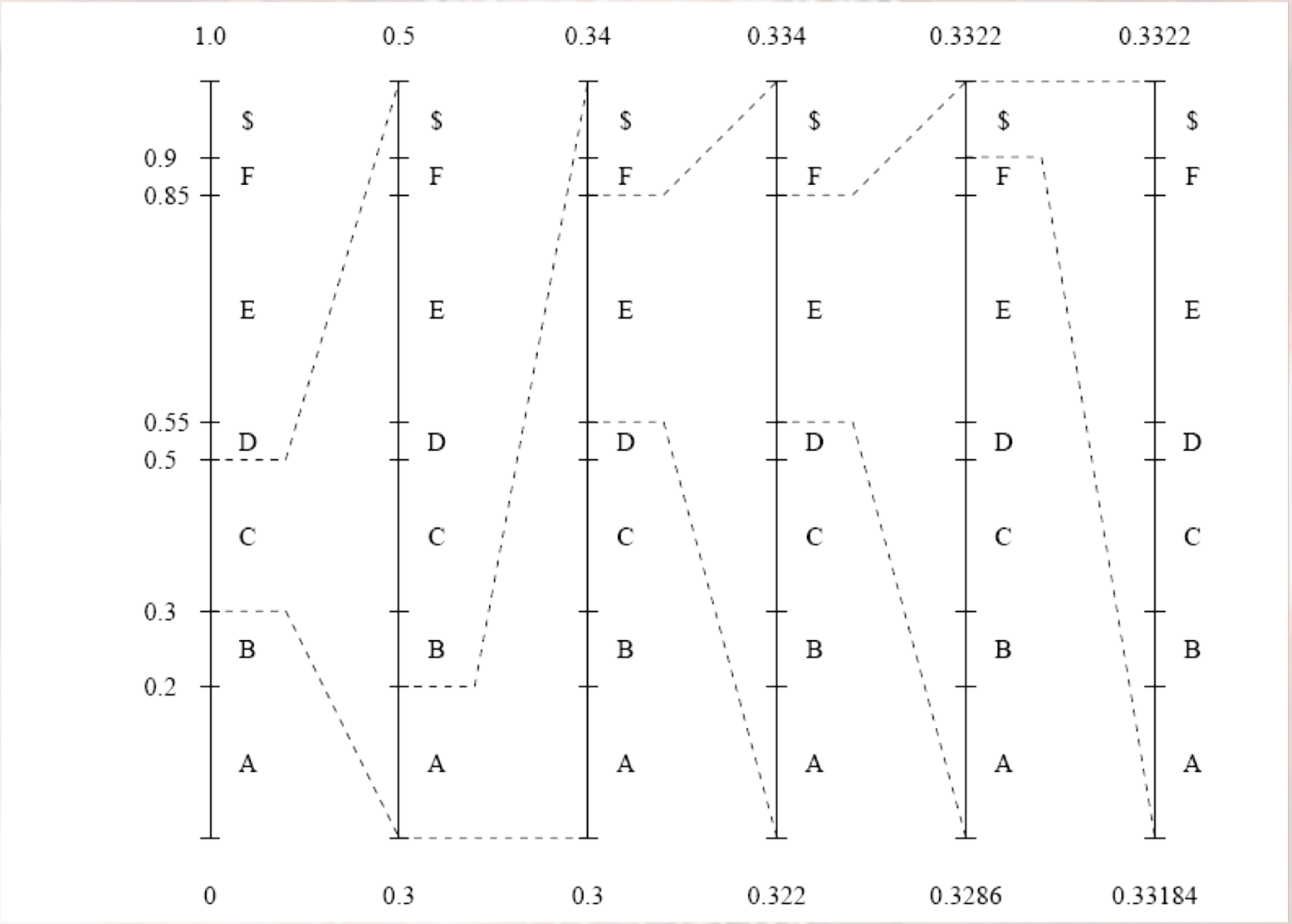
- When we encode the first symbol, we concentrate only on the subinterval corresponding to that symbol, and divide it just as we divided the initial interval [0,1), so that the following symbols correspond to the subintervals within it, and the widths of the subintervals are again proportional to the symbol probabilities.
- When we encode the next symbol, we do the same thing with the subinterval corresponding to that symbol (and located within the interval corresponding to the previous symbol).
- We repeat the procedure recursively until the end of the message, and the coded message is represented with any number from the final interval.

Arithmetic coding - example

- Let the message we encode be: CAEE$
- The procedure is illustrated by the following table in which for each symbol of the message the (sub)interval we observe is specified:

Symbol	lower	upper	width
	0	1	1
C	0.3	0.5	0.2
A	0.3	0.34	0.04
E	0.322	0.334	0.012
E	0.3286	0.3322	0.0036
$	0.33184	0.3322	0.00036

Arithmetic coding - example



Arithmetic coding - example

- Finally, it is necessary to choose the shortest possible code word, i.e., the number from the final interval with the least number of digits.
- If we represented the data in decimal format, not binary, that number would obviously be 0.332, but in binary format, that number has an infinite number of digits.
- Since in a binary number system we have only 2 options for each digit, it is easy to construct an algorithm that will generate a binary number with the least digits having value from some interval.

Arithmetic coding

1. Set b to zero and the counter to 1
2. As long as *b* is less than the lower limit of the interval
   1. Set the *i*-th binary fractional digit of *b* to 1.
   2. If *b* is greater than the upper limit of the interval, return the *i*-th digit to zero,
   3. Increment the counter by 1.
3. The condition for completing the loop will be explained later!

- The final fractional number *b* corresponds to the required number, and represents the shortest code of the original message.
- In our example, this is the number **0.33203125**
  - that is, in binary format **.01010101**

Arithmetic coding – why it works?

- The width of the final interval corresponds exactly to the probability product of all transmitted symbols.
- If we consider symbols to be independent, then that product is just the probability of the whole message, p_m.
- In general, if we want to represent numbers from very narrow intervals within [0, 1), we will have to spend a lot of digits, because narrow intervals correspond to small probabilities p_m.

Arithmetic coding – why it works?

- In any interval of width p_m within [0,1) there is a real number that can be represented by a binary fraction whose number of binary digits is the smallest integer greater than $\log_2(1/p_m)$.
- This quantity $\log_2(1/p_m)$ is exactly the amount of information in the whole message, which means that we encoded the message with the smallest possible number of bits, i.e., with a maximum of one extra bit for the whole message!
- With Huffman, we encode each symbol with up to a bit of excess, due to integer code length requirement !

Arithmetic coding – why it works?

- For our example, the probability of a CAEE$ message is:
  - $p_m = p_C p_A p_E p_E p_\$ = 0.2 \cdot 0.2 \cdot 0.3 \cdot 0.3 \cdot 0.1 =$
  - $p_m = 0.00036$
- The minimum number of bits for encoding the whole message would be equal to the entropy:
  - $-\log_2(p_m) = 11.44$ bit
- In a real system we use 12 bits and send a shortest encoded message :
  - **.010101010000**
- Thus, the number of passes of the fraction search loop is 12.

Arithmetic coding

- The decoding process is very simple and comes down to reading the figure with the interval sub-division backwards:
  - We look within which interval the current number (code word) is located;
  - The current decoded symbol corresponds to that interval.
  - The next step is to subtract the lower limit of the identified interval from the current value of the number and scale to a unit interval by dividing by the width of the interval (i.e., the probability) of the decoded symbol.
  - We return the thus modified number to the first step of the algorithm and repeat the procedure.

Arithmetic coding

- Illustration of decoding a message, described by a 12-bit fraction *b*= **.010101010000**

Current value	Output symbol	Lower limit	Upper limit	Interval width
0.33203125	C	0.3	0.5	0.2
0.16015625	A	0	0.2	0.2
0.80078125	E	0.55	0.85	0.3
0.8359375	E	0.55	0.85	0.3
0.953125	$	0.9	1	0.1

Arithmetic coding

- The potential problem arises due to the fact that the decoding process can go on indefinitely.
- This is solved by introducing a symbol that symbolizes the end of the message ($ in the example).
- The introduction of this symbol undermines optimality, but nevertheless this procedure is in most cases superior to Huffman's coding.
- One of the reasons for the weaker application of this algorithm is due to the fact that it is protected by many patents!

Arithmetic coding– problem

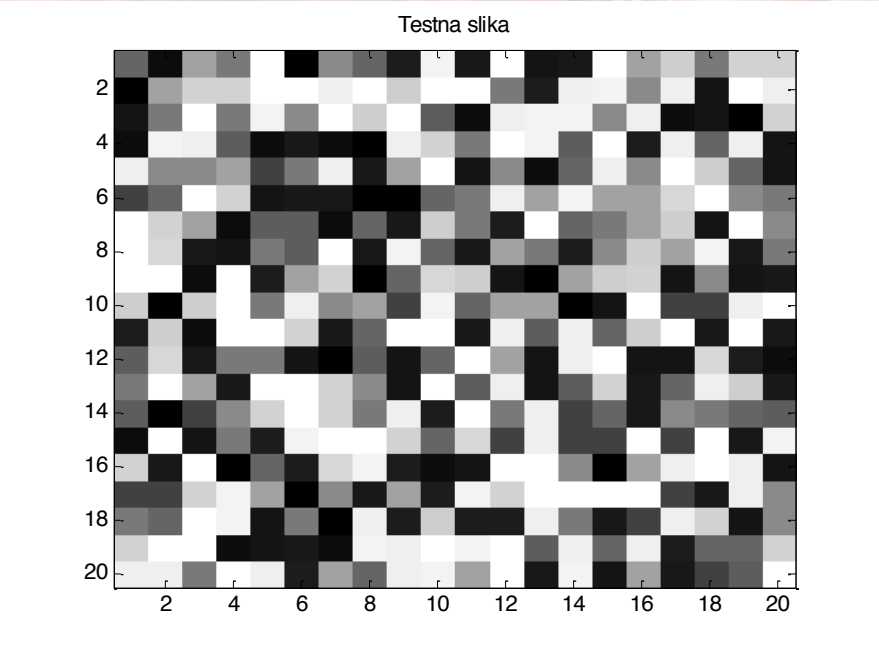
- That by chance our ideal coded message was mistakenly altered in such a way that we only raised the last bit of the message to 1
  - **.010101010001** (0.33227539062 5)
- ... this would have a fatal consequence on decoding, because the number is outside the final encoding interval [0.33184,0.3322) and is decoded as:
  - CAEFAEBEA$ and corresponds to a different 26-bit fraction :
  - **.010101010001**10000000000000

