

1º Semestre 16/17 SI

Comunicação e Processamento de Sinais

**Relatório do 2º Trabalho Prático**

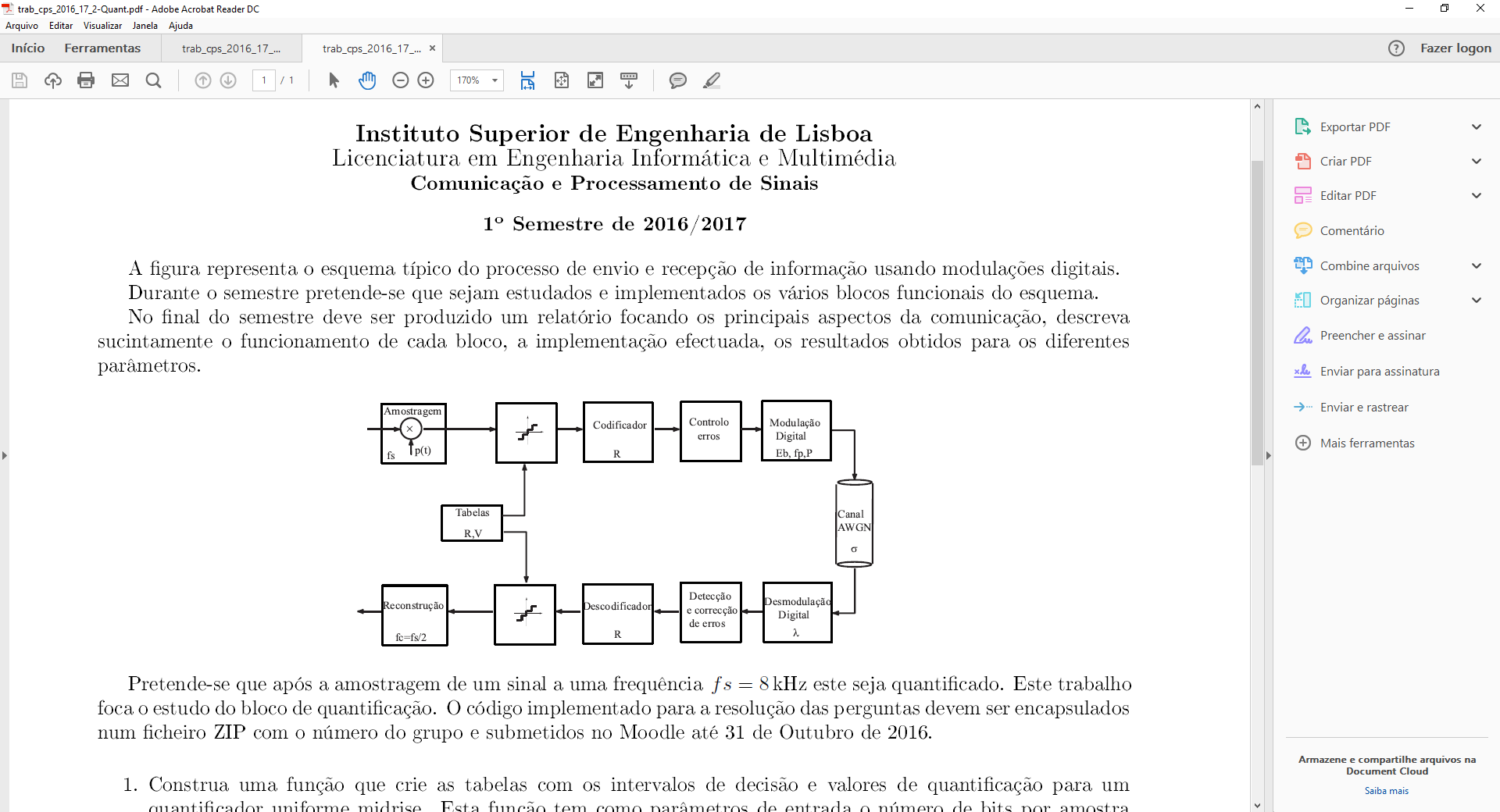
Eng. Pedro Fazenda

**Trabalho realizado por:**

Diogo Fernandes nº39205

Rui Santos nº39286

**Introdução**



Este segundo trabalho foca-se nos blocos de Quantificação e Codificação e suas operações inversas, iremos utilizar código desenvolvido no primeiro trabalho para executar o código que pretendemos fazer neste trabalho.

