Computing the needed redundancy overhead of an erasure resilient code for a successful media transmission over a network with random packet loss probability

Emin Gabrielyan
EPFL / Switzernet
emin.gabrielyan@{epfl.ch, switzernet.com}

2005-10-27
HTML – DOC – PDF

Table of contents:

Abstract: Packetized communications over internet behave like erasure channels. For example, files or real-time media sent over the internet are chopped into packets, and each packet is either received without error or not received. Assuming Maximum Distance Separable (MDS) erasure resilient codes; we present a method for computing the codeword overhead (needed for a satisfactory transmission) as a function from the random loss probability in the network.

I. Secure Media Streaming

Assuming an erasure channel and Maximum Distance Separable (MDS) systematic codes (such as Reed Solomon codes, or see an erasure resilient checksum code example) given is the following problem:

- We are streaming an on-line media from a sender to the receiver over a network with a packet's random loss probability *p*
- In order to compensate *p* percent of network losses, the sender, after every *M* transmitted media packets, is adding some redundant packets relative to these last *M* packets. Since the encoder is systematic and is using MDS codes, the receiver can restore the initial media if any of *M* packets out of all transmitted packets are survived (out of *M* media packets together with their redundant packets)

- Since we are dealing with on-line media (such as a VOIP phone conversation), *M* cannot be infinitely large
- Mean of the number of lost packets at the receiver for a block of *N* transmitted packets is $N \cdot p$, but with random loss pattern, the probability of receiving less than $N \cdot p$ can not be neglected
- Let $N \geq M$ be the number of packets in the block containing *M* media packets and $N - M$ redundancy packets. Let the desired probability of unsuccessful decoding at the receiver be DER (Decoding Error Rate)

By sufficiently increasing the *N*, for a given *M* and *p* we can reduce the probability of decoding failure to any desired rate. Our question is, which is the smallest *N* for a given $0 \leq p < 1$ that keeps the decoding failure probability at the receiver below a given DER.

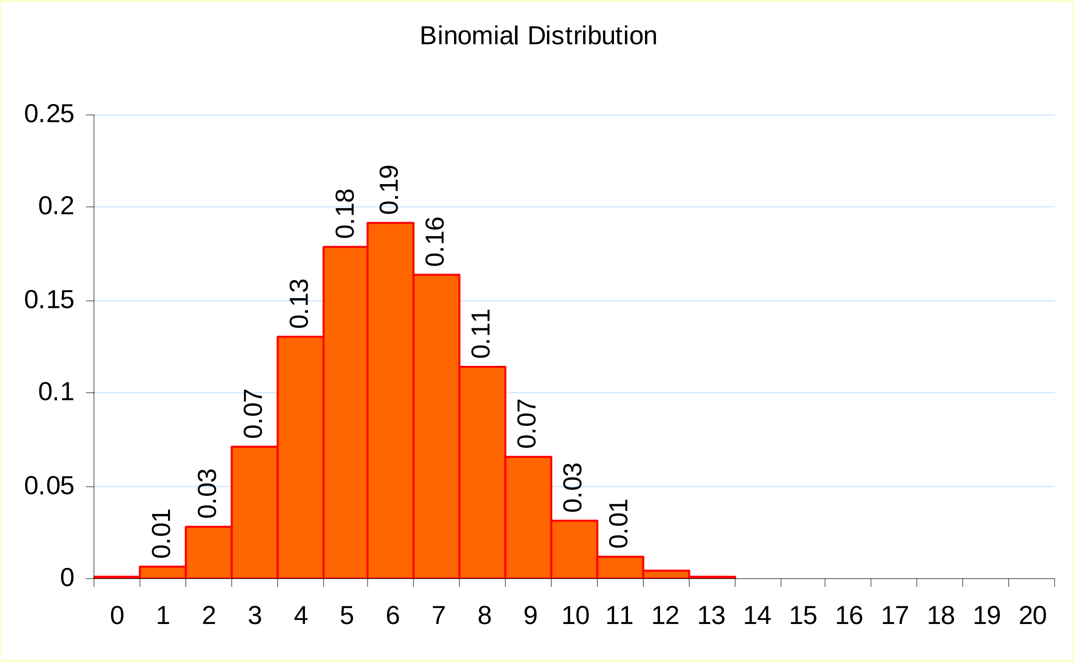
II. Binomial Distribution

The probability of having *n* erasures in a block of *N* packets with a random loss probability *p* is known as binomial distribution and denoted as:

$$P_p(n|N) = \begin{pmatrix} N \\ n \end{pmatrix} \cdot p^n \cdot q^{N-n}$$

where $\begin{pmatrix} N \\ n \end{pmatrix} = \dfrac{N!}{n! \cdot (N-n)!}$

and $q = p - 1$

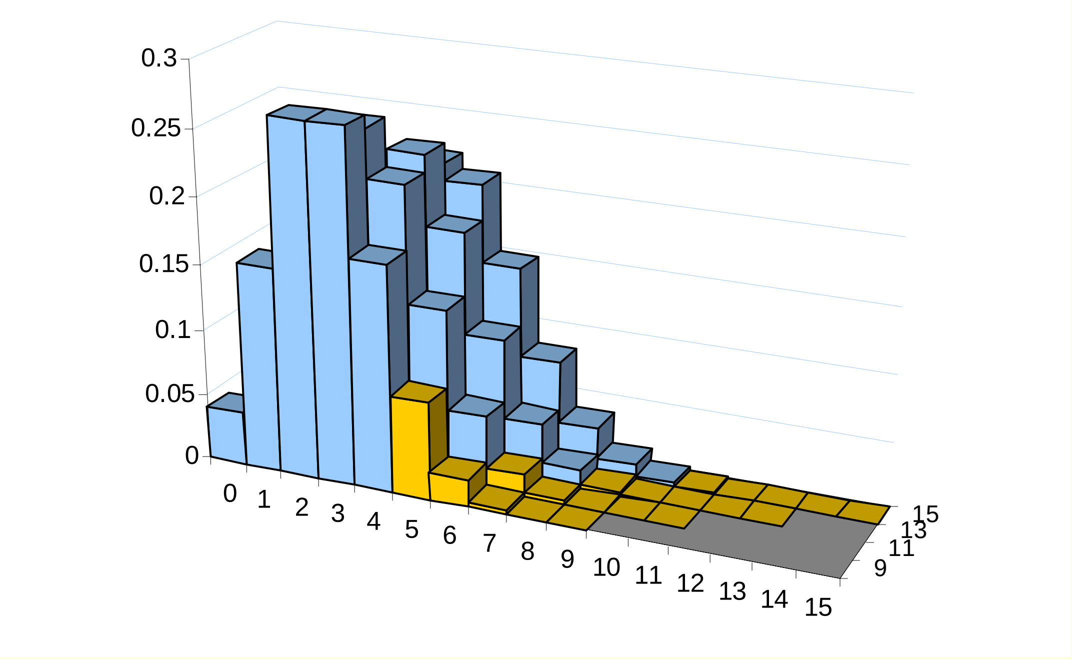


The above plot shows the distribution of n erasures out of $N = 20$ packets with $p = 0.3$

The probability of having less than M survived packets in a block of N packets is the probability of having $N - M + 1$ or more erasures; it is computed and denoted as follows:

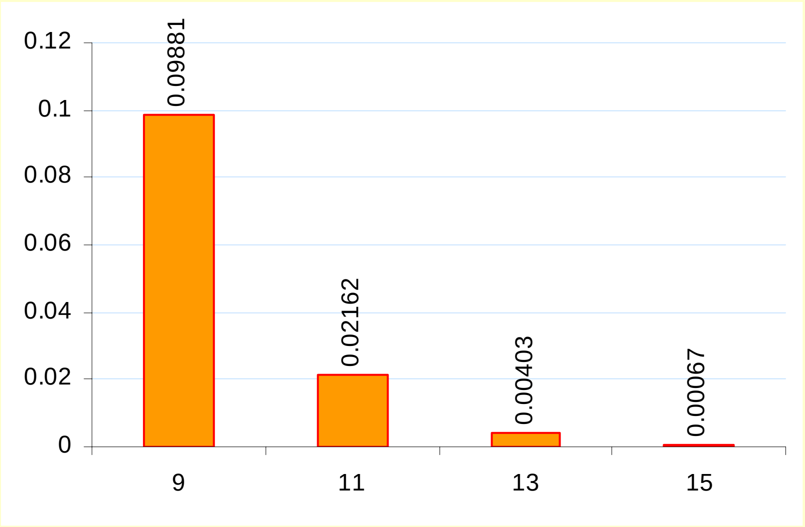
$$I_p(N - M + 1, M) = \sum_{n=N-M+1}^{N} \binom{N}{n} \cdot p^n \cdot q^{N-n}$$

For a given M and p, with increase of N this probability decreases. Assume M is 5 and $p = 0.3$. The below plot shows four Binomial distributions for a value of N starting from 9 and enlarging up to 15.



For a given N the sum of histograms marked in yellow $I_p(N - M + 1, M)$ is the probability of having more than $N - M$ erasures, i.e. the probability of decoding failure at the receiver.

These probabilities are plotted on the chart below.



If our objective (in the same example with $M=5$ and $p=0.3$) is to have a DER below 1%, we can transmit the media in blocks of 13 packets (containing 8 redundant packets).

For a given *M* and DER we need a fast method for computing the minimal required *N* needed for successful media streaming at network's random loss probability *p*.

III. Fast computation of Binomial Distribution

The below representation of binomial contains less multiplications and divisions and we have no components containing very large numbers and therefore have less precision errors:

$$\binom{N}{k}\cdot p^k\cdot q^{N-k}=\begin{cases} q^N & if \quad k=0 \\ \prod_{i=1}^{k}\left(\frac{N-k+i}{i}\cdot pqq\right) & if \quad 0<k\le\frac{N}{2} \\ \prod_{i=1}^{N-k}\left(\frac{k+i}{i}\cdot qpp\right) & if \quad \frac{N}{2}<k<N \\ p^N & if \quad k=N \end{cases}$$

$$pqq=p\cdot q^{\frac{N-k}{k}}$$

$$qpp=q\cdot p^{\frac{k}{N-k}}$$

Additional speedup is obtained by applying the above method for computing the binomial only at $k=round(N\cdot p)$ and by obtaining the remaining values of the binomial, iteratively, from the value computed at *k*:

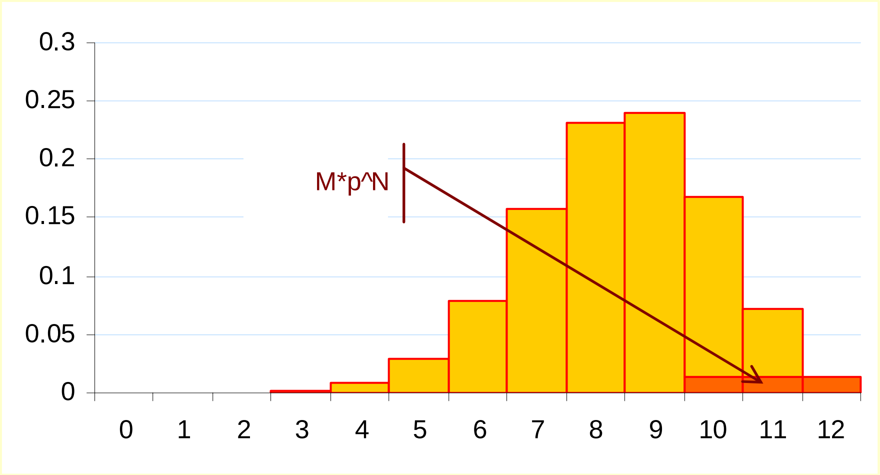
$$p_p(n|N)=\begin{cases} p_p(n-1|N)\cdot\dfrac{N-n+1}{n}\cdot\dfrac{p}{q} \\ \qquad\qquad or \\ p_p(n+1|N)\cdot\dfrac{n+1}{N-n}\cdot\dfrac{q}{p} \end{cases}$$