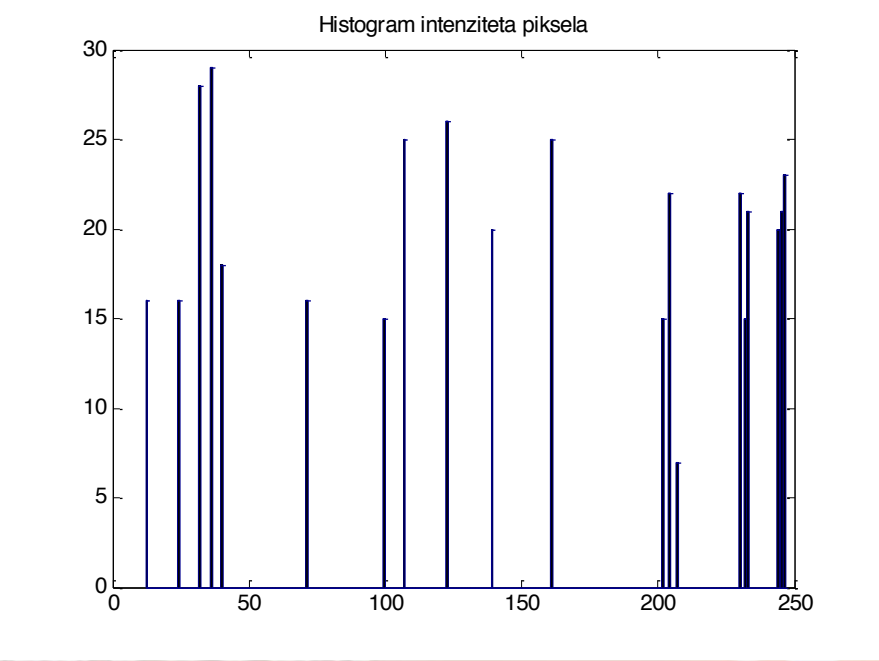
Arithmetic coding– problem

- Incorrectly decoded message:

Current value	Output symbol	Lower limit	Upper limit	Interval width
0.332275390625	C	0.3	0.5	0.2
0.161376953125	A	0	0.2	0.2
0.806884765625	E	0.55	0.85	0.3
0.85628255208333	F	0.85	0.9	0.05
0.12565104166668	A	0	0.2	0.2
0.62825520833339	E	0.55	0.85	0.3
0.26085069444465	B	0.2	0.3	0.1
0.60850694444649	E	0.55	0.85	0.3
0.19502314815497	A	0	0.2	0.2
0.97511574077485	$	0.9	1	0.1

Example of applying an entropy encoder to an image





The entropy of the image is 4.2688 bits per pixel, so the minimum number of bits for the whole image is 20*20*4.2688 = 1707.5, as opposed to 20*20*8 = 3200 without compression.

Conclusion

- The Huffman encoder and the arithmetic encoder are both entropy encoders:
  - compression is achieved by applying a code of variable length in accordance with the statistics of the source;
  - this shortens the average code length and brings the output data stream of the encoded message closer to the theoretical minimum defined by entropy;
  - an additional advantage of the arithmetic code is due to the fractional length of the code of each symbol.

What have we learned?

- definition of coding and compression
- types of coding
- information
- measure of information, entropy
- entropy encoders
- variable length codes
- Huffman codes
- Arithmetic encoder

Entropy and quality

prof.dr.sc. Davor Petrinović

Source coding for compression

- Problem description:
  - We have an amplitude continuous time discrete process, which we want to encode for compression purposes;
  - we transform the amplitude-continuous process into a discrete one by digitalization (quantization);
  - we want to apply an entropy encoder to the quantizer output, for compression.
  - What is the relationship between quality (quantization error) and entropy of the output coded message?

Source coding for compression

- Coding with **entropy constraint**
  - The channel has the finite available data rate (capacity) through which coded information is sent.
  - How to design a quantizer so that after entropy coding the message "fits" into the channel, while achieving the highest possible quality?
- Coding with **distortion constraint**
  - The reconstructed signal must have a quantization error less than the prescribed one.
  - How to design a quantizer that minimizes the required channel rate by which this is achieved, and how much is the required rate?

Source coding for compression

- The answer to these questions is given by **Information Theory** (IT), and especially its branch related to **source coding**.
- Part of IT is a theory that deals with the study of the relationship between quality and data rate and is called **Rate-Distortion theory**.
- Fundamental work in this area ... Claude Shannon, 1948-, while working in the Bell Laboratory.

Source coding for compression

- All modern encoders of media signals intensively use knowledge from Information Theory.
- We will illustrate the application of IT on the example of **Entropy Constrained Scalar Quantization (ECSQ)**.
- ECSQ has a number of practical applications in encoding, especially in audio signal encoding.
  - It has great efficiency in encoding longer sequences of statistically independent (or at least uncorrelated) data samples.

Entropy Constrained Scalar Quantization, ECSQ

- ECSQ quantization is typically applied to:
  - signal samples in the original time or spatial domain,
  - on the signal transformation coefficients in one of the transformation domains,
  - to samples of subband signals in structures that use filter-banks for spectral decomposition of signals,
  - for quantization of prediction error signal samples in encoders that use predictors for the purpose of temporal or spatial decorrelation,
  - on model parameters, in codecs based on a model with a large number of parameters.

Entropy Constrained Scalar Quantization, ECSQ

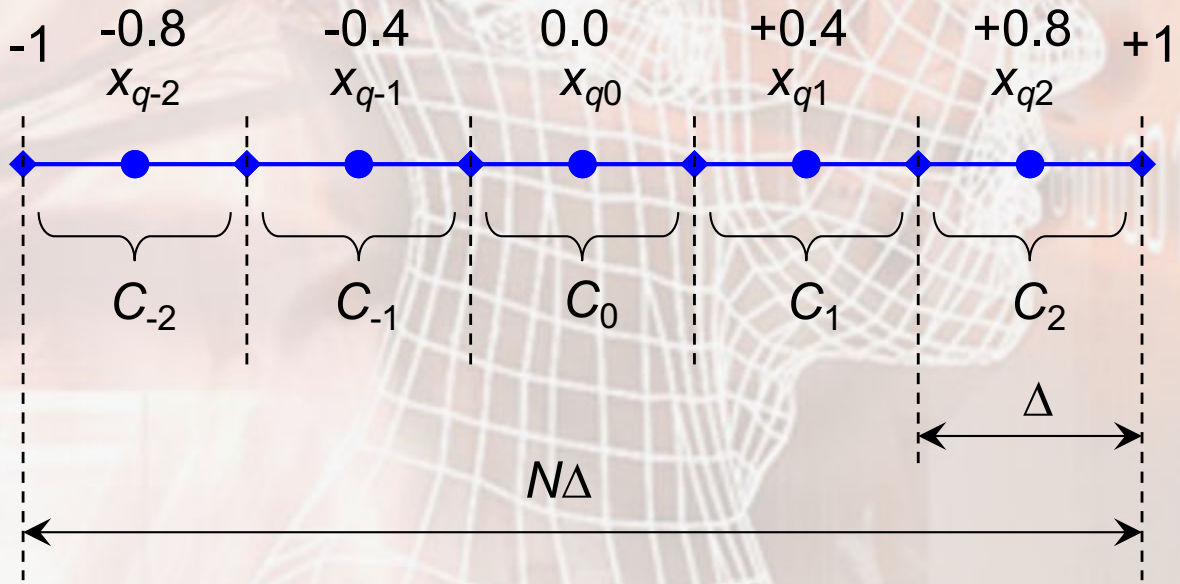
- The input to the encoding process is a digitized media signal of nominal accuracy;
  - Although this signal is already amplitude-discretized by the input A/D converter, it is treated as “infinitely” accurate because …
  - the input resolution is generally much higher than the resolution of the ECSQ quantizer used within the compression encoder.
  - Signed normalized fractional interpretation of the input signal (+/-1) is usually assumed, by dividing the integer value with 2^{b-1}

Uniform scalar quantization-example

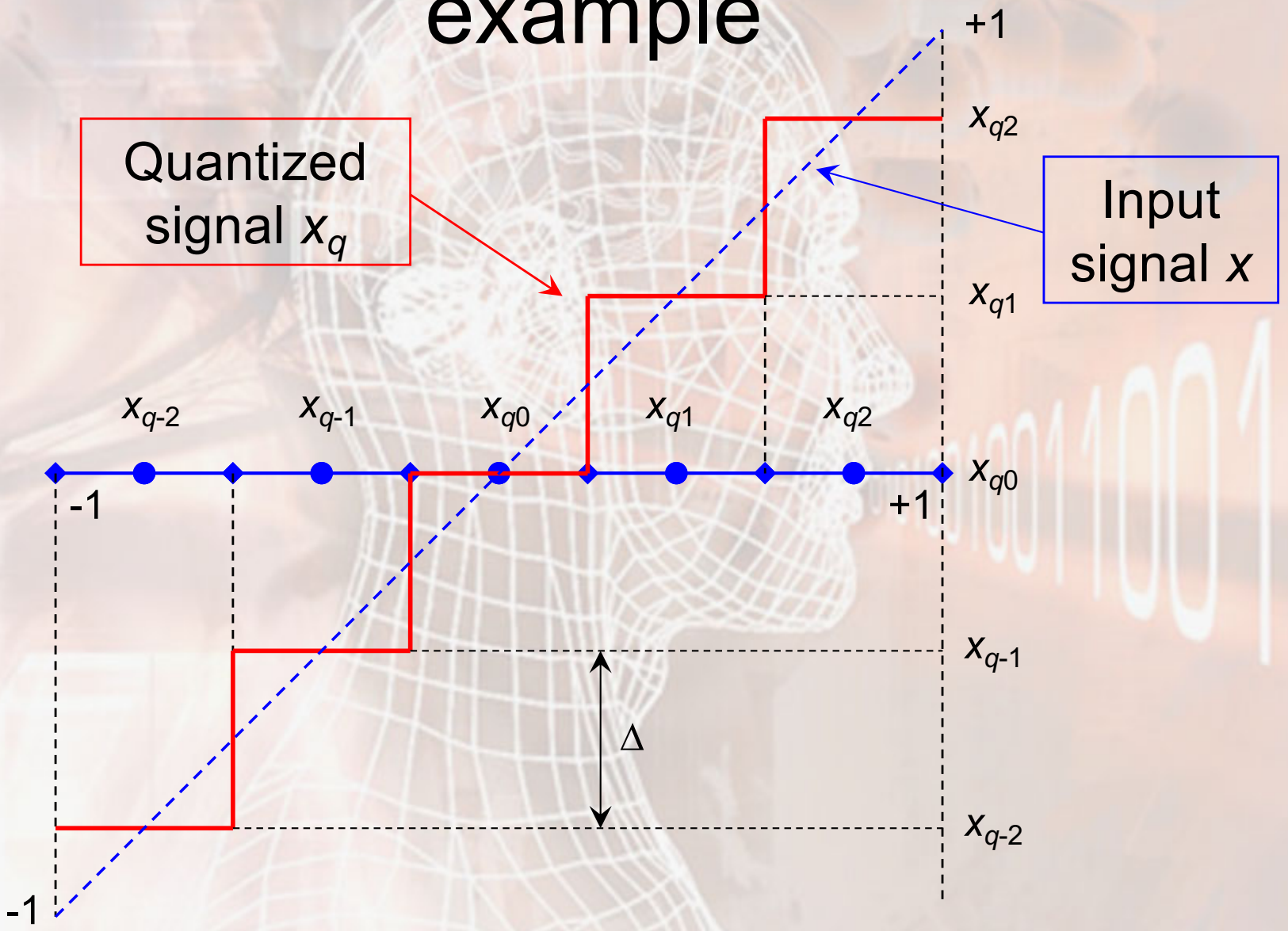
- Example:
  - The input process in the fractional notation +/- 1 is quantized with a uniform quantizer with N=5 quantization levels, i.e., with a step size $\Delta=2/N=0.4$
  - quantization classes are $C_i=[\Delta\cdot i-\Delta/2,\ \Delta\cdot i+\Delta/2)$
    - C_{-2}=[-1.0, -0.6) ..... code **-2** (centroid x_{q-2}= -0.8)
    - C_{-1}=[-0.6, -0.2) ..... code **-1** (centroid x_{q-1}= -0.4)
    - C_0 =[-0.2, +0.2) ..... code **0** (centroid x_{q0}= 0)
    - C_1 =[+0.2, +0.6)..... code **1** (centroid x_{q1}= 0.4)
    - C_2 =[+0.6, +1.0] ..... code **2** (centroid x_{q2}= 0.8)
  - Reconstruction is performed by **replacing the code *i* with the corresponding centroid** of the *i*-th class $x_{qi}=\Delta\cdot i$