Exercícios

Primeira Parte - Quantificação

1. Construa uma função que crie tabelas com os intervalos de decisão e valores de quantificação para um quantificador uniforme midrise. Esta função tem como parâmetros de entrada o numero de bits por amostra (R) e o valor máximo a quantificar (Vmax). Como parâmetros de saída tem dois Numpy arrays com valores e quantificação e os intervalos de decisão.

def createTable(R, Vmax):

valoresDecisao = np.arange(-Vmax , Vmax+((2.\*Vmax)/(2.\*\*R)) , (2.\*Vmax)/(2.\*\*R))

niveisQuantificacao = np.arange(-Vmax+(Vmax/(2.\*\*R)) , Vmax , (2.\*Vmax)/(2.\*\*R))

return valoresDecisao, niveisQuantificacao

**2.**Construa uma função que dado um Numpy array com as amplitudes de um sinal amostrado, retorne um Numpy array do mesmo tamanho com o sinal quantificado. Esta função tem com parâmetros de entrada o sinal (x), as tabelas com os valores de quantificação e intervalos de decisão. Como parâmetros de saída retorna o Numpy array com sinal de saída (xq) e um Numpy array com o índice dos valores de quantificação usados (vq).

def quantificacao(sinalAmostrado, NQ, VD):

sinalQuantificado = np.zeros(len(sinalAmostrado))

indiceQuant = np.zeros(len(sinalAmostrado))

maximo = np.amax(abs(sinalAmostrado))

for a in range(len(sinalAmostrado)):

indice = VD > sinalAmostrado[a]

if(sinalAmostrado[a]==maximo):

indiceQuant[a] = len(NQ)-1

sinalQuantificado[a] = NQ[len(NQ)-1]

else:

indiceTrue = np.nonzero(indice)

aux = int(indiceTrue[0][0]-1)

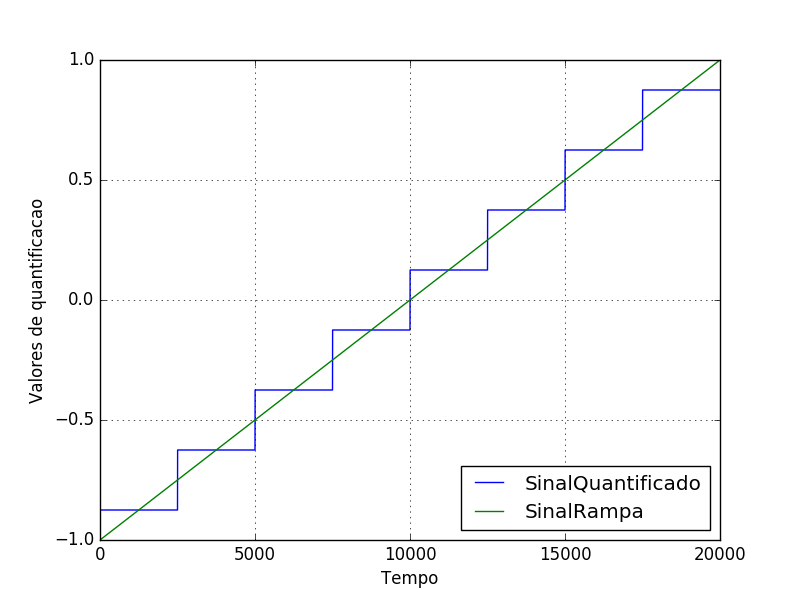
indiceQuant[a] = aux

sinalQuantificado[a] = NQ[aux]

