Codificação e Compressão

Aula 3 – Codificação e MATLAB

Prof. Adão Antonio de Souza Junior - IFSul

Nas aulas anteriores

1. Formas de compressão:
   a) Sem perdas: Cada bit do sinal é codificado e mensagem recuperada é idêntica ao texto original.

      Ex: ZIP, COMPRESS, etc.

   b) Com perdas: O sinal original em geral não é discreto e existem partes dele que são redundantes ou irrelevantes para o sentido geral.

      Ex: AAC, MP3 (Audio), JPEG2000, PNG (Imagem), MP4, WMV9, H264 (Vídeo), etc.

Nas aulas anteriores

1. Formas de compressão:
   a) Sem perdas: Cada bit do sinal é codificado e mensagem recuperada é idêntica ao texto original.

      Ex: ZIP, COMPRESS, etc.

   b) Com perdas: O sinal original em geral não é discreto e existem partes dele que são redundantes ou irrelevantes para o sentido geral.

      Ex: AAC, MP3 (Audio), JPEG2000, PNG (Imagem), MP4, WMV9, H264 (Vídeo), etc.

Na codificação com perdas a ultima etapa é sempre um processo de compressão sem perdas

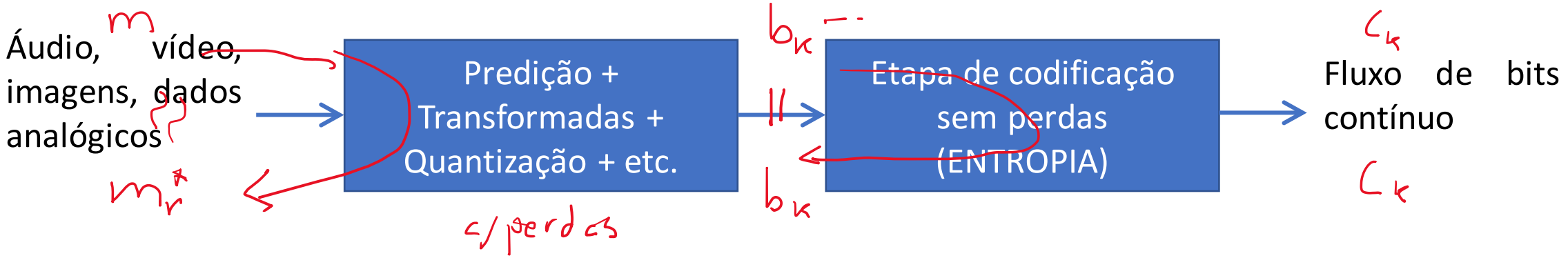
Nas aulas anteriores

1. Formas de compressão:
   a) Sem perdas: Cada bit do sinal é codificado e mensagem recuperada é idêntica ao texto original.

      Ex: ZIP, COMPRESS, etc.

   b) Com perdas: O sinal original em geral não é discreto e existem partes dele que são redundantes ou irrelevantes para o sentido geral.

      Ex: AAC, MP3 (Audio), JPEG2000, PNG (Imagem), MP4, WMV9, H264 (Vídeo), etc.



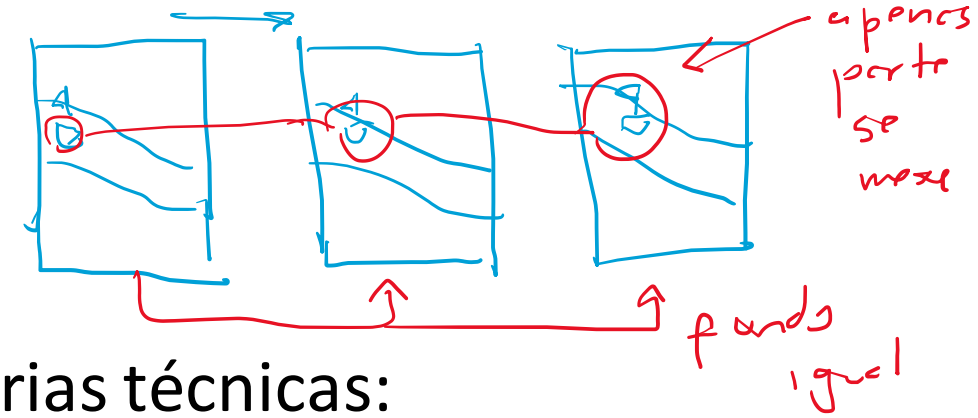
Nas aulas anteriores (2)

2. Na **codificação sem perdas** se usam várias técnicas:
   a) Códigos de tamanho variável: onde se busca atribuir um código de tamanho aproximado a quantidade de informação de cada símbolo.
   b) RLE (Run Length Encoding): onde se utilizam como símbolos corridas representadas por tuplas (c, s) onde c é o numero de repetições e s é o símbolo a ser repetido.
   c) Métodos de dicionário: onde não se define uma codificação a priori para o alfabeto e o dicionário é montado ao mesmo tempo em que se comprime.

Nas aulas anteriores (3)

2. Na **codificação com perdas** se usam várias técnicas:
   a) Quantização variável: o processo de transformação analógico digital pode ser feito de uma forma em que os símbolos são mais significativos e são necessários menos símbolos para reconstruir o sinal analógico. Há varias técnicas que se encontram aqui (quantização não uniforme, companding, quantização vetorial, quantização adaptativa/preditiva)
   b) Transformadas: nem sempre o sinal que se deseja adquirir é melhor representado no domínio em que é adquirido. Usando-se transformadas é possível localizar a dimensão na qual o sinal é mais esparso e codifica-lo nessa dimensão.

Nas aulas anteriores (4)



2. Na **<u>codificação com perdas</u>** se usam várias técnicas:
   c) Predição: em muitos casos existe alguma forma de redundância que pode ser usada para que se preveja os dados e codifique apenas os erros de forma adicional. Notem que a predição pode ser espacial, de canal paralelo, de escalabilidade, etc. Não apenas temporal, embora seja o caso mais comum.
   d) Considerações específicas ao meio: em alguns casos a informação relevante deve ser identificada pelos sentidos como audição e visão. Nesses casos, pode-se levar em consideração o que se sabe sobre o sistema auditório e visual humano a fim de reduzir as informações que não são perceptíveis.

      Ex: eliminar frequências próximas a frequências muito altas (mascaramento em frequência) ou a sons muito altos (mascaramento no tempo), usar uma resolução maior na informação de luminosidade do que na de cor em imagens, considerar que variações de cor são mais perceptíveis na vertical que na horizontal, etc...