Uniform scalar quantization-example

- ... continued:
  - uniform arrangement of quantization classes C_i over the value range of the input process X:



Uniform scalar quantization-example



Uniform scalar quantization-example

- The output process of quantization codes $I=\{i\}$ has five symbols $i \in \{-2, -1, 0, 1\ 2\}$
- What is the entropy of this discrete source I?
- ... is determined by the probabilities of individual symbols

$$H(I) = -\left(\begin{array}{l} p_I(-2)\log p_I(-2) + p_I(-1)\log p_I(-1) + \\ + p_I(0)\log p_I(0) + p_I(1)\log p_I(1) + p_I(2)\log p_I(2) \end{array} \right)$$

- the probability of the symbol $p_I(i)$ can be determined by integrating the *pdf* function of the process $f_X(x)$ within the i-th class

$$p_I(i) = \int_{x \in C_i} f_X(x)dx = \int_{\Delta i - \Delta/2}^{\Delta i + \Delta/2} f_X(x)dx$$

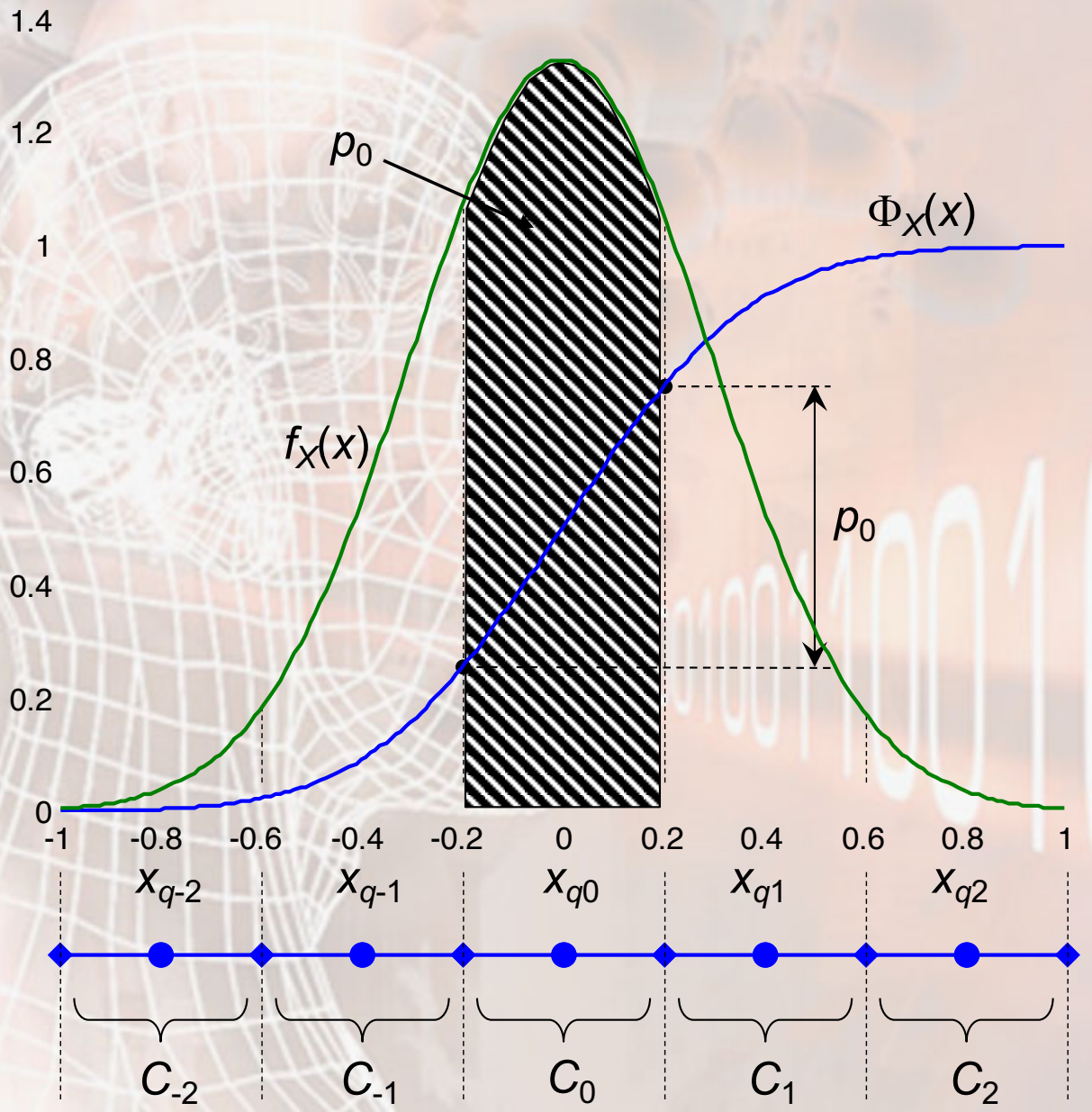
- e.g., probability of symbol *i*=0 is:

$$p_I(0) = \int_{-\Delta/2}^{+\Delta/2} f_X(x)dx$$

- or by differentiating the *cdf* of the process at the boundaries of the class:

$$p_I(0) = \Phi_X(\frac{\Delta}{2}) - \Phi_X(-\frac{\Delta}{2})$$

$$\Phi_X(x) = \int f_X(x)dx$$



Uniform scalar quantization-example

- ... continued:
  - Entropy $H(I)$ is greatest when all output symbols are equally likely $p_I(i)=1/N$, for i=-2, -1, 0, 1 i 2;
  - This is the case when the process has a uniform density:

$$f_X(x) = \frac{1}{\Delta N} \qquad p_I(i) = \frac{1}{\Delta N} \int_{\Delta i - \Delta/2}^{\Delta i + \Delta/2} dx = \frac{1}{\Delta N} \Delta = \frac{1}{N}$$

  - We get: .... max($H(I)$)=log(N)=2.32 bit
  - Note: if we use the natural logarithm in the expression for entropy, $H(I)$ is expressed in ***nats*** (**n**atural **u**nit of **i**nformation), and if we use the logarithm of base 2, then it is in ***bits***!

Uniform scalar quantization-example

- ... continued:
  - What is the distortion *D* due to quantization ?

$$D = E[(x - x_q)^2] = \int_x f_X(x) \cdot (x - x_q)^2 \, dx$$

$$x_q = \Delta \cdot \text{round}(x / \Delta)$$

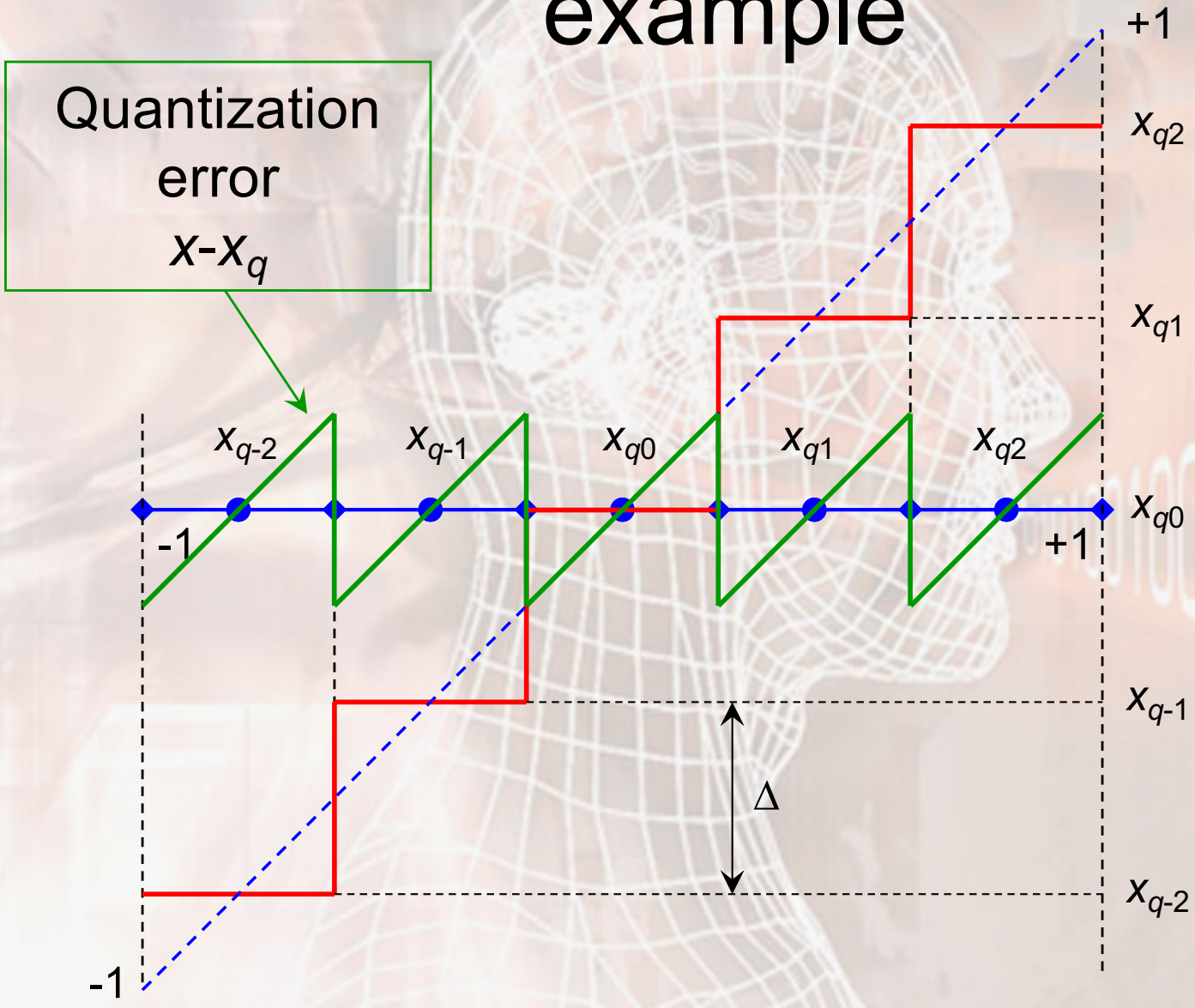
  - The quantized signal x_q is from a finite set of centroids:

$$x_q \in \{x_{q_i}\} = \{x_{q_{-2}}, x_{q_{-1}}, x_{q_0}, x_{q_1}, x_{q_2}\}$$
$$= \{-2\Delta, -\Delta, 0, \Delta, 2\Delta\}$$

