Performance of optimal codes at finite length

Joseph J. Boutros (ENST) and Kalpesh Patel (Nexans)

November 10, 2006

1 Channel model and notations

We consider the simple additive white Gaussian noise (AWGN) channel. In his 1959 paper [1], Claude E. Shannon showed that an optimal code is built by uniformly placing codewords on an $n$-dimensional sphere. An upper and a lower bound for the word error rate performance $P_{ew}$ of such a spherical code have been established by Shannon on an AWGN channel [1] for finite $n$. The main code parameters are its length $n$ and its information rate $R$. The length $n$ is the number of real dimensions. The information rate $R$ is expressed in bits per real dimension. The spherical code is an ensemble of $2^{nR}$ points uniformly placed on a sphere in $\mathbb{R}^n$.

A quick review of Shannon results and its generalization to a Rayleigh fading channel can be found in [7]. For $n \geq 100$, the upper and lower bounds of $P_{ew}$ are superimposed. Hence, an accurate approximation for $P_{ew}$ is its lower bound $Q(\theta_0)$, the probability of a codeword being moved outside its cone of half-angle $\theta_0$.

Before you read [1] and [7], let me summarize all the numerical evaluations by two formulas. The first one is used to find $\theta_0$ from $n$ and $R$, the second one to evaluate $Q(\theta_0)$, where $G = \frac{1}{2} \left[ \sqrt{\frac{2E_s}{N_0}} \cos \theta + \sqrt{\frac{2E_s}{N_0}} \cos^2 \theta + 4 \right]$. The two standard signal-to-noise ratios are related by $E_s / N_0 = R \times E_b / N_0$.

For the cone half-angle, please use

$$2^n R \approx \frac{\sqrt{2\pi n} \sin \theta_0 \cos \theta_0}{\sin^n \theta_0}$$

(1)

For the word error rate $P_{ew}$, please use

$$Q(\theta_0) \approx \frac{1}{\sqrt{n\pi}} \frac{1}{\sqrt{1 + G^2 \sin \theta_0}} \left[ G \sin \theta_0 \exp \left( -\frac{E_s}{N_0} + \frac{1}{2} \sqrt{\frac{2E_s}{N_0}} G \cos \theta_0 \right) \right]^n$$

$$\sqrt{\frac{2E_s}{N_0}} G \sin^2 \theta_0 - \cos \theta_0$$

(2)

The approximations are highly accurate in the above expressions. There is no need to compute the exact integrals involving the solid angle and the error probability [1]. In the next section, I give a C program implementation for $Q(\theta)$. Sections 3 and 4 illustrate some numerical examples.
2 C program implementation

/*****
Last Modification: 6 November 2006
*****
****
N.B.: Use formula (51) on page 632 of Shannon’s 1959 paper,
See also
*****/

#include "mathcom.h"

static int n;
static double RateFunction();

int main(argc, argv)
int argc;
char **argv;
{
  char chain[200], filename[200];
  double theta, rate;
  double a, b, c;
  int iter;
  double val, diff;
  double A, G, EL, factor, Pe, log_Pe, snrdb, snr;
  double snrdb1, snrdb2, snrstep;
  printf("######## Optimal Code (Spherical) Performance on AWGN Channel ######### \n");
  printf("Q(theta) formula (51) of Shannon 1959, page 632. \n");
  printf("Using the asymptotic rate formula (28), page 624 Shannon 1959\n");
  printf("Computing the cone half-angle for a given rate\n");
  printf("R=(1/n-1)*log2(sin(theta))+1/n*log2(cos(theta)*sqrt(2PIn)) \n\n");

  printf("Real Space dimension (from 20 up to 200000) [1000] : ");
  ReadString(chain);
  if(chain[0]) sscanf(chain,"%d", &n);
  else n=1000;

  if((n<20)||(n>200000))
    {
      fprintf(stderr,"%s: n out of range 100...200000\n", argv[0]);
      return(-1);
    }

  printf("Information Rate in bits/dimension (from 1/10 up to 8) [0.50] : ");
  ReadString(chain);
  if(chain[0]) sscanf(chain,"%lf", &rate);
  else rate=0.50;
if((rate<0.1)||(rate>8.0))
{
    fprintf(stderr,"%s: rate out of range 1/10...8\n", argv[0]);
    return(-1);
}