Na primeira semana

- Entropia

  Para um conjunto de símbolos $\Xi = \{s_1, s_2, \dots s_M\}$, com probabilidades $\Pi(\Xi) = \{p_1, p_2, p_3 \dots p_M\}$, a entropia é dada por:

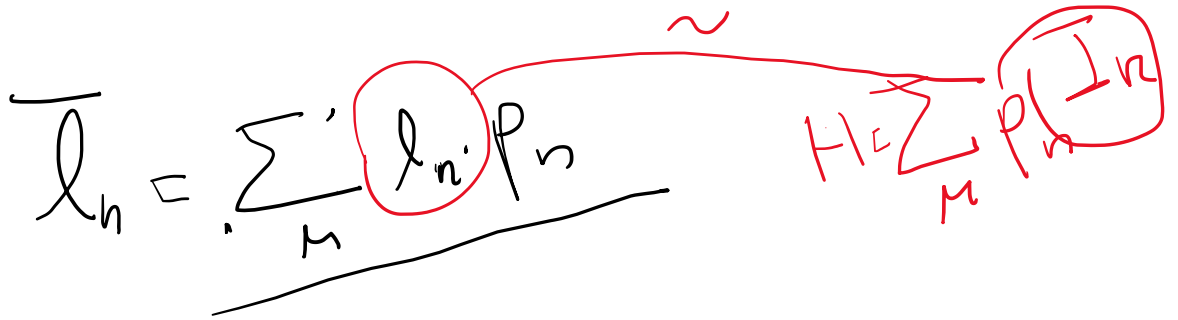
$$H(\Xi) = \sum_{n=1}^{M} p_n I_n = \sum_{n=1}^{M} p_n \log_2\left(\frac{1}{p_n}\right) = -\sum_{n=1}^{M} p_n \log_2 p_n =$$

- Teorema fundamental da codificação de fonte (TFCF)

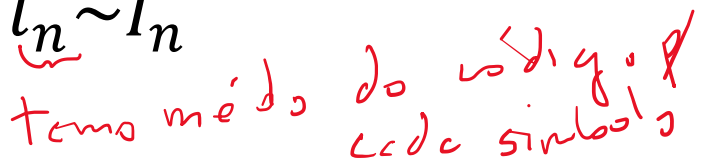
$$\bar{l} \geq H(\Xi)$$

Onde $\bar{l}$ é o tamanho médio de código (em bits) por simbolo para esse alfabeto.

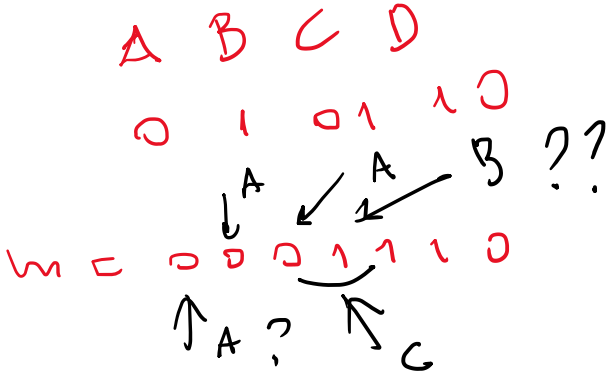
Na segunda semana



- A fim de se chegar a algo próximo do menor tamanho possível de código por símbolo (de acordo com o TFCF) idealmente o tamanho de cada código deveria ser aproximadamente o tamanho da quantidade de informação que esse código agrega, ou seja $l_n \sim I_n$



- É necessário construir códigos de tamanho variável que possam ser decodificados de forma instantânea e biunivoca (unicamente decodificavel, ou UD):
  - Códigos Elias
  - Códigos Rice
  - Códigos de Huffman



Kraft-McMillan

Aula 3 – Codificação e MATLAB

Prof. Adão Antonio de Souza Junior - IFSul

Desigualdade de Kraft-McMillan

- Dados M códigos de um alfabeto com comprimentos l_n, se:

$$\sum_{n=1}^{M} 2^{-l_n} \leq 1, \exists\ UD$$

- Ou seja, existem códigos que com essa distribuição de tamanhos que são UD.
- De mesmo modo existem códigos instantâneos, mas isso não garante que esse especificamente seja instantâneo, apenas que existe um desse tamanho que é.

Ex: (0, 01, 11) → UD não instantâneo

(0, 10, 11) → UD instantâneo

- Não deve ser vista como uma garantia de que é decodificável e sim como uma forma de descartar os que não são.

Desigualdade de Kraft-McMillan (2)

Aplicações

1) O código {10, 110, 1110, 1111} pode ser UD?

l = {2,3,4,4}=1/2+1/3+1/4+1/4= 1.333 >1 NÃO É UD

2) O código onde $l_i = \log_2 M$ pode ser UD?
3) O código onde $l_i = i$ pode ser UD?
4) O código onde $l_i = 2^i$ pode ser UD?