  - so, it is more convenient to express the distortion *D* as the sum of the total distortion within the individual quantization classes D_i

$$D = \sum_i D_i = \sum_i \int_{x \in C_i} f_X(x) \cdot (x - x_{q_i})^2 \, dx$$

Uniform scalar quantization-example



Uniform scalar quantization-example

- ... continued:
  - For a signal with the uniform probability density *pdf* $f_X(x)=1/(\Delta N)$, the distortion can be easily determined:

$$D=\sum_i \frac{1}{\Delta N}\int_{\Delta i-\Delta/2}^{\Delta i+\Delta/2}(x-\Delta i)^2\,dx=\sum_i \frac{1}{\Delta N}\int_{-\Delta/2}^{+\Delta/2}x'^2\,dx'=\sum_i \frac{\Delta^2}{12N}=\frac{\Delta^2}{12}$$

  - while the variance of the input process is found as:

$$\sigma_x^2=E[(x-E[x])^2]=E[x^2]=\int_x f_X(x)x^2\,dx$$

$$\sigma_x^2=\frac{1}{\Delta N}\int_{-\Delta N/2}^{\Delta N/2}x^2\,dx=\frac{(\Delta N)^2}{12}$$

Uniform scalar quantization-example

– Finally, we obtain the *SQNR* ratio for the uniform probability density process $f_X(x)=1/(\Delta N)$ quantized by a uniform quantizer with *N* levels that completely cover the value range of the process *X*:

$$SQNR(N) = 10\log_{10}\frac{\sigma_x^2}{D} = 20\log_{10}(N) \quad [dB]$$

– with the entropy of the output symbols *H*(*I*):

$$H(I) = \log_2(N) \quad [bit]$$

– which gives the following simple relation:

$$SQNR(N) = (20\log_{10} 2)H(I) = 6.02H(I) \quad [dB]$$

Uniform scalar quantization-example

- Discussion:
  - entropy expressed in bits and quantization distortion expressed in dB are linearly related;
  - to reduce the distortion by 6dB it is necessary to increase the output entropy by 1 bit;
  - due to the uniform distribution of the input process, the entropy encoder does not achieve compression with a variable code length, because the codes of all symbols should be of the same length;
  - the output entropy of any other input process must be lower;
  - the distortion for an arbitrary $f_X(x)$ must be determined by summing the expected error D_i found by the integration within each quantization class.

What have we learned?

- coding with entropy or distortion constraint
- information theory, branches and applications
- scalar quantization with constrained entropy and its applications
- uniform scalar quantization - an example
- relationship of quality and entropy

High rate (quality) quantization theory

prof.dr.sc. Davor Petrinović

Entropy Constrained Scalar Quantization, ECSQ

- An issue ...
  - what if the *pdf* function of the process is not uniform;
  - what if the process does not have a bounded domain;
  - is a uniform arrangement of quantizer centroids and associated classes the best solution for an arbitrary process *pdf* ?
  - how must centroids be arranged to minimize the error *D* for a given entropy *H*(*I*), or vice versa to minimize the entropy for a given distortion *D*?

Entropy Constrained Scalar Quantization, ECSQ

- Unfortunately, there is no analytical solution for the general case,
- ... but by applying the ***High Rate Quantization Theory*** (HR), an asymptotic solution for high resolutions (i.e., for high quality reconstruction) can be determined.
- This solution proves to be useful for evaluating encoder behavior even for very low resolutions!

HR-ECSQ

- The key idea of HR theory is the following:
  - if the quantization classes are narrow enough, then when integrating within one class it can be assumed that the probability density function is approximately constant and can be approximated from the value of the pdf function at the centroid point of that class:

$$f_X(x)\Big|_{x\in C_i} \cong f_X(x_{q_i})$$

  - This greatly simplifies the calculation of integrals in entropy and distortion expressions.

HR-ECSQ

- With the described assumption of high data rate, we obtain the following solution :
  - ... indeed, a quantizer with a uniform spacing of quantization levels Δ, is the best solution for **every input process**!
  - The ratio of the output entropy $H(I)$ and the distortion D due to quantization with step Δ is:

$$H(I) = h(X) - \frac{1}{2}\log_2(12D) \qquad D = \frac{\Delta^2}{12}$$

  - where $h(X)$ is the differential entropy of process X

Differential entropy

- Differential entropy is a generalization of the expression for entropy to the processes of a continuous variable ...
  - the expression is completely analogous, except that the sum is replaced by the integral over the value range of the process:

$$h(X) = -\int_x f_X(x) \cdot \log(f_X(x))\, dx$$

$$h(X) = E[-\log(f_X(x))]$$

HR-ECSQ

- The expression for the relationship between entropy and quality of ECSQ quantizers can also be written in the following form :

$$SQNR = (20\log_{10} 2)H(I) + SQNR_0 \quad [dB]$$

  - the offset of this linear expression, $SQNR_0$, depends on the properties of the input process, ... its **variance and differential entropy**:

$$SQNR_0 = (20\log_{10} 2)\cdot(\frac{1}{2}\log_2(12\sigma_x^2) - h(X))$$

$$SQNR_0 = 20\log_{10}(\sqrt{12\sigma_x^2}\,2^{-h(X)})$$

HR-ECSQ

- Consider the influence of the $SQNR_0$ ...
  - higher offset $SQNR_0$, provides better quality for the same output entropy $H(I)$;
  - for processes with the same variance σ_x^2, lower $h(X)$ implies higher $SQNR_0$, and thus better quality;
  - Thus, the differential entropy $h(X)$ is a **measure of the complexity of the input process** from the point of view of quantization efficiency and entropy coding.

HR-ECSQ

- How to determine the ECSQ quantizer step size for a given channel capacity, i.e., for constrained output entropy $H(I)$?
  - from the expression for the relationship between distortion and entropy it follows :

$$\Delta = 2^{h(X)-H(I)}$$

  - where the differential entropy $h(X)$ is determined from the known probability density function of the input process $f_X(x)$.

HR-ECSQ

- How to determine the required output entropy $H(I)_{\min}$ and the step of the ECSQ quantizer Δ for a specified minimum quality $SQNR_{min}$?
  - Based on $f_X(x)$ we calculate $h(X)$ and determine the offset of the linear relationship between entropy and *SQNR:*

$$SQNR_0 = 20\log_{10}\left(\sqrt{12\sigma_x^2}\, 2^{-h(X)}\right)$$

  - Entropy $H(I)_{\min}$ and step size Δ are then found as:

$$H(I)_{\min} = \frac{SQNR_{\min} - SQNR_0}{20\log_{10} 2}$$

$$\Delta = 2^{h(X) - H(I)_{\min}}$$

Differential entropy - example 1

- Let us determine the differential entropy for several typical processes of variance σ_x^2.
- Uniform probability density process:

$$f_{X_{unif}}(x) = \begin{cases} \dfrac{1}{\sqrt{12\sigma_x^2}} & \text{za } |x| < \sqrt{3\sigma_x^2} \\ 0 & \text{otherwise} \end{cases}$$

$$\Phi_{X_{unif}}(x) = \begin{cases} \dfrac{x+\sqrt{3\sigma_x^2}}{\sqrt{12\sigma_x^2}} & \text{za } |x| < \sqrt{3\sigma_x^2} \\ 0 & \text{otherwise} \end{cases}$$

Differential entropy - example 1

– The differential entropy for the process of uniform probability density, zero expectation and given variance σ_x^2 is:

$$h(X_{unif}) = \frac{\log_2(12\sigma_x^2)}{\sqrt{12\sigma_x^2}} \int_{-\sqrt{3\sigma_x^2}}^{\sqrt{3\sigma_x^2}} dx = \log_2 \sqrt{12\sigma_x^2} \ [bit]$$

$$SQNR_{0_{unif}} = 10\log_{10}(12\sigma_x^2 \cdot 2^{-2h(X_{unif})})$$

$$= 10\log_{10}(12\sigma_x^2 \cdot 2^{-2\log_2\sqrt{12\sigma_x^2}}) = 0 \ [dB]$$

Differential entropy - example 1

- It is obvious that the relationship between the quality *SQNR* and entropy *H*(*I*) has a zero offset $SQNR_{0unif}$, which is equal to the result we derived earlier in the example for the case of the uniform distribution process without HR assumption.
- The quantization step size Δ for a given output entropy of quantization indices *H*(*I*) for the process of uniform distribution and variance σ_x^2 is calculated according to the following expression:

$$\Delta_{unif} = 2^{h(X_{unif})} \cdot 2^{-H(I)} = \sqrt{12\sigma_x^2} \cdot 2^{-H(I)}$$

Differential entropy - example 2

– Random process of normal distribution with zero expectation and variance σ_x^2, so-called Gaussian white noise:

$$f_{X_{norm}}(x)=\frac{1}{\sqrt{2\pi\sigma_x^2}}\cdot e^{-\frac{x^2}{2\sigma_x^2}}$$

$$\Phi_{X_{norm}}(x)=\frac{1}{2}\left(1+\mathrm{erf}\left(\frac{x}{\sqrt{2\sigma_x^2}}\right)\right)$$

$$\mathrm{erf}(z)=\frac{2}{\sqrt{\pi}}\int_0^z e^{-t^2}\,dt$$

– where erf( ) is a so-called *error function* that is numerically implemented in Matlab.