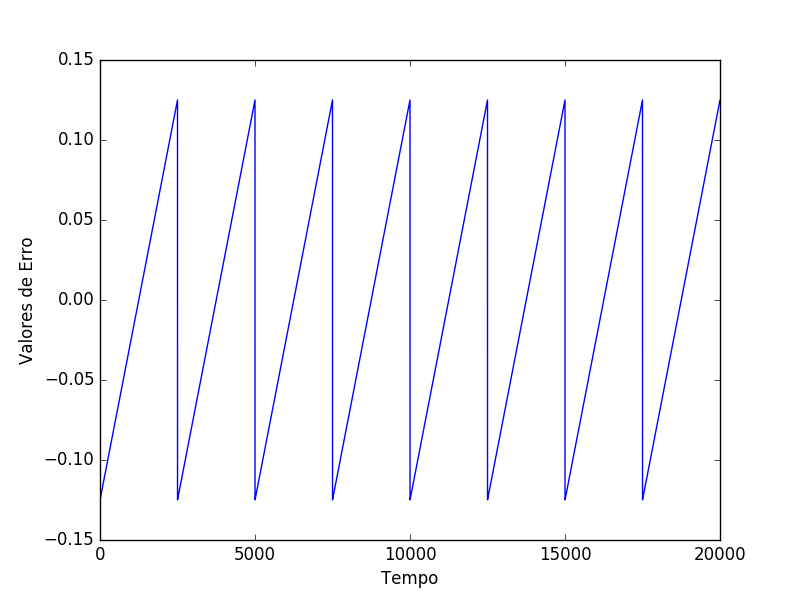
return sinalQuantificado , indiceQuant.astype('int16')

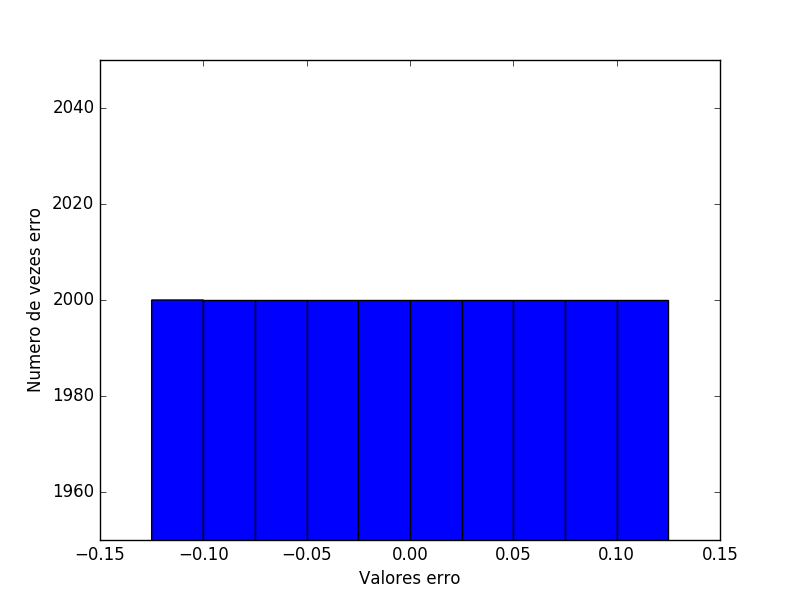
**3.**Com as funções criadas nos pontos anteriores, e assumindo que o numero de bits é R=3:

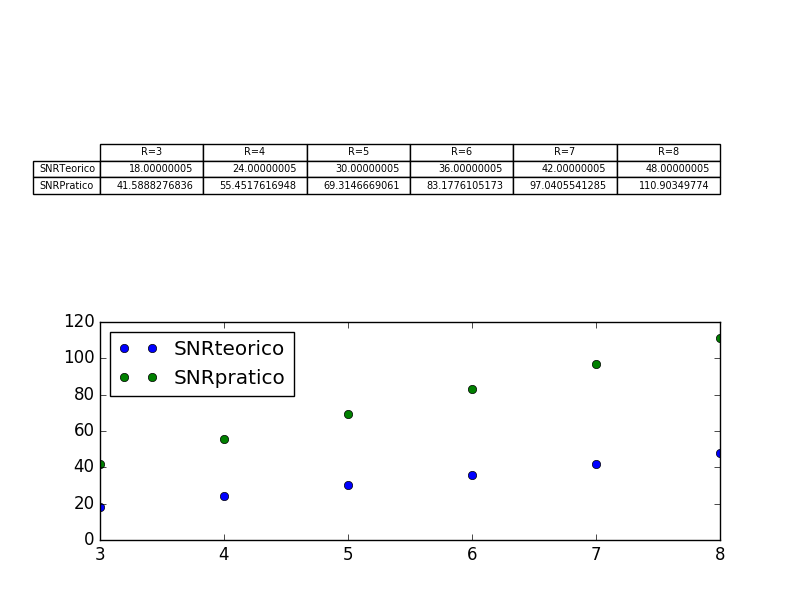
**a)** quantifique um sinal de rampa. Represente as amostras do sinal original e do sinal quantificado em função do tempo.



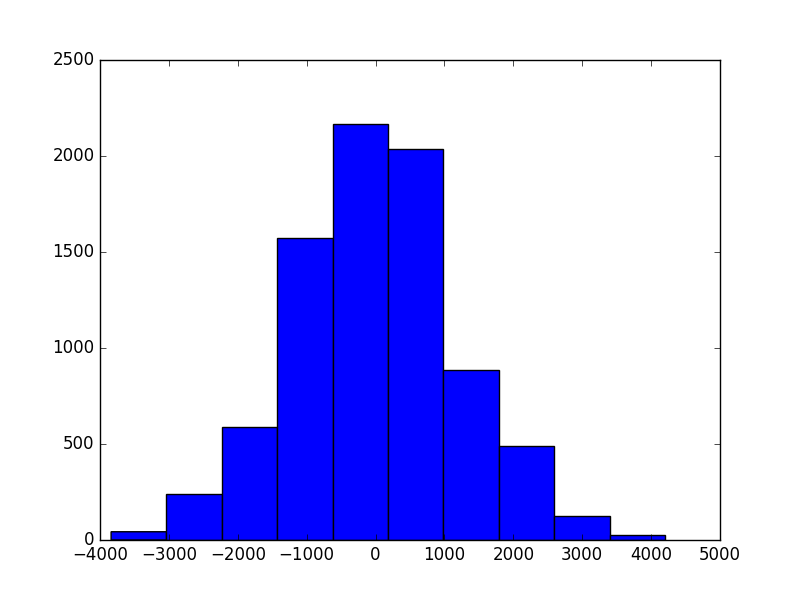
**b)** represente o erro de quantificação em função do tempo e o seu histograma.