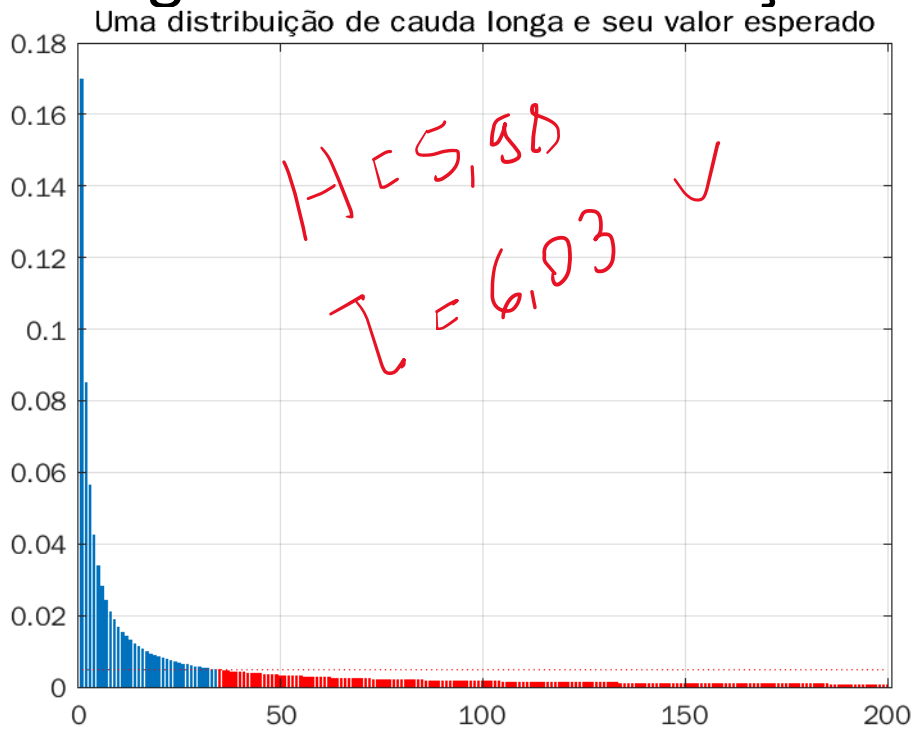
Códigos de Escape

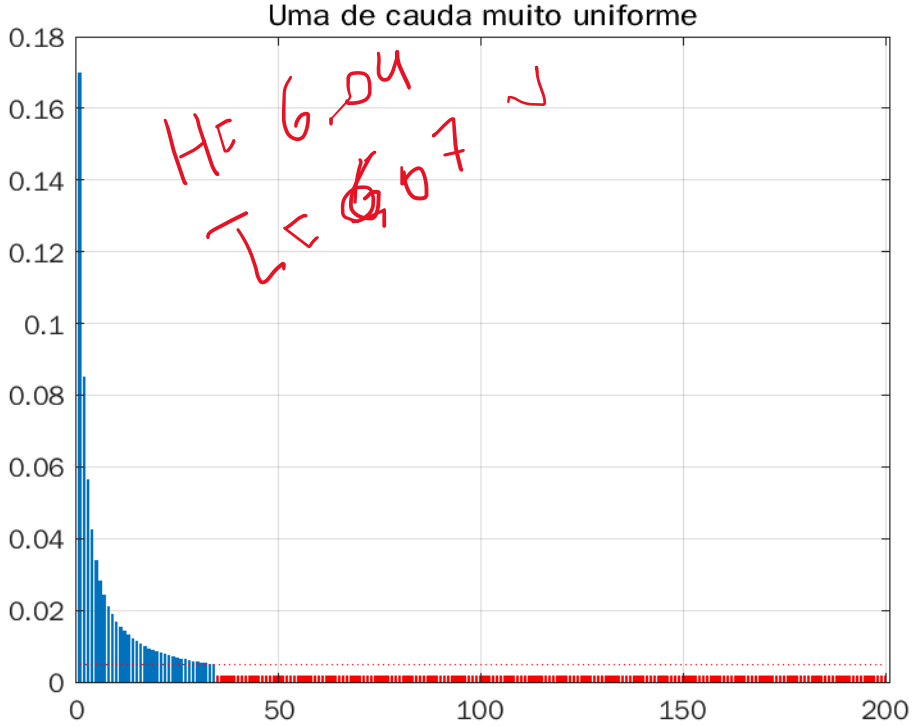
Aula 3 – Codificação e MATLAB

Prof. Adão Antonio de Souza Junior - IFSul

Códigos de escape

- Imaginem uma distribuição de probabilidades como as vistas abaixo





- Em que ponto seria interessante estabelecer um escape?

Considerações sobre o escape

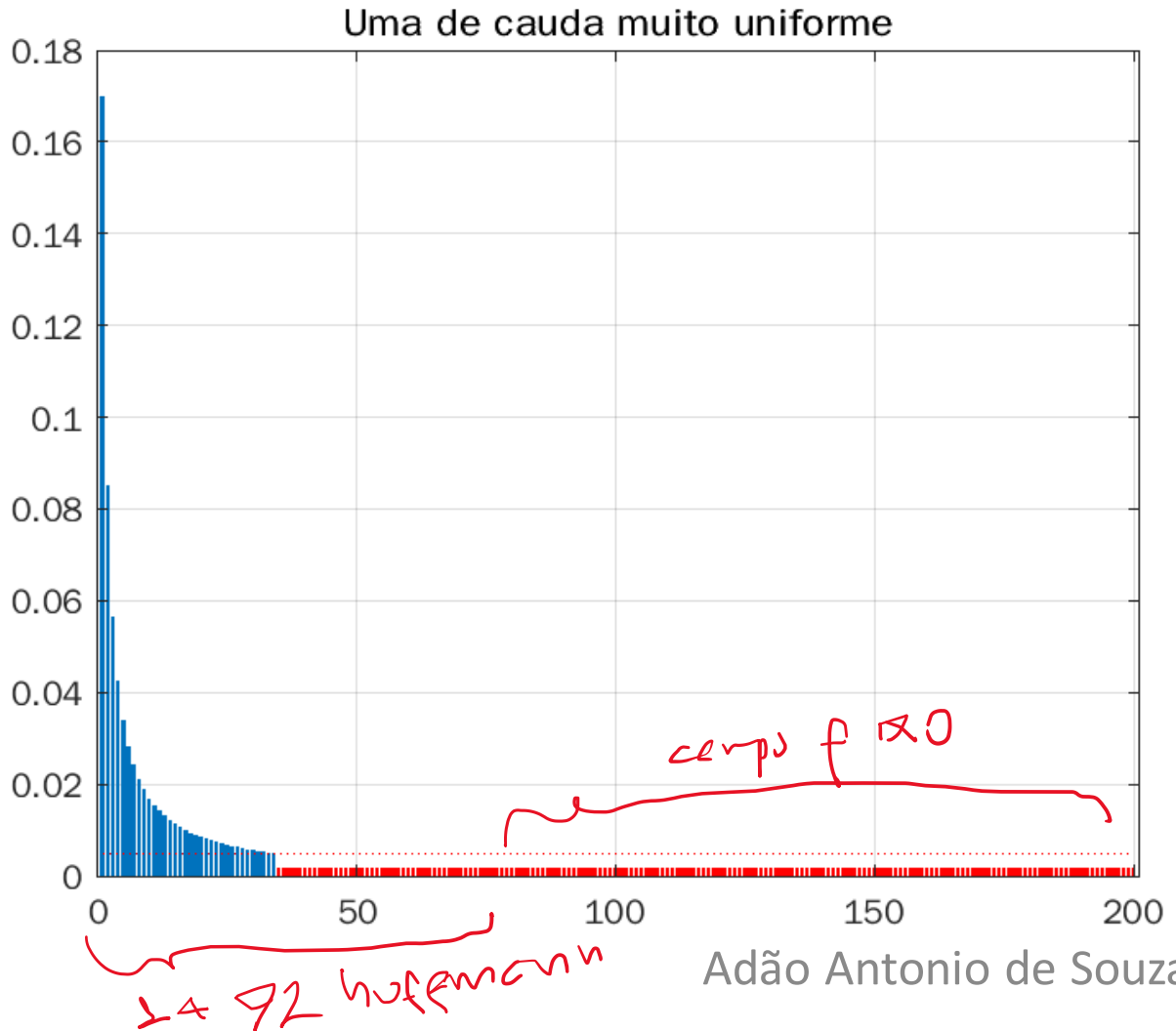
- Notem que, a principio Huffman é menos bem sucedido para códigos uniformes, assim, quanto mais uniforme a cauda pior será o tamanho médio obtido por Huffman
- No caso com a entropia para os gráficos e o tamanho médio obtido por Huffman é similar em desempenho:

H = 5.98, L = 6.03 (esquerda)

H = 6.04, L = 6.07 (direita)

- A questão é que na cauda longa temos tipicamente simbolos que ocorrem uma vez no conjunto de amostras.
- Do mesmo modo podem haver elementos do alfabeto que não ocorrem no conjunto de amostras.
- Nos dois casos devemos entender que as probabilidades usadas são uma aproximação e escolher uma solução que atenda todo o conjunto.
- A solução para isso é criar um caracter de ESCAPE. Vamos ao exemplo.

Exemplo de Escape



200 símbolos estão no alfabeto
Apenas 34 símbolos tem probabilidades diferentes
Todos os demais tem uma ocorrência e probabilidade igual (166)

Decido fazer um campo fixo para os 128 simbolos de menor probabilidade.

Construo um alfabeto com os 72 de maior probabilidade e acrecento mais um simbolo, cuja probabilidade é igual a probabilidade TOTAL de se cair em um dos 128 que foram removidos.

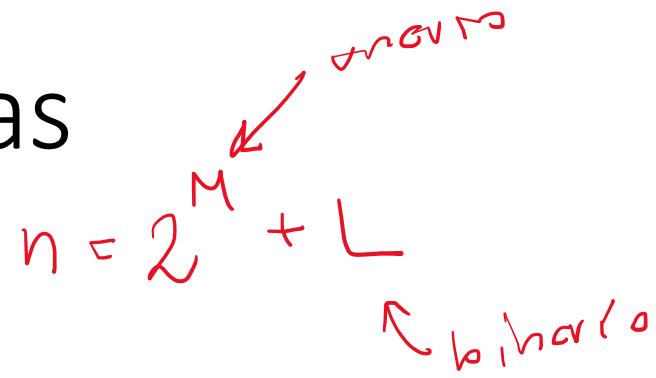
Lesc = 1.62, Lcesc= 4.46, Ltotal → 6.07

Funções de VLC

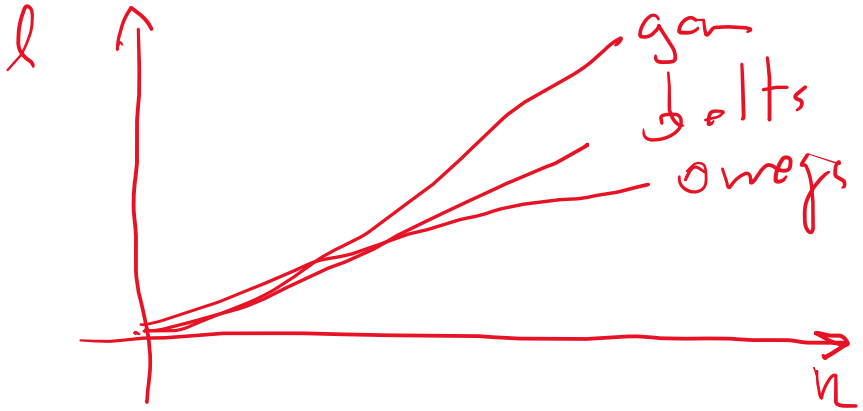
Aula 3 – Codificação e MATLAB

Prof. Adão Antonio de Souza Junior - IFSul

Nos exercícios são pedidas algumas atividades



- A partir de um código Elias Gamma
  - Onde estariam os códigos equivalentes a parte unaria e binaria?
  - Como se fariam funções com isso?
- Como se tornaria o código em uma função Elias_Gamma?
- A partir dai, o que precisamos mudar para gerar um Código para Elias_Delta?
- E se for para mudar para Elias Omega?



```
% Implementa o código Elias Gamma
clear all; close all; clc;

N = 25;
i = 1:N; % Gera uma sequencia de números para converter

% Efetua a codificação por Elias Gamma
for i=1:N
    % Passo 1: calcula o número em binário
    beta = dec2bin(i);
    % Passo 2: Seu comprimento é um número M
    M = length(beta);
    % Passo 3: Gera um prefixo formado por M-1 zeros seguidos de 1
    if (M-1) > 0
        prefixo = repmat('0',1,M-1);
    end;
    % Passo 4: Anexa prefixo ao numero (uso celulas pq sao strings de tamanho
    % diferente em cada indice
    if (M-1) > 0
        numero{i} = strcat(prefixo, beta);
    else
        Numero{i} = beta;
    end;
end;

% Exibe os primeiros L elementos do conjunto de códigos
L = 25;
for i=1:L
    codigo = numero(i);
    disp(sprintf('n:%d -> %s',i, codigo{:}));
end;
```

Operando símbolos no MATLAB

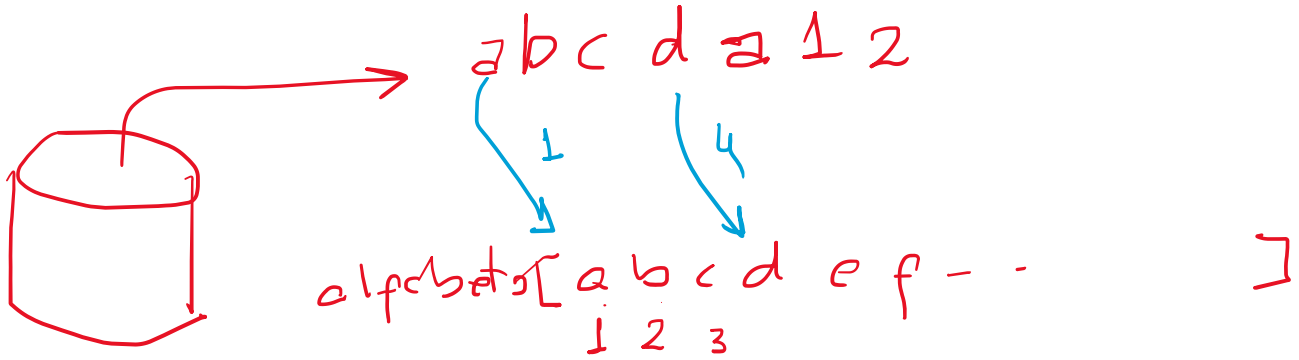
Aula 3 – Codificação e MATLAB

Prof. Adão Antonio de Souza Junior - IFSul

Alguns pontos importante

1. Lendo e salvando arquivos de dados
    1. Representação dos dados lidos
2. Lendo e salvando arquivos de voz
3. Lendo e salvando arquivos de imagem
4. Criando símbolos a partir de dados
    1. Codificação de imagem por blocos
    2. Codificação de imagem por bitplanes
    3. Run-Lenght Encoding