Differential entropy - example 2

– The differential entropy for the random process of normal distribution, zero expectation and variance σ_x^2 is:

$$h(X_{norm}) = \frac{-1}{\sqrt{2\pi\sigma_x^2}} \int_{-\infty}^{+\infty} e^{-\frac{x^2}{2\sigma_x^2}} \log_2 \left(\frac{1}{\sqrt{2\pi\sigma_x^2}} e^{-\frac{x^2}{2\sigma_x^2}} \right) dx =$$

$$h(X_{norm}) = \frac{-1}{\ln(2)\sqrt{2\pi\sigma_x^2}} \int_{-\infty}^{+\infty} e^{-\frac{x^2}{2\sigma_x^2}} \left(-\frac{1}{2}\ln(2\pi\sigma_x^2) - \frac{x^2}{2\sigma_x^2} \right) dx =$$

Differential entropy - example 2

– continued ... :

$$h(X_{norm})=\frac{\frac{1}{2}\ln(2\pi\sigma_x^2)}{\ln(2)}\cdot\frac{1}{\sqrt{2\pi\sigma_x^2}}\int_{-\infty}^{+\infty}e^{-\frac{x^2}{2\sigma_x^2}}dx+\frac{1}{\ln(2)2\sigma_x^2}\cdot\frac{1}{\sqrt{2\pi\sigma_x^2}}\int_{-\infty}^{+\infty}x^2e^{-\frac{x^2}{2\sigma_x^2}}dx$$

1

$$\int_{-\infty}^{+\infty}x^2e^{-\frac{x^2}{2\sigma_x^2}}dx=\left(-\sigma_x^2x\cdot e^{-\frac{x^2}{2\sigma_x^2}}+\frac{\sqrt{2\pi\sigma_x^6}}{2}\operatorname{erf}\left(\frac{x}{\sqrt{2\sigma_x^2}}\right)\right)\Bigg|_{-\infty}^{+\infty}=$$

$$=\left(-\sigma_x^2x\cdot(e^{-\infty}-e^{-\infty})+\frac{\sqrt{2\pi\sigma_x^6}}{2}(\operatorname{erf}(\infty)-\operatorname{erf}(-\infty))\right)=\sqrt{2\pi\sigma_x^6}$$

0 1 -1

Differential entropy - example 2

– continued ... we finally get $h(X_{norm})$:

$$h(X_{norm}) = \frac{1}{2}\log_2(2\pi\sigma_x^2) + \frac{1}{\ln(2)2\sigma_x^2} \cdot \frac{1}{\sqrt{2\pi\sigma_x^2}} \sqrt{2\pi\sigma_x^6}$$

$$= \frac{1}{2}\log_2(2\pi\sigma_x^2) + \frac{1}{2}\log_2(e) = \frac{1}{2}\log_2(2\pi e\sigma_x^2) \; [bit]$$

$$SQNR_{0\,norm} = 10\log_{10}(12\sigma_x^2 \cdot 2^{-2h(X_{norm})})$$

$$= 10\log_{10}(12\sigma_x^2 \cdot 2^{-\log_2 2\pi e\sigma_x^2})$$

$$= 10\log_{10}(\frac{12\sigma_x^2}{2\pi e\sigma_x^2}) = 10\log_{10}(\frac{6}{\pi e}) = -1.53 \; [dB]$$

Differential entropy - example 2

- Finally, we find the expression for the quantization step size Δ for a given entropy $H(I)$ for noise of normal distribution and variance σ_x^2 :

$$\Delta_{norm} = 2^{h(X_{norm})} \cdot 2^{-H(I)} = \sqrt{2\pi e \sigma_x^2} \cdot 2^{-H(I)}$$

- In conclusion, ... the normal distribution process has 1.53dB higher distortion than the uniform distribution process for the same variance and the same output entropy.
- This is because the quantization step Δ_{norm} must be 1,193 times larger than Δ_{unif} to get the same entropy $H(I)$.

$$\frac{\Delta_{norm}}{\Delta_{unif}} = \frac{\sqrt{2\pi e \sigma_x^2}}{\sqrt{12\sigma_x^2}} = \sqrt{\frac{\pi e}{6}} = 1.193$$

HR-ECSQ

- It can be theoretically proven that the process of normal probability density is the most complex for ECSQ quantization of all processes with the same variance!
- This is because it has the largest possible differential entropy *h*(*X*).
- So, ... for the input process of an unknown *pdf* function for which we know only the variance σ_x^2 it always holds

$$h(X) \leq h(X_{norm}) = \log_2 \sqrt{2\pi e \sigma_x^2} \ [bit]$$

- ... which can serve as an upper bound when designing a quantizer.

HR-ECSQ

- In Example 2, we showed:
  - that a scalar quantizer with constrained output entropy can be designed and theoretically analyzed even for continuous processes with an infinite domain, which is the case for the Gaussian process;
  - in practice, due to the design of the entropy encoder, the set of output symbols still must be limited.

Generating processes in Matlab

- Generate realization of three different processes of unit variance and zero mean value :

```
n=[0:N-1];      % vremenska os

w=pi*(0.1+0.8*rand);% slucajna frekvencija za sinus i
                    % trokut ... od 0.1*pi do 0.9*pi
if (sig==1),        % sinusni signal
   y=sin(w*n);      % vrem. diskretni signal
elseif (sig==2),    % slucajni sum normalne razdiobe
   y=randn(1,N);    % generiraj sum
elseif (sig==3),    % pilasti napon
   fa=w*n;          % trenutna faza
   fa=fa-2*pi*floor(fa/2/pi); % osnovna vrijednost faze
   y=fa/pi-1;       % pila od -1 do 1
end;

y=y-mean(y);        % ukloni srednju vrijednost
y=y/sqrt(var(y));   % normaliziraj varijancu na 1
```

Design of ECSQ quantizers in Matlab

- Calculation of differential entropy $h(X)$ and required quantization step size Δ:

```
% Diferencijalne entropije za nase signale

sigma=1;
if (sig==1),          % za sinus
   hX=1/2*log2(pi^2*sigma^2/2);
elseif (sig==2),      % sum normalne razdiobe
   hX=1/2*log2(2*pi*exp(1)*sigma^2);
elseif (sig==3),      % trokutni signal
   hX=1/2*log2(12*sigma^2);
end;

% Korak kvantizacije za zadanu izlaznu entropiju
% H nalazimo kao:
D=2^(hX-H);
```

Quantization of the process in Matlab

- Calculation of expected and actual quantization error and the implementation of scalar quantization itself :

```
% Ocekivana srednja kvadratna pogreska kvantizacije
% je direknto odredjena sa korakom D
E_er_oc=1/12*D^2;

% Obzirom da su signali jedinicne varijance,
% ocekivani SNR za taj korak kvantizacije je:
SNR_oc(i)=10*log10(1/E_er_oc); % [dB]

% sada provedi stvarnu kvantizaciju realizacije procesa
yi=round(y/D);          % izlazni indeksi kvantizatora
yq=D*yi;                % kvantizirani signal

er=y-yq;                % pogreska kvantizatora
E_er=mean(er.^2);       % ocekivanje kvadrata pogreske
E_y=mean(y.^2);         % ocekivanje kvadrata signala

SNR(i)=10*log10(E_y/E_er); % Stvarni odnos signal sum [dB]
```

Output entropy

- Calculation of real *pdf* and entropy of output indices of quantized realization of the process using histograms:

```
% min & max izlazni kod
yi_min=min(yi);  yi_max=max(yi);

% svi kodovi od min do max
kod=[yi_min:yi_max];

% odredi vjerojatnost pojave pojedinog koda u
% ovoj realizaciji slucajnog procesa koristenjem
% histograma
pdf=hist(yi,kod)/N;

% Odredi entropiju realizacije
kod_postoji=find(pdf>0);
HI(i)=-pdf(kod_postoji)*log2(pdf(kod_postoji))';
```

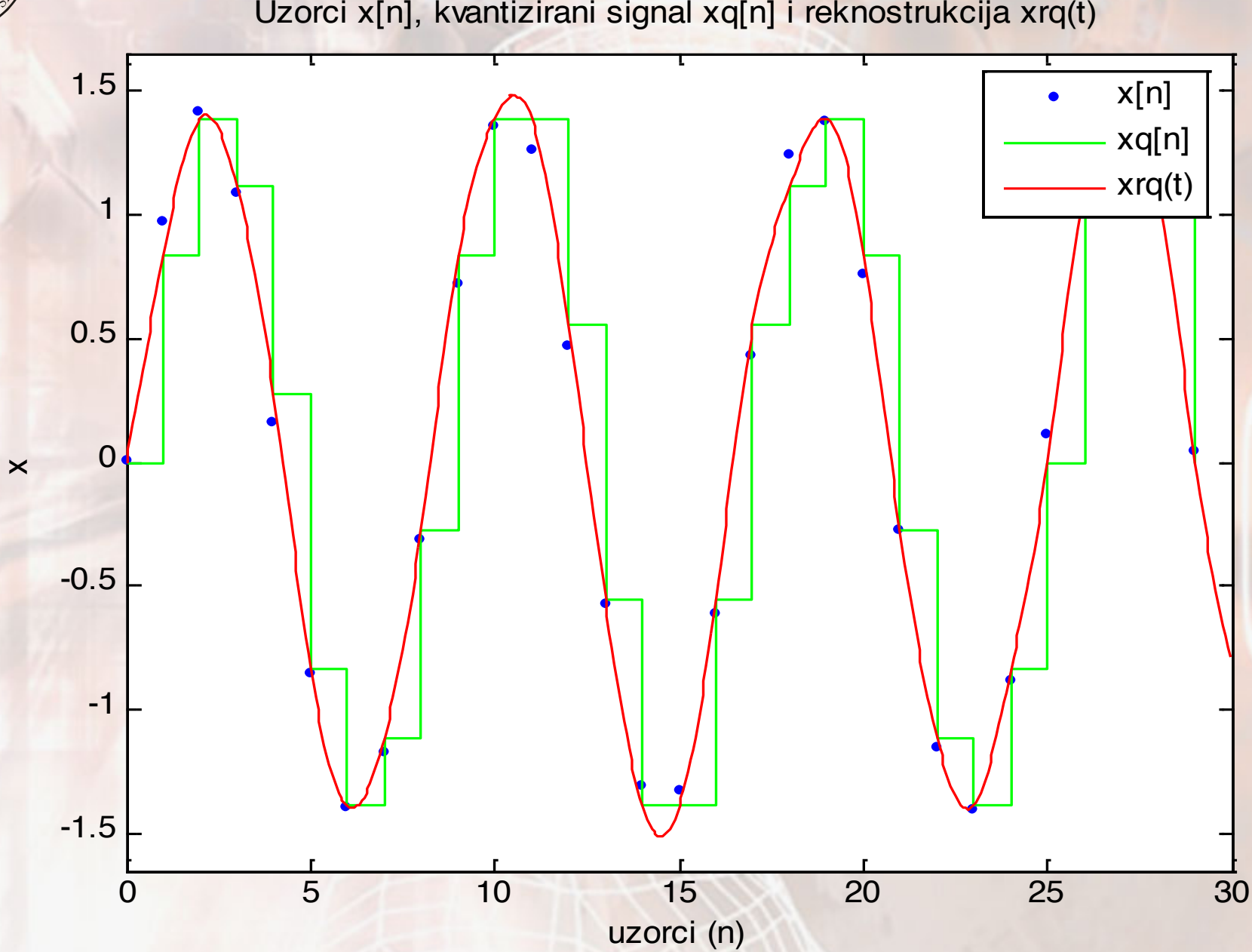
Expected output entropy

- Calculation of expected *pdf* and entropy of output indices by differentiation of analytical *cdf* of input process at the edges of quantization classes :