Below is an implementation of the algorithm in AMPL:

```
reset;
param ver symbolic = "a46a";

param p;
param q=1-p;
param N;

param k=round(N*p);

param pq=
 if k in interval [1,N/2]   then p*q^((N-k)/k) else
 if k in interval (N/2,N-1] then p^(k/(N-k))*q ;

param Bk=
  if k==0 then     q^N else
  if k==N then     p^N else
  if k in interval [1,N/2]   then prod{i in 1..k}   ((N-k+i)/i*pq) else
  if k in interval (N/2,N-1] then prod{i in 1..N-k} ((k+i)/i*pq) ;

param Binomial{n in 0..N} =
  if n=k then Bk
  else if n>k then Binomial[n-1]*(N-n+1)/n*p/q
  else if n<k then Binomial[n+1]*(n+1)/(N-n)*q/p;

data;
param N=20;
param p=0.3;
model;

param file symbolic = ver & "-out.csv";
printf ",Binomial Distribution\n" > (file);
for{n in 0..N}
{
  printf "%d,%.20f\n",n,Binomial[n] > (file);
}
close (file);
```

IV. Computing the minimally needed length of the transmitted codeword for maintaining the decoding failure probability satisfactorily low

The decoding failure probability for a given N is always more than $M \cdot p^N$



Binomial distribution for $N = 12$, $M = 3$, $p = 0.7$

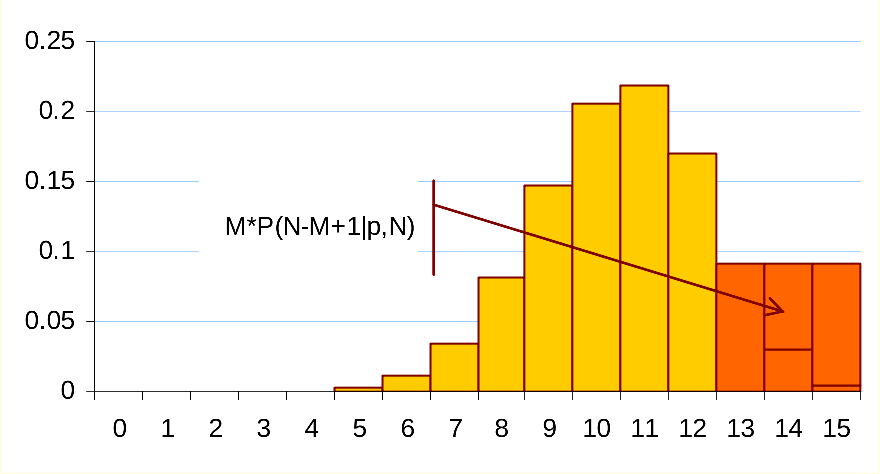
Therefore the lower bound for N seeking a failure probability less than or equal to DER is:

$$\underline{N} = \log_p\left(\frac{DER}{M}\right)$$

From the other side if for a given $\overline{N}$

$M \cdot P_p\left(\overline{N} - M + 1 \middle| \overline{N}\right) \le DER$

Then $N \le \overline{N}$



Binomial distribution with $N = 15$, $M = 3$, $p = 0.7$

The upper bound $\overline{N}$ for N is chosen as the smallest from sequence $2 \cdot \underline{N}, 4 \cdot \underline{N}, 8 \cdot \underline{N}, \ldots,$ for which $M \cdot P_p\left(\overline{N} - M + 1 \middle| \overline{N}\right) \le DER$:

$$\overline{N} = \min_{i=1}^{c\geq 4}\left(2^{i} \cdot \underline{N}\right)$$

such that:

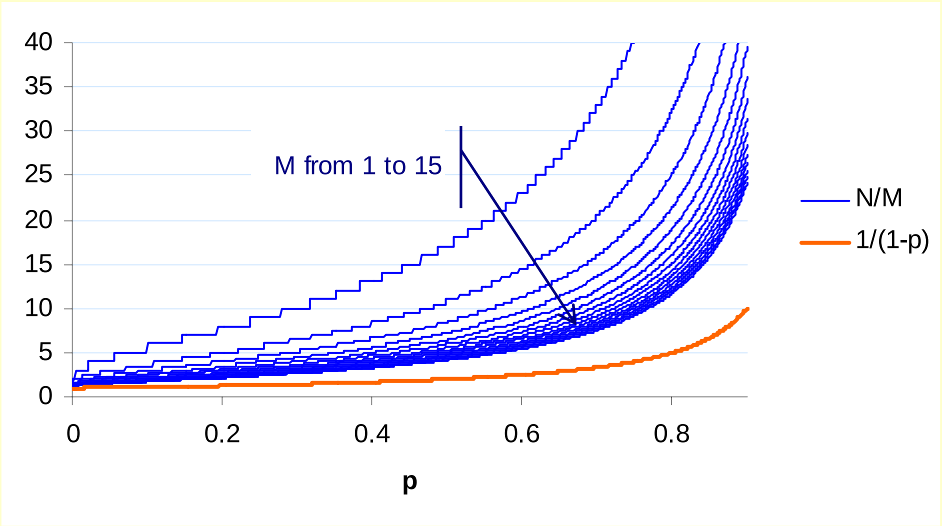
$$\prod_{k=1}^{M-1}\left(\frac{2^{i} \cdot \underline{N} - M + 1 + k}{k} \cdot p^{\frac{2^{i} \cdot \underline{N} - M + 1}{M-1}} \cdot q\right) \leq \frac{DER}{M}$$

For every $N \in \left\{\underline{N} ... \overline{N}\right\}$ in the interval from the lower bound to the upper bound, we compute the exact value of the failure probability, and chose the lowest *N* at which the failure probability is below DER.

Below is an implementation of the algorithm in AMPL:

```
reset;
param ver symbolic = "a73a";

param M integer >=1;
param DER;

set P within interval [0,1] = setof{p in 0..0.9 by 0.001} round(p,10);

param minN{p in P} = max(M,if p=0 then 0 else
floor(log10(DER/M)/log10(p)));

param maxN_probe{p in P, pr in 1..4} = minN[p]*2^pr;

param Binomial_probe{p in P, pr in 1..4} =
  if M>=2 then
    prod{i in 1..M-1}
    (
      (maxN_probe[p,pr]-M+1+i)/i *
        p^((maxN_probe[p,pr]-M+1)/(M-1))*(1-p)
    )
  else
    p^maxN_probe[p,pr];

param maxN{p in P} = maxN_probe[p, min{pr in 1..4:
Binomial_probe[p,pr]<=DER/M} pr];

set NN{p in P} = {minN[p]..maxN[p]};

param Binomial1{p in P, Ntr in NN[p]}=
  if M>=2 then
    prod{i in 1..M-1}((Ntr-M+1+i)/i*p^((Ntr-M+1)/(M-1))*(1-p))
  else
    p^Ntr;

param pbyq{p in P} = p/(1-p);
param Binomial{p in P, Ntr in NN[p], n in Ntr-M+1..Ntr} =
  if n=Ntr-M+1
  then
    Binomial1[p,Ntr]
  else
    Binomial[p,Ntr,n-1]*(Ntr-n+1)/n*pbyq[p];

param Failure{p in P, Ntr in NN[p]} = sum{i in Ntr-
M+1..Ntr}Binomial[p,Ntr,i];

param goodN{p in P} = min{