Estrutura interna



- Os dados lidos são salvos em uma estrutura (cell array ou array)
- **Os índices dessas estruturas** passam a ser usados para **representar esses dados** na fase de compressão
- Ao descomprimir as palavras de tamanho variável se obtem os índices das estruturas.
- **Esses índices obtidos**, junto com a tabela local de decodificação são usados para reconstruir a mensagem.

Estrutura interna

- Os dados lidos são salvos em uma estrutura (cell array ou array)
- **Os índices dessas estruturas** passam a ser usados para **representar esses dados** na fase de compressão
- Ao descomprimir as palavras de tamanho variável se obtem os índices das estruturas.
- **Esses índices obtidos**, junto com a tabela local de decodificação são usados para reconstruir a mensagem.

Ou seja, não importa o arquivo de entrada, sempre estamos codificando índices na etapa de entropia

Exemplo 1 – Arquivo Texto e audio

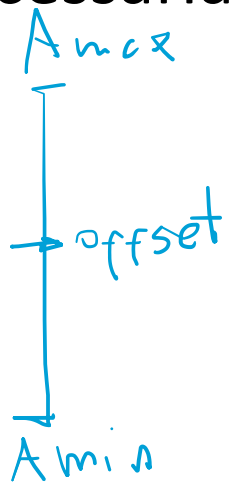
- O mesmo arquivo MATLAB mostra o caso de codificação para arquivo texto e para áudio (Exemplo1_Texto_e_Som)
- Para o exemplo de texto usaremos um texto com alfabeto restrito chamado exemplo1.txt
- Para o exemplo de áudio usaremos um arquivo de áudio WAV chamado exemplo1.wav

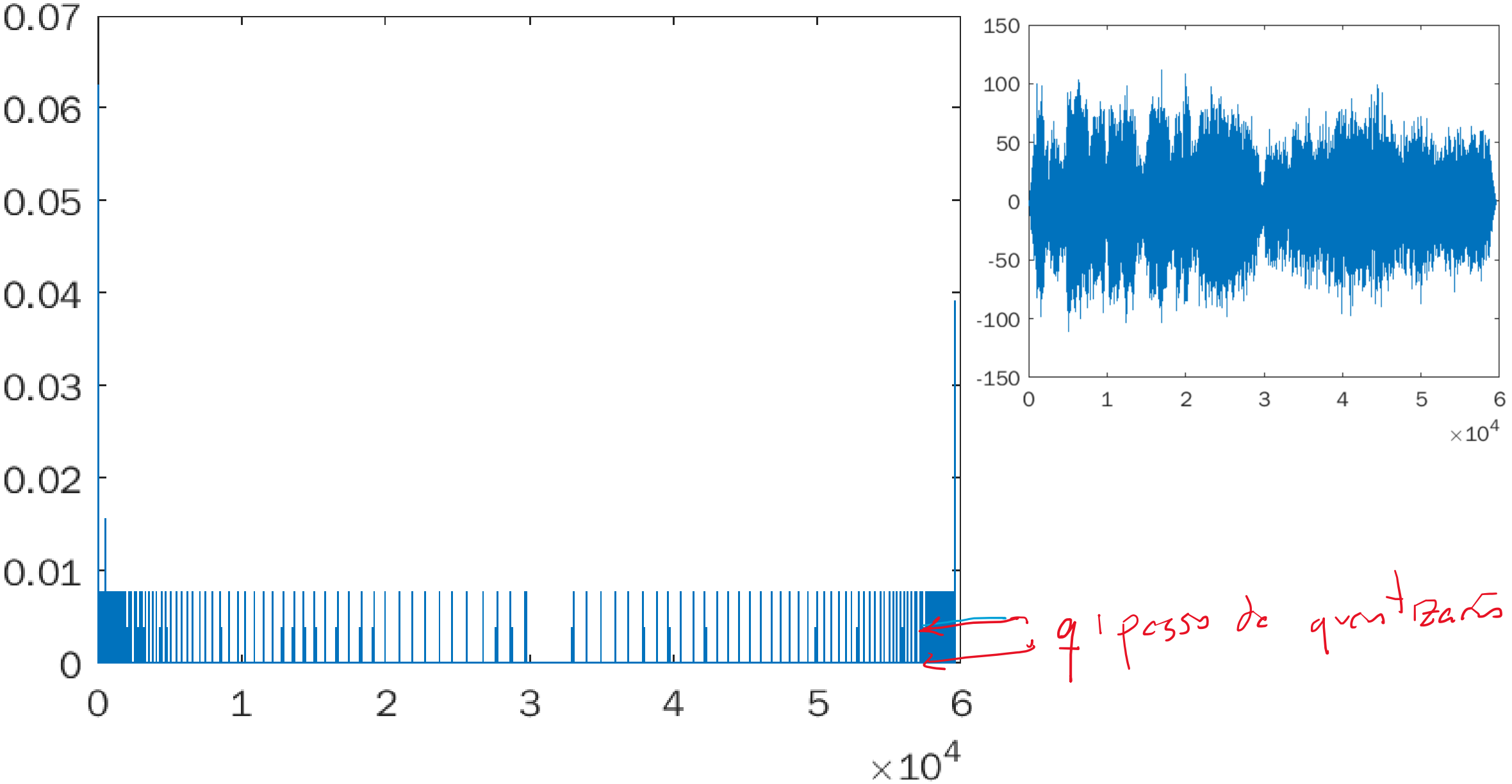
```
mensagem_em_simbolos =
textread('exemplo1.txt', '%c');


L = length(mensagem_em_simbolos);


% Note que eu sei todos os caracteres possíveis de
serem usados no texto.
% Nesse caso são apenas os caracteres ASCII entre
'0' e '~';


Tabela =
'0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUV
WXYZ[\]^-`abcdefghijklmnopqrstuvwxyz{|}~ ';
```