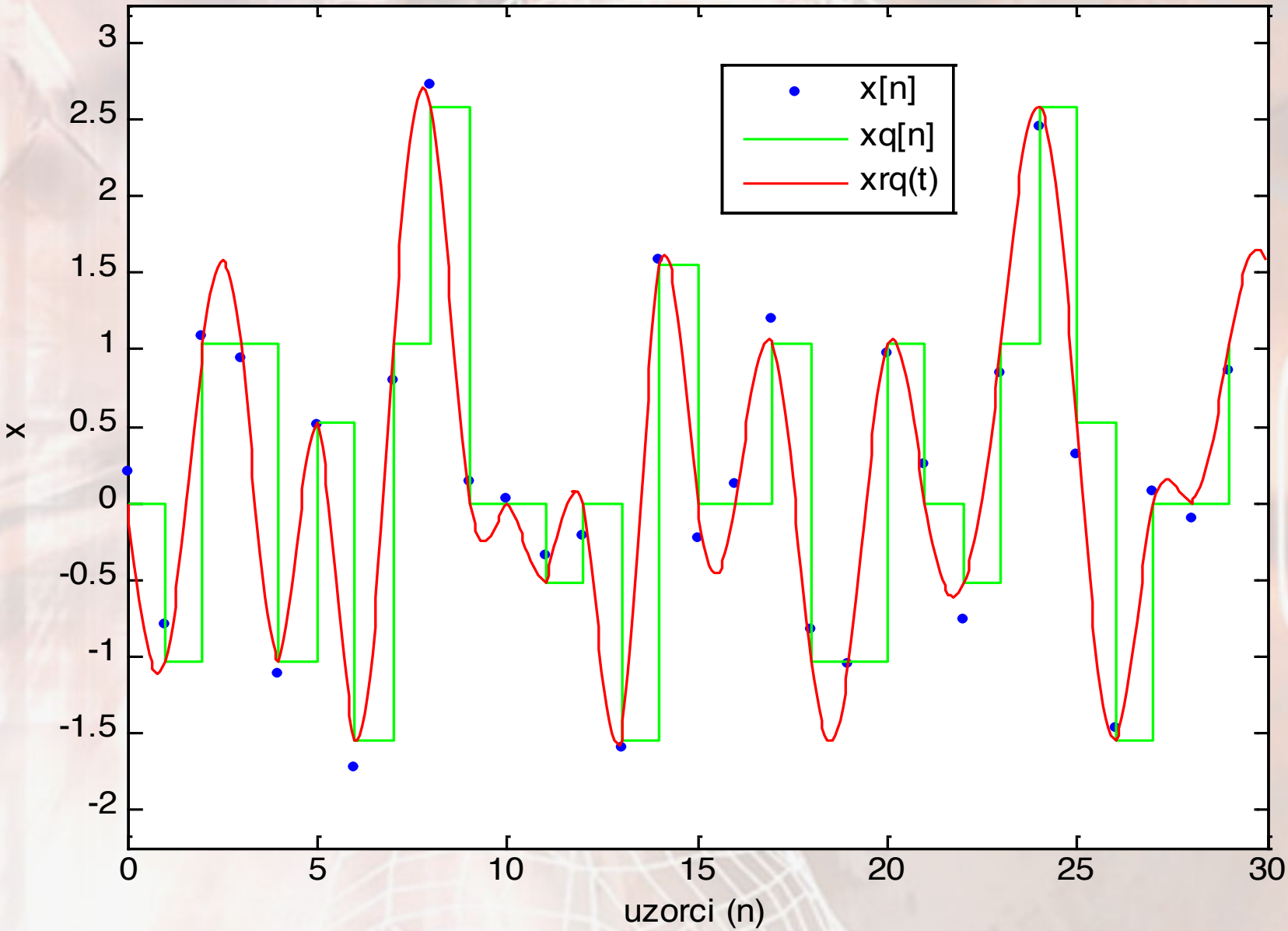
```
delt=max(abs(y));          % tjemeni faktor
% Pragovi kvantizacije u rasponu koristenih kodva
prag=[yi_min-1/2 [yi_min:yi_max]+1/2]*D;

arg=prag/delt;             % argument acos-a i trokuta mora biti
arg=max(min(arg,1),-1); % unutar +/- 1
% Otipkaj cdf funkciju nasih signala jedinicne varijace
if (sig==1),               % za sinusni signal
   cdf_prag=acos(-arg)/pi;
elseif(sig==2),            % za sum normalne razdiobe
   cdf_prag=1/2*(1+erf(prag/sqrt(2)));
elseif(sig==3),            % za pilasti signal
   cdf_prag=(1+arg)/2;
end;
% ocekivana vjerojatnost pojedinog simbola je
% diferencija cdf-a izmedju susjenih pragova
pdf_oc=diff(cdf_prag);
HI_oc(i)=-pdf_oc*log2(pdf_oc)';   % ocekivana izlazna entropija
```

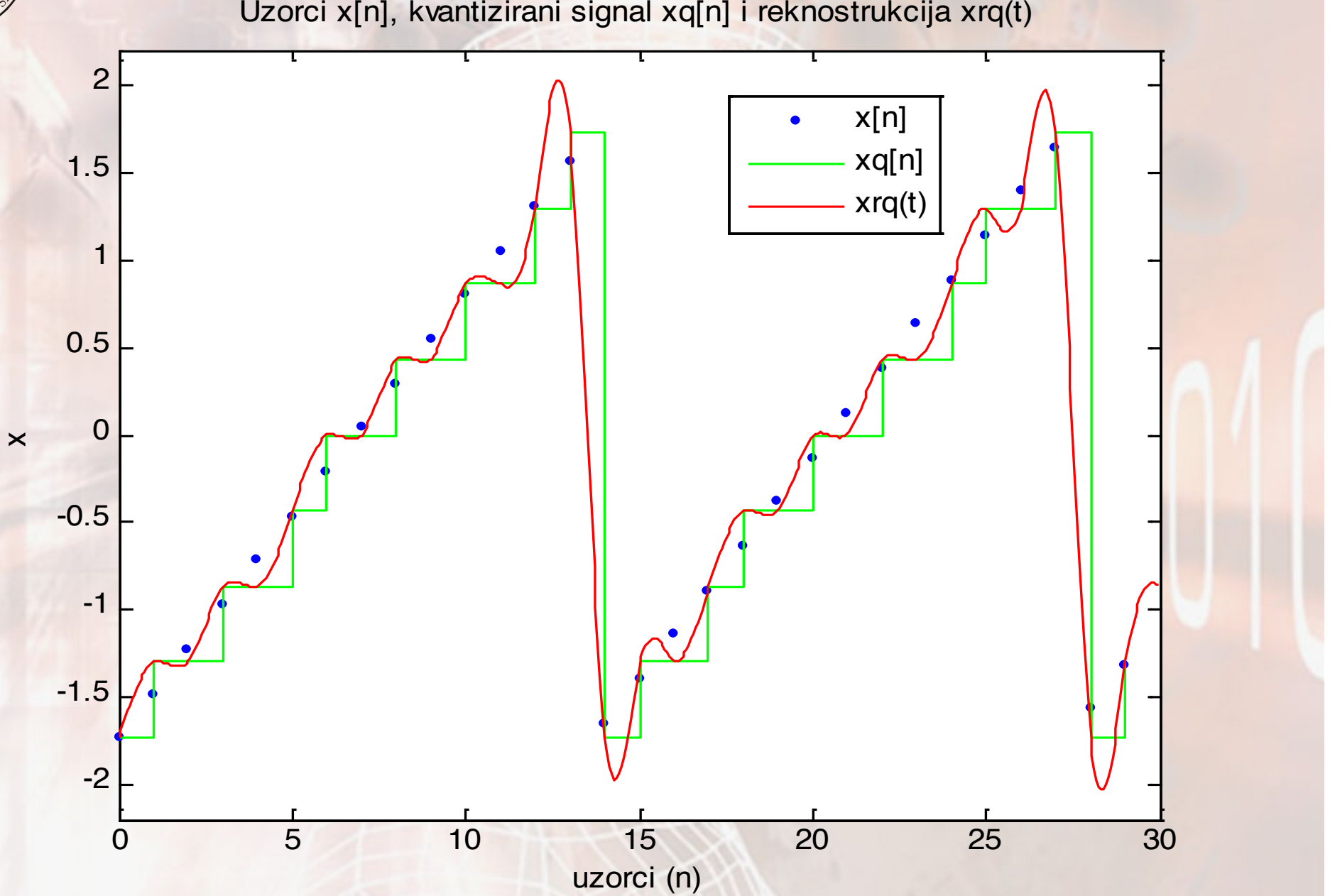
Process 1 – sine signal $\sigma_x^2=1$, $H(I)=3$



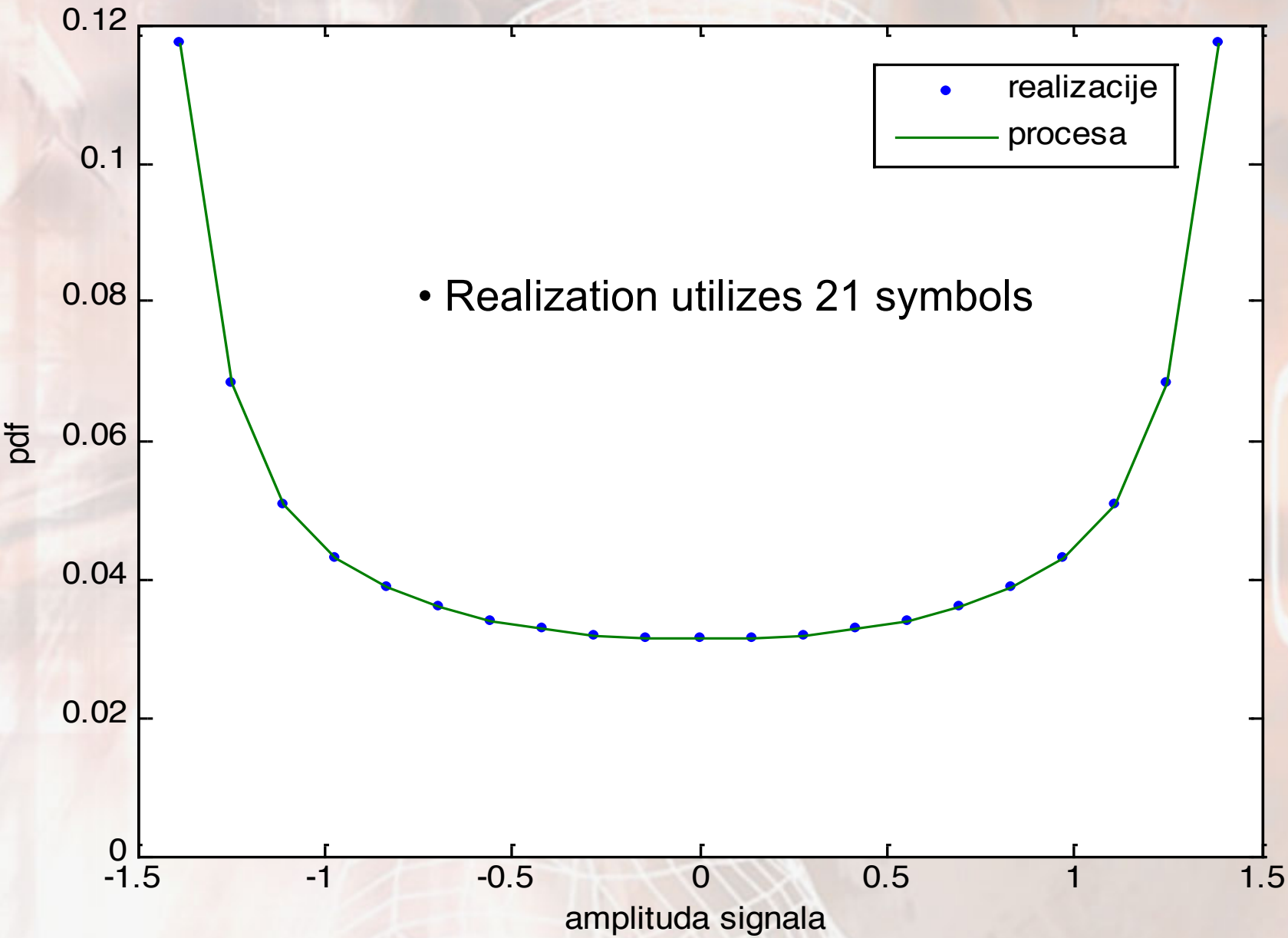
Process 2 – random noise $\sigma_x^2=1$, $H(l)=3$