/* rate=1/10 <=> theta0=00.20 degrees */
/* rate=8  <=> theta0=69.50 degrees */
iter=0;
  a=0.20;  b=70.0;
do
  {
      c=(a+b)/2.0;
      val=RateFunction(c);
      if(val < rate) b=c; else a=c;
      c=(a+b)/2.0; ++iter;
      diff=b-a;
  }
while((diff>1e-8)&&(iter<1000));

printf("The cone half-angle for \%1.4f bits/dimension is equal to \%1.8f degrees \n", rate, c);
theta=c*PI/180.0; /* we switch to radians */

if(SQR(tan(theta)) >= (0.25*n) )
{
    fprintf(stderr,"OmegaFunction(): Warning, theta too close to 90 or n is small !\n");
}

printf("The modulation alphabet is spherical, Es=energie per real symbol.\n");
printf("n*P=squared radius of the sphere= n*2*Es=n*2*rate*Eb, P=2*rate*Eb \n");
printf("A^2=P/N0=2*rate*Eb/N0, the SNR per bit is SNR=Eb/N0 \n");
printf("Here, N0 is the noise variance per real component.\n\n");

snrdb1=-1.50;
printf("Start Eb/N0 in dB [\%1.2f]: ", snrdb1);
ReadString(chain);
if(chain[0]) sscanf(chain,"%lf", &snrdb1);

snrdb2=20.00;
printf("End Eb/N0 in dB [\%1.2f]: ", snrdb2);
ReadString(chain);
if(chain[0]) sscanf(chain,"%lf", &snrdb2);

snrstep=0.05;
printf("SNR step in dB [\%1.2f]: ", snrstep);
ReadString(chain);
if(chain[0]) sscanf(chain,"%lf", &snrstep);
sprintf(filename,"optimal_spherical_code_awgn_n=%d_rate=%1.2f_WER.dat", n, rate);
printf("Output file name for WER versus SNR [%s]: ", filename);
ReadString(chain);
if(chain[0]) strcpy(filename, chain);

sprintf(chain,"rm -f %s", filename);
system(chain);

printf("Eb/N0 (dB) || Word Error Rate (WER)\n");
Pe=1.0;
for(snrdb=snrdb1; snrdb <= snrdb2+0.01; snrdb += snrstep)
{
    if(Pe<1e-20) break;
    snr=exp10(0.1*snrdb);
    A=sqrt(2*rate*snr);
    if(atan(1.0/A) > theta)
    {
        /***
         fprintf(stderr,"Warning: theta=%1.4f but lower limit is cot^{-1}(1/A)=%1.4f !\n", theta*180.0/PI, atan(1.0/A)*180.0/PI);
         fprintf(stderr,"%s: Please increase the SNR for this rate=%1.4f.\n", argv[0], rate);
        ***/
        continue;
    }
    G=0.5*(A*cos(theta)+sqrt(A*A*cos(theta)*cos(theta)+4.0));
    EL=0.5*A*A-0.5*A*G*cos(theta)-log(G*sin(theta));
    factor=sqrt(n*PI)*sqrt(1.0+G*G)*sin(theta)
            *(A*G*sin(theta)*sin(theta)-cos(theta));
    Pe=exp(-n*EL)/factor;
    if(Pe<1.0)
    {
        printf("%1.2f %1.4e \n", snrdb, Pe);
        WritePlotFile(filename, snrdb, Pe, "%1.2f", "%1.4e");
    }
}/* end of snrdb loop */

return(0);
}/* end of main() */

/*---------------------------------------------------------------*/
static double RateFunction(theta)
    double theta; /* in degrees, not radians */
{
    return((1.0/n-1.0)*log2(sin(theta*PI/180.0))
            +1.0/n*log2(cos(theta*PI/180.0)*sqrt(2.0*PI*n))
             );
}/* end of RateFunction() */

/*---------------------------------------------------------------*/
3 Word Error Rate versus Information Rate

We illustrate in this section the word error rate of optimal spherical codes on a Gaussian channel at finite length $n = 100, 500, 1000, \text{ and } 2000$. Ten different values of the information rate $R$ are considered, $1/10 \leq R \leq 4$. The word error rate $P_e$ versus the signal-to-noise ratio per bit $E_b/N_0$ is given in Figures 1-4.