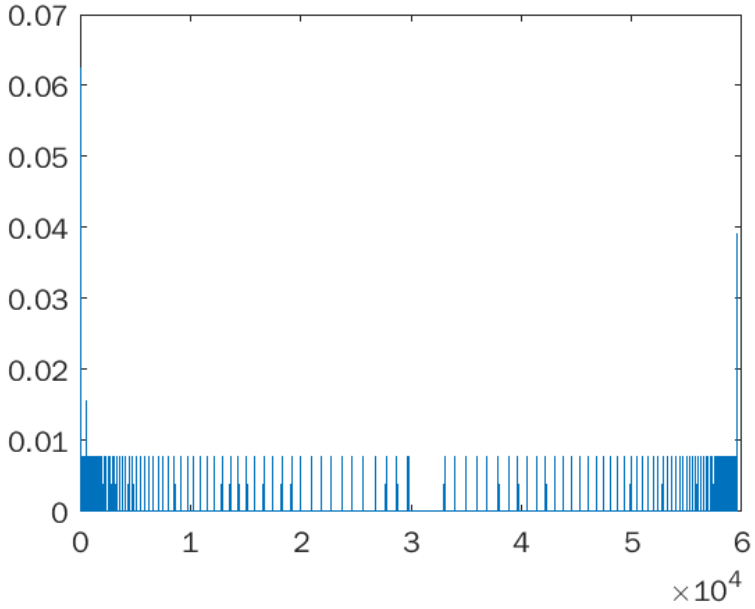
```
% Notem que o índice de cada simbolo na tabela
corresponde ao simbolo assim
% 1 corresponde ao simbolo '0', 23 corresponde
ao simbolo F e assim por
% Diante

% A codificação é o processo pelo qual todos os
simbolos serão substituidos
% por referencias a tabela

N = length(mensagem_em_simbolos)
for i=1:N % percorre to texto e substitui os
simbolos pelos seus códigos
texto(i,:)=min(find(Tabela==mensagem_em_sim
bolos(i)));
end;


% Note-se que, a fim de reconstruir o texto basta
ter o vetor de indices


mensagem_reconstruida  = Tabela(texto);
```

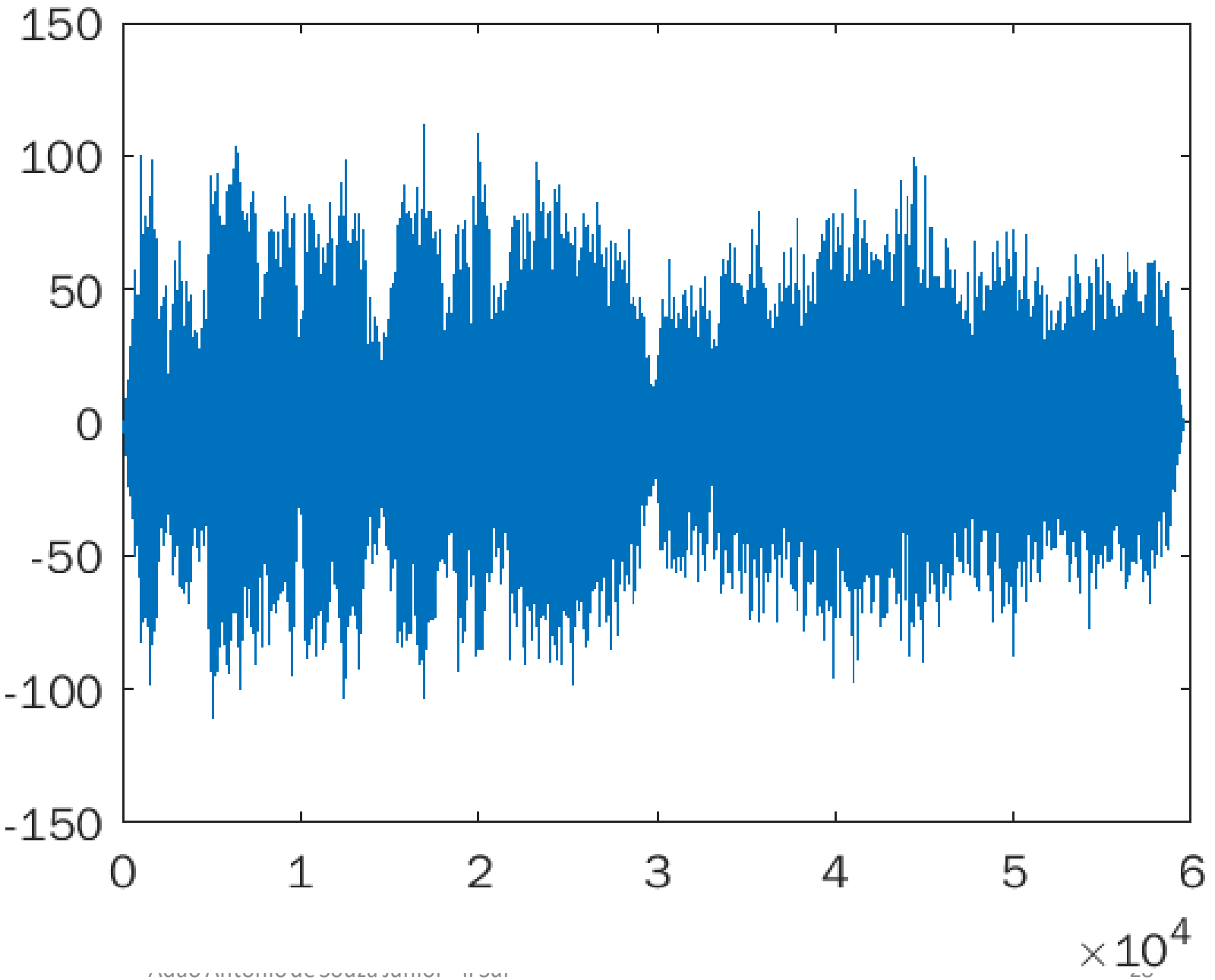
Arquivo de audio

- No caso do áudio o problema é diferente pois os dados já são numéricos, mas não correspondem a números inteiros positivos.
- Além disso o áudio possui informações adicionais necessárias para sua decodificação:
- Fs: Sua taxa de aquisição (e reprodução)
- DR ou Amplitude (faixa de valores e/ou amplitude)
- Offset ou Centro (centro da faixa de valores.