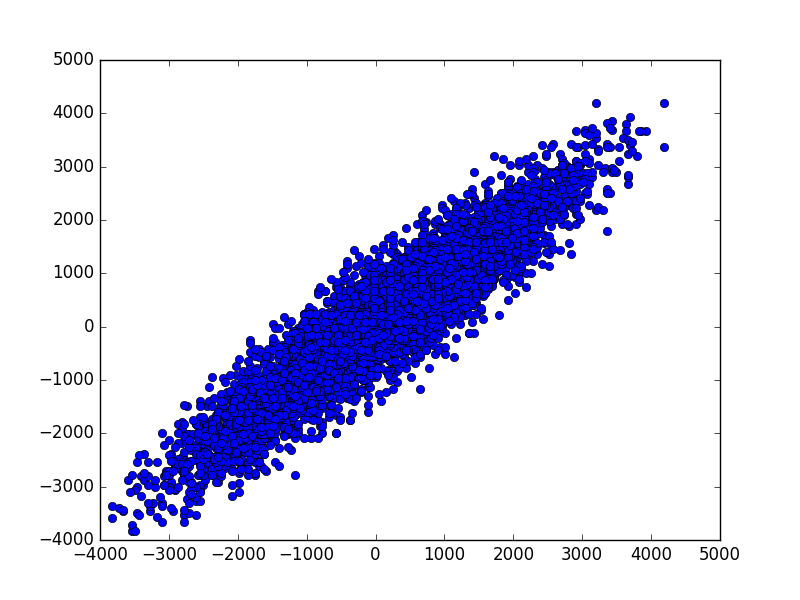


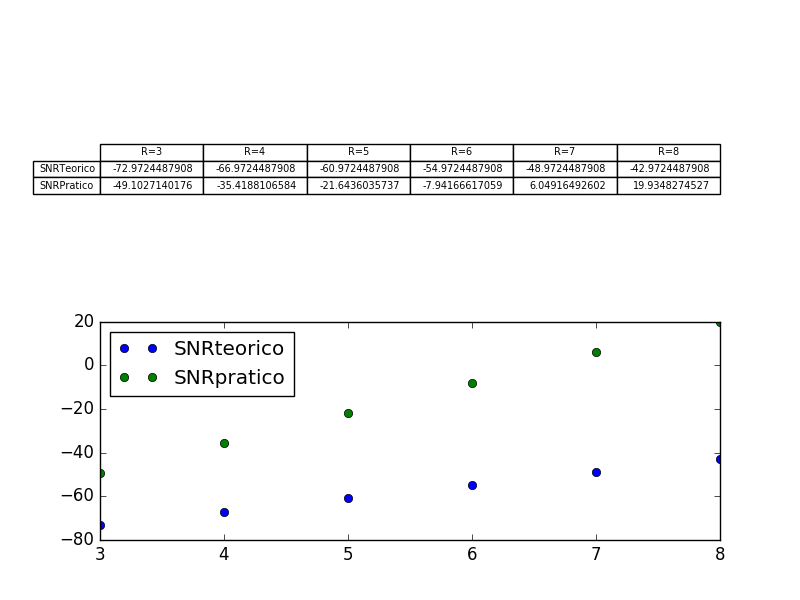


**c)** para cada valor de R={3,4,5,6,7,8} meça a SNR e compara com o valor teórico. Construa um gráfico com os valores de SNR teórica e medida em função do numero de bits R.

**4.**Faça a leitura do ficheiro de áudio previamente gravado para um Numpy array.

**a)** represente o histograma no sinal de áudio

**b)** represente cada amostra do sinal em função da amostra anterior. Comente o gráfico.

**c)** para cada valor de R= {3,4,5,6,7,8} quantifique o sinal e meça a SNR. Ouça o sinal quantificado para os diferentes valores de R. Construa um gráfico com os valores de SNR em função do numero de bits R. Analise o gráfico e tire conclusões.

Segunda Parte – Codificação

**1.** Construa uma função que dado um Numpy array (com dimensão N) com valores inteiros, retorne um Numpy array (com dimenso N.Rx1) com os mesmos valores convertidos para binário. Esta função ainda deve receber como parâmetros de entrada o numero de bits (R) a usar na conversão de cada inteiro. Construa também a função que dado um Numpy array com valores os bits, faça a conversão para inteiros (considerando R bits).

def codificaSinal(IQ, R):

sinalCodificado = np.zeros(len(IQ)\*R)

count=0

binario = '{0:0'+str(R)+'b}'

for i in range(len(IQ)):