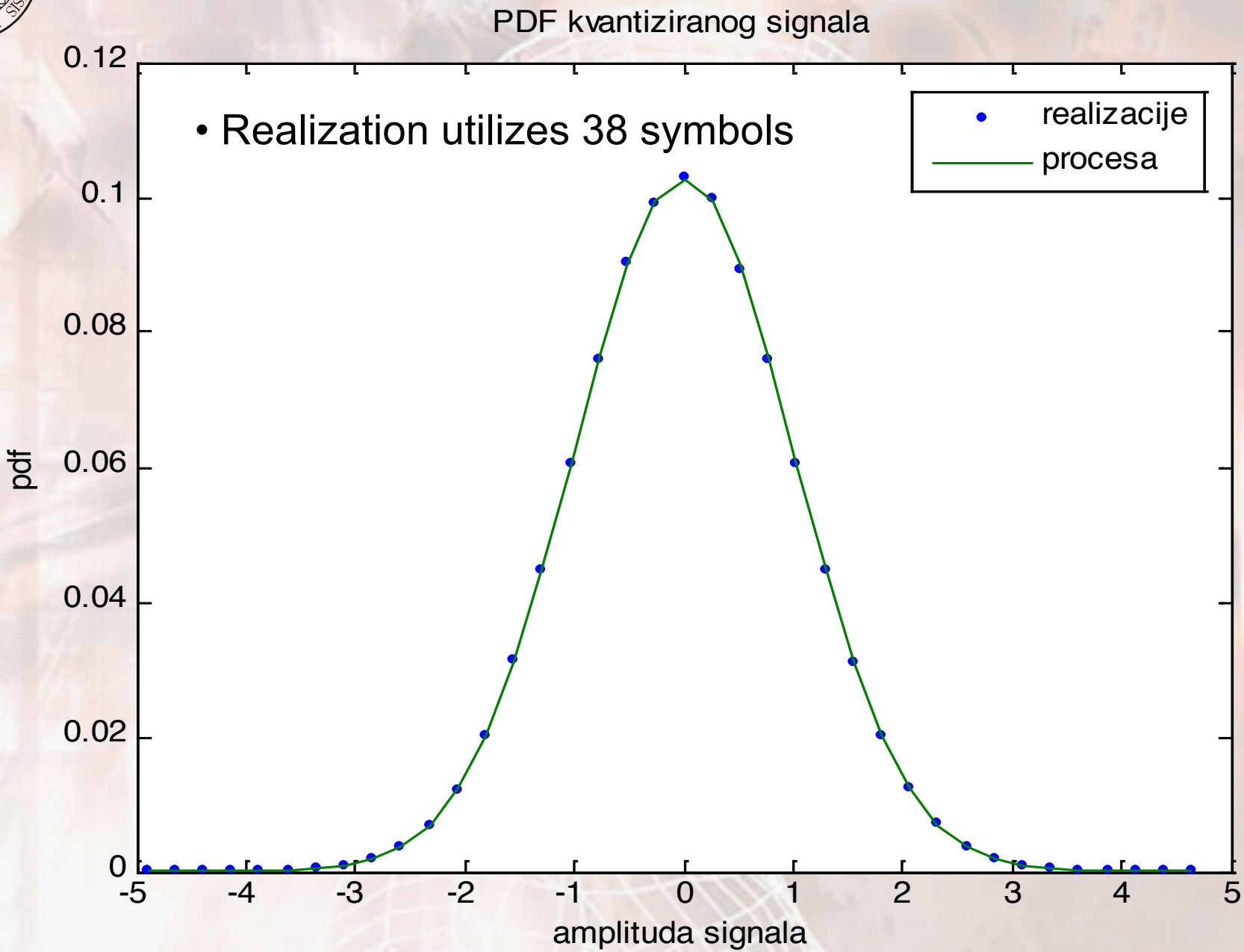
Process 3 – triangular signal $\sigma_x^2=1$, $H(l)=3$



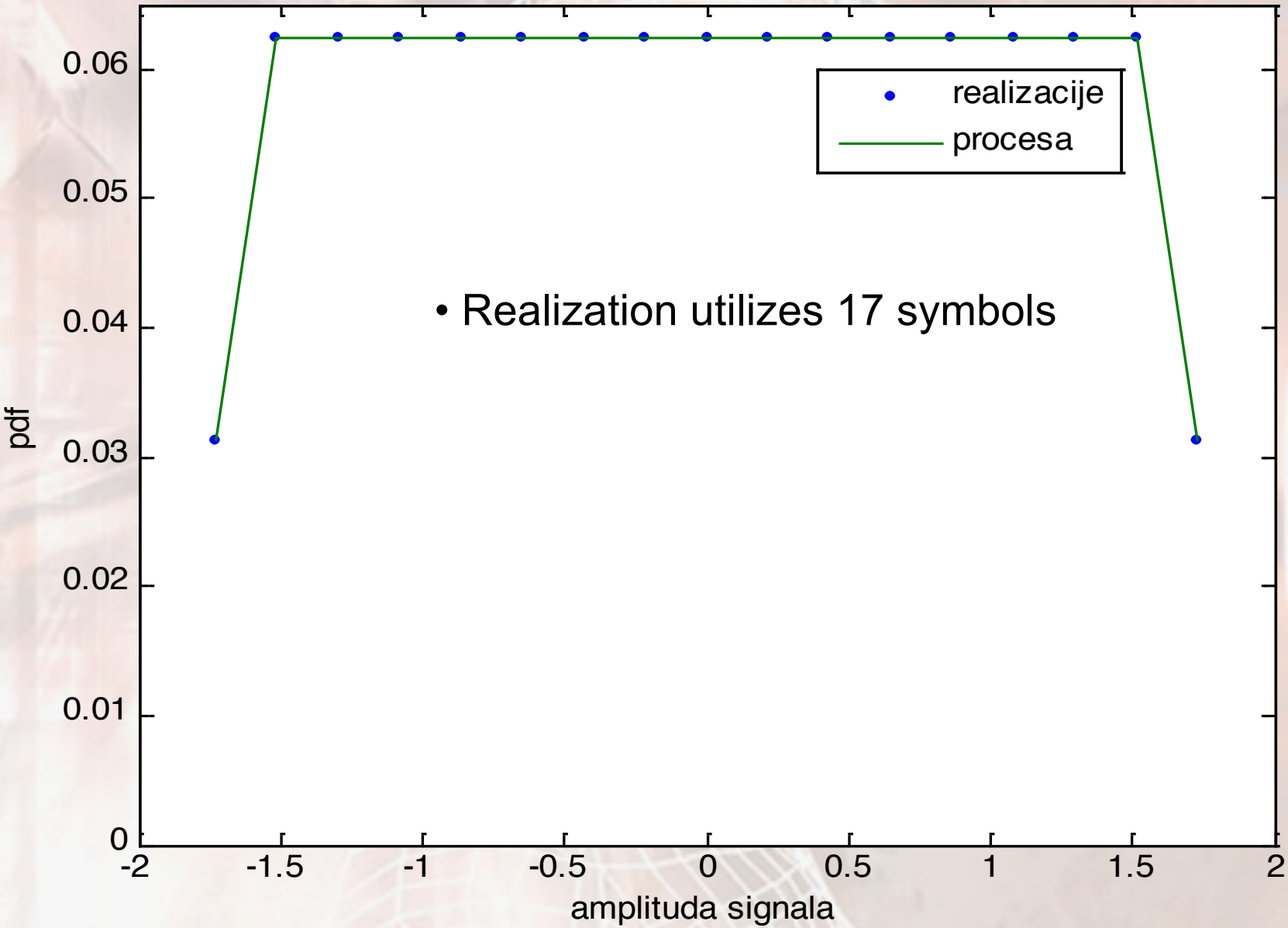
1. Sine – pdf of output symbols $H(I)=4$