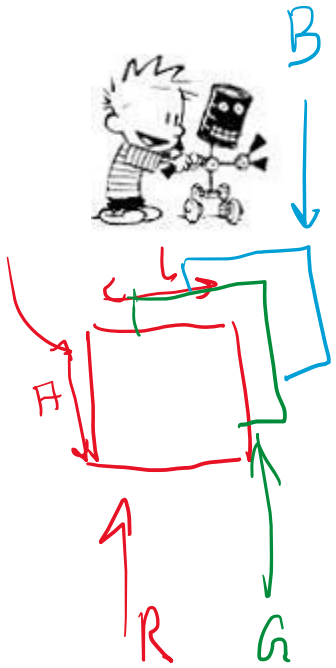


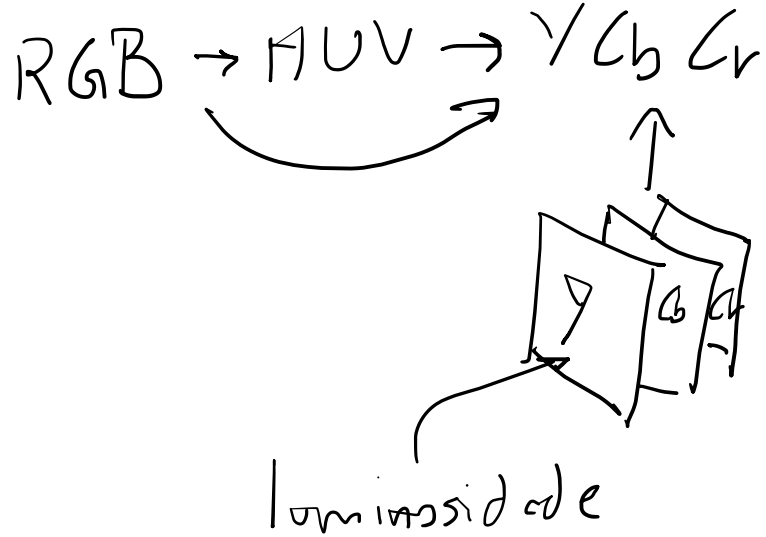

0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
0 1 2 3 4 5 6
×10^4
150
100
50
0
-50
-100
-150
0 1 2 3 4 5 6
×10^4
q i passo de quantização


0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
0
1
2
3
4
5
6
×10^4


150
100
50
0
-50
-100
-150
0
1
2
3
4
5
6
×10^4

Leitura de imagem

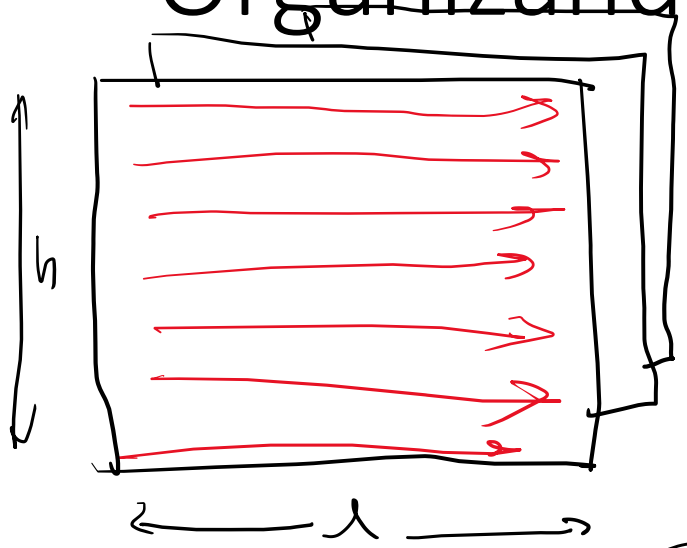


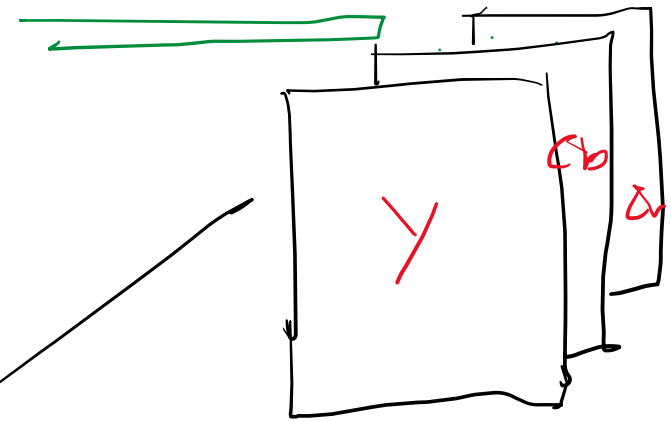


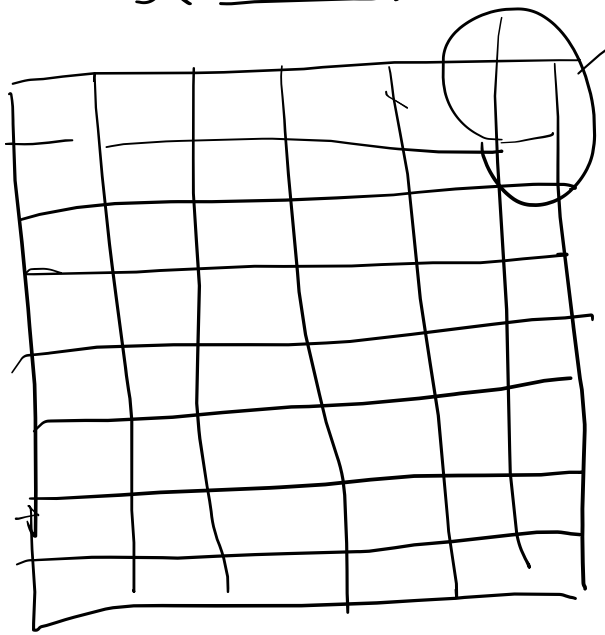
Leitura de imagem

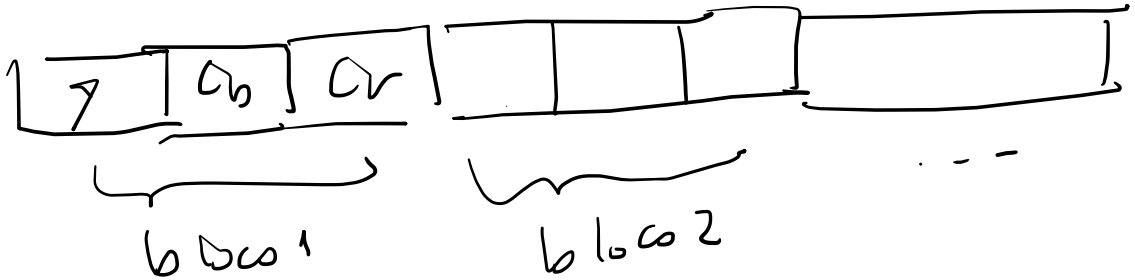
- A imagem é armazenada em conjuntos de matrizes com as dimensões dadas pela largura x altura x numero de componentes
- Quando a imagem é colorida ela é composta por três imagens
  - RGB : Uma para cada componente
  - YCbCr : Mais usado em compressão pois a primeira tem a informação de luminância (intensidade da cor) e as duas demais da cor em si.
- O MATLAB permite facilmente tornar matrizes em vetores a questão é da organização que se deseja e de se manter esse processo reversível