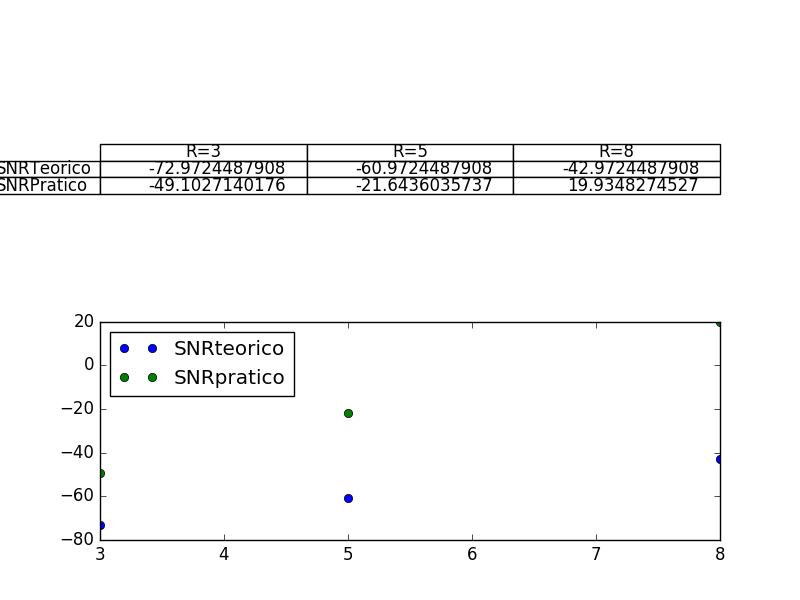
aux = binario.format(IQ[i])

for x in range(0,R):

sinalCodificado[count+x]=int(aux[x])

count+=R

return sinalCodificado.astype('int16')

**2.** Com base nas funções já implementadas, faça a codificação e descodificação PCM uniforme do sinal de áudio previamente gravado. Verifique a SNR obtida e ouça o sinal descodificado. Use par ao efeito R = 3,5,8

**3.** considere o sinal , amostrado a uma frequência de 8000 hz

**a)** usando um quantificador midrise com R=3, faça a codificação PCM das 5 primeiras amostras do sinal.

|  |
| --- |
| Decisão |
| 1 |
| 0.75 |
| 0.5 |
| 0.25 |
| 0 |
| -0.25 |
| -0.5 |
| -0.75 |
| -1 |

|  |  |
| --- | --- |
| Quantificação | Codificação |
| 0.875 | 000 |
| 0.625 | 001 |
| 0.375 | 010 |
| 0.125 | 011 |
| 0.125 | 100 |
| 0.375 | 101 |
| 0.625 | 110 |
| 0.875 | 111 |

L = 23 = 8

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nº amostra | Expressão | Resultado | Quantificado | Codificado |
| 1ª |  | 1 | 0.875 | 000 |
| 2ª |  | 0 | 0.125 | 011 |
| 3ª |  | -1 | -0.875 | 111 |
| 4ª |  | 0 | 0.125 | 011 |
| 5ª |  | 1 | 0.875 | 000 |

**b)** usando um quantificador não uniforme lei A (R=3), faça a codificação das primeiras 5 amostras do sinal.

A lei A tem um valor normalizado de 87.56

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nº amostra | Expressão | Resultado | Quantificado | Codificado |
| 1ª |  | 1 | 0.875 | 000 |
| 2ª |  | 0 | 0.125 | 011 |
| 3ª |  | 1 | 0.875 | 000 |
| 4ª |  | 0 | 0.125 | 011 |
| 5ª |  | 1 | 0.875 | 000 |

**c)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| n | M[n] | Mp[n] | E[n] | Eq[n] | Mq[n] | Cod. |
| 0 | 1 | 0 | 1 | 0.875 | 1 | 000 |
| 1 | 0 | 1 | -1 | -0.875 | -0.875 | 111 |
| 2 | 1 | 0 | 1 | 0.875 | 1 | 000 |
| 3 | 0 | 1 | -1 | -0.875 | -0.875 | 111 |
| 4 | 1 | 0 | 1 | 0.875 | 1 | 000 |