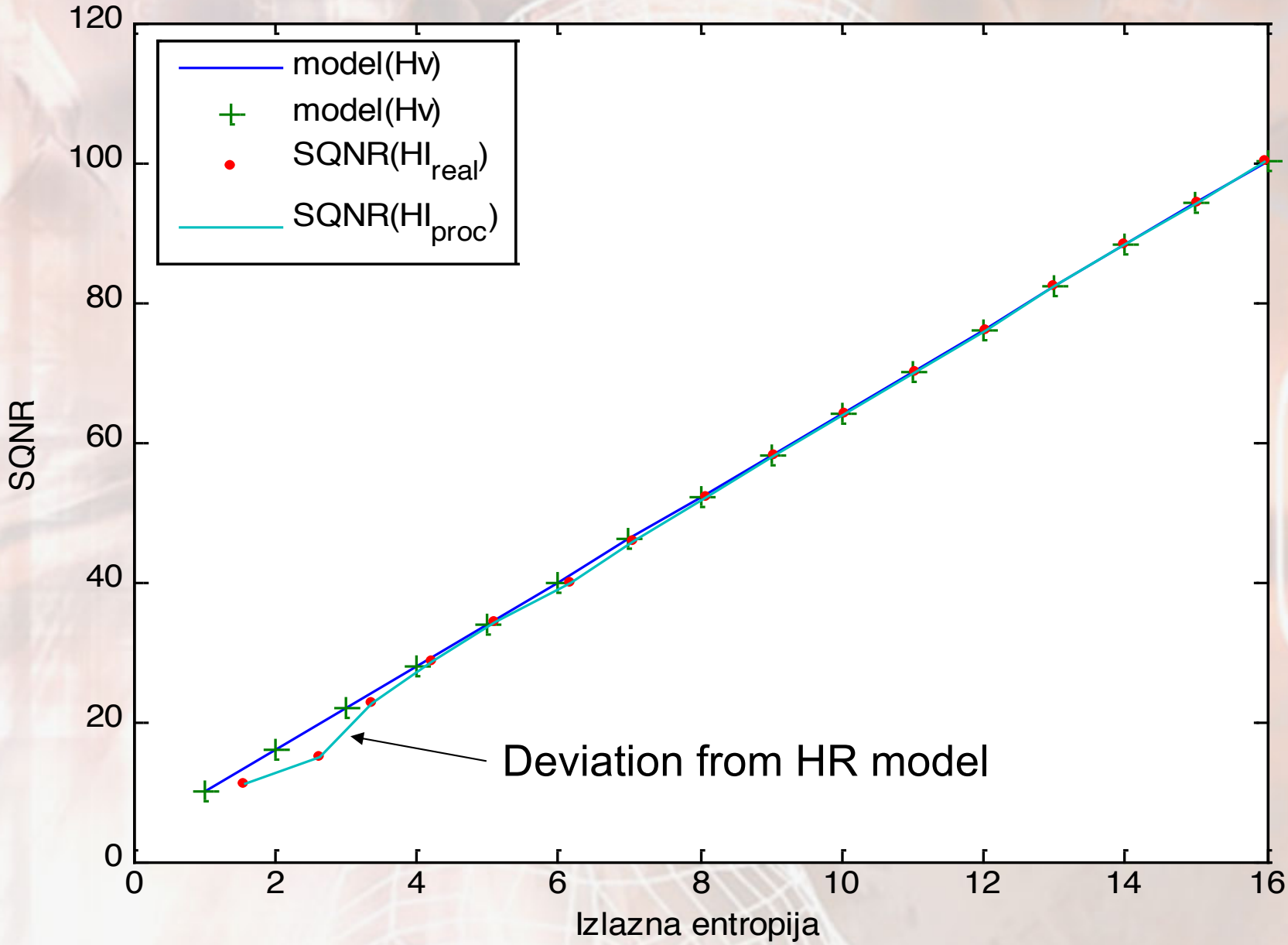
2. Gaussian noise – pdf of output symbols $H(I)=4$



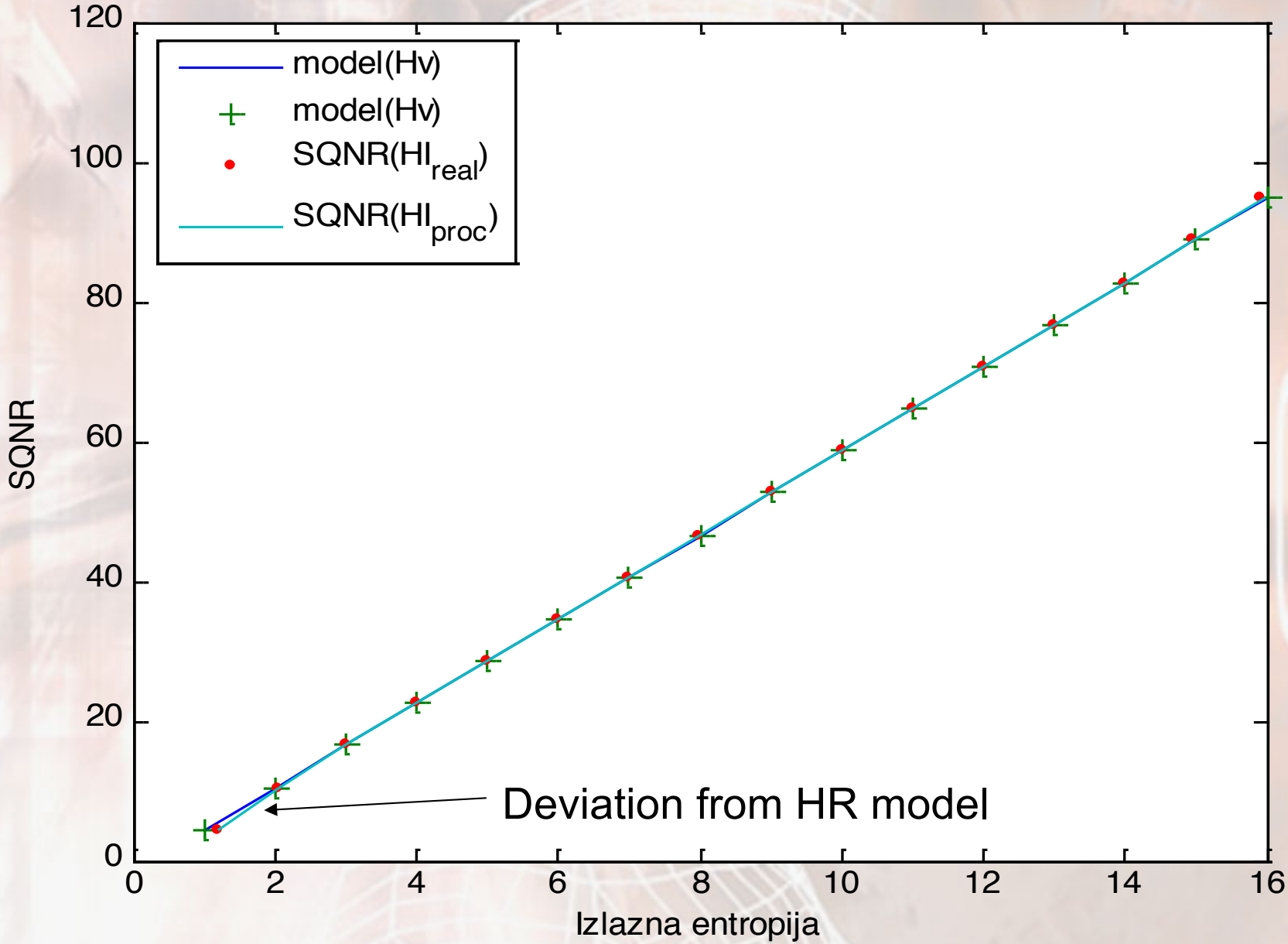
3. Triangle - pdf of output symbols $H(I)=4$



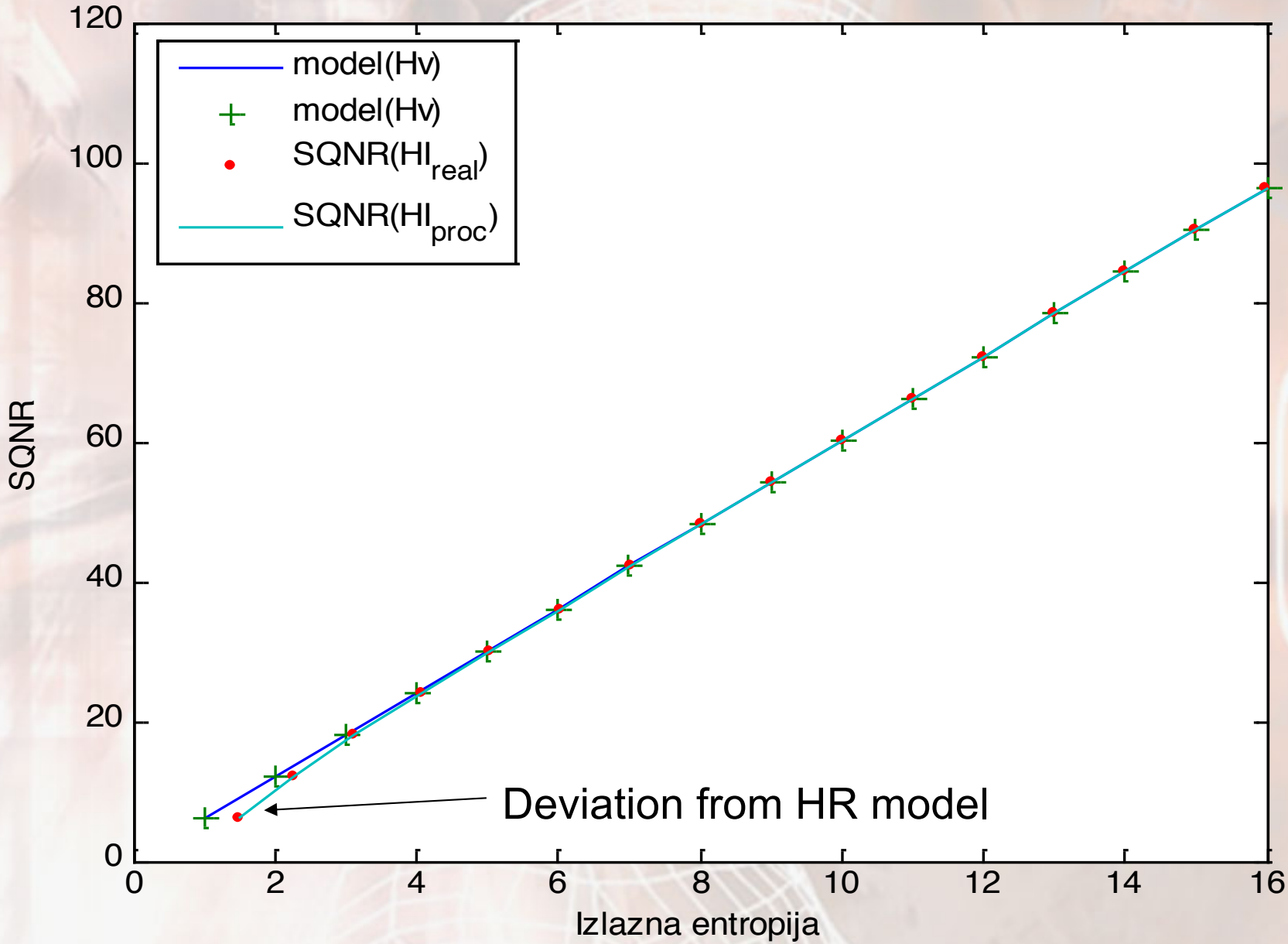
1. Sine – R-D curve $H(I)$=1..16



2. Gaussian noise – R-D curve $H(I)$=1..16



3. Triangle – R-D curve *H*(*I*)=1..16



What have we learned?

- High data rate quantization theory
- basic idea of HR theory
- the relation of entropy and distortion of the ECSQ quantizer
- differential entropy of the process
- signal/quantization noise ratio as a function of entropy
- determining quantizer step sizes
- determination of entropy for a given quality
- an example of a uniform probability density process
- an example of a random process of normal probability density
- upper bound of entropy
- example of HR analysis in Matlab