Organizando imagem em blocos





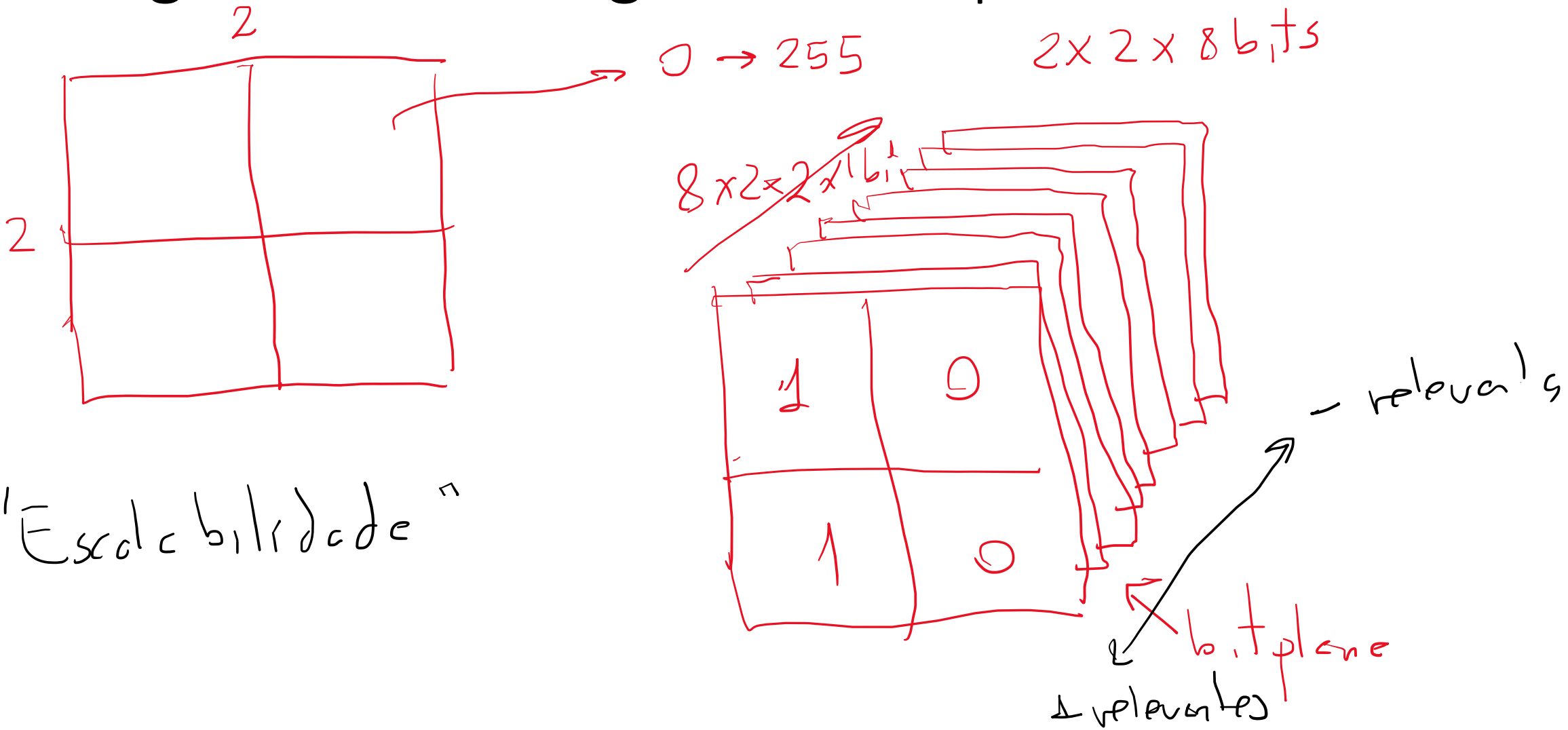




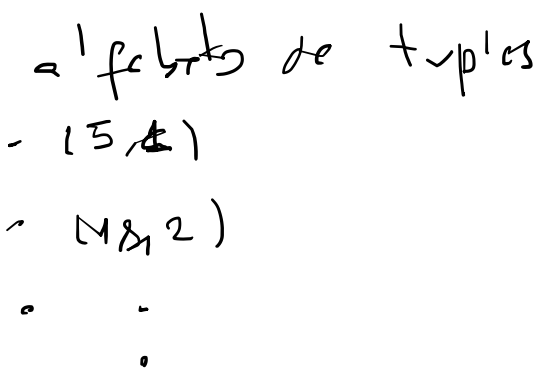
16x16 8x8 4x4 — 4x8 8x4 — padrões de fragmentação

MPEG2
↳ 4:1:1
4:2:2
4:4:4

Organizando imagem em bitplanes



Corridas (RLE)



- Quando se tem grande repetição de sequencias de algum tipo de elemento se introduz o conceito de tupla
- Uma tupla é formada pelo índice de um símbolo e o numero de vezes que esse índice repete:

Ex: (1,5) – repete o símbolo 1 cinco vezes em sequencia

(2,1) – coloca o símbolo 2 uma única vez

- Quando se trabalha com RLE ele é uma etapa antes de VLC, ou seja, o alfabeto representado agora irá abordar tuplas que indicam corridas.

Atividades

- Partindo do código feito em aula, resolva as tarefas apresentadas (codificação por huffmann do texto da semana 1).

- Partindo do feito em aula, crie um arquivo que representa imagens na configuração 4:4:4 YCbCr.
  - Partindo dessa configuração gere um stream de bits codificado sabendo que se usam blocos de 8x8 e que os dados são sempre alternando (YCbCrYCbCr…)

- Partindo do bitplane mais significativo criem um codigo que codifica em tuplas de 1 e 0. Façam tambem a decodificação.
- Semana que vem falaremos sobre o primeiro trabalho de avaliação grande.