![Performance of Optimal Codes, $n=100$ real dimensions, AWGN channel](image1)

**Figure 1:** Word error rate versus signal-to-noise ratio for length $n = 100$.

![Performance of Optimal Codes, $n=500$ real dimensions, AWGN channel](image2)

**Figure 2:** Word error rate versus signal-to-noise ratio for length $n = 500$. 


Figure 3: Word error rate versus signal-to-noise ratio for length $n = 1000$.

Figure 4: Word error rate versus signal-to-noise ratio for length $n = 2000$. 
4 Application to VDSL2

Turbo trellis coded modulations (TTCM) [4] have been proposed for channel coding of VDSL2 using QAM modulations for a large range of spectral efficiency values [8]. The TTCM is serially concatenated with an outer code. The latter is a standard Reed-Solomon (RS) of length 255 and dimension 239 defined over the field $GF(256)$. As designed in [8], a 128-QAM constellation is adequately partitioned and labelled, two bits in a QAM symbol label are precoded with a rate $1/2$ binary parallel turbo code yielding a final spectral efficiency of 5 bits/sec/Hz. After taking into account the RS coding rate, the information rate of RS+TTCM is $R = 239/255 \times 5 = 2.34$ bits per real dimension. The TTCM block length is 1022 QAM symbols (Turbo interleaver of size 2044 bits). Hence, the code is in a real space of dimension $n = 2044$.

Let us compare the performance of the RS+TTCM to the error rate of an optimal code having the same parameters. In order to convert the word error probability $P_{ew}$ given by (2) into a bit error probability $P_{eb}$, we propose the following: Assume that a codeword on the $n$-dimensional sphere is surrounded by $n$ neighbours and assume that decoding errors yield only one of those neighbours. Each codeword is labelled by $nR$ bits. The considered neighbours are labelled by $\log_2(\tau_n)$ bits, suppose that $\log_2(\tau_n) < nR$. If random binary labelling is used to index the $\tau_n$ neighbours, then we have

$$P_{eb} \approx \frac{\frac{1}{2} \log_2(\tau_n)}{nR} P_{ew}$$

Let $\tau_n^*$ denote the greatest value attained by the kissing number of an $n$-dimensional sphere packing. It is known that [5]

$$2^{0.2075 n(1+o(1))} \leq \tau_n^* \leq 2^{0.401 n(1+o(1))}$$

The lower bound has been proved by Kabatiansky and Levenshtein [3] and the upper bound by Wyner [2]. Finally by using the right inequality in (4) we get

$$P_{eb} \lesssim \frac{0.401}{2R} P_{ew}$$

Figure 5 illustrates the bit error rate of RS+TTCM versus optimal codes at finite length. The coding gain gap is about 2.45 dB (3dB from capacity limit at $n = +\infty$). The capacity limit is given by the rate-distortion bound

$$P_{eb}(n = +\infty) \geq H_2^{-1} \left( 1 - \frac{C}{R} \right)$$

Since $R$ is the rate per real dimension, then $C = \frac{1}{2} \log_2(1 + 2R \frac{E_b}{N_0})$. Multilevel coded modulations with multistage decoding [6] exhibit performance similar to those of RS+TTCM. We believe that feasible coded modulations exist at less than 1dB from optimal codes at such high information rates.

5 Download in PDF format

Please use the URL below to download this document in PDF

http://www.comelec.enst.fr/~boutros/coding/Qtheta_large_n_anyR.pdf
Figure 5: Reed-Solomon (255, 239)$_{256}$ concatenated with a Turbo trellis coded 128-QAM versus optimal spherical codes of finite length. Information rate is 2.34 bits per dimension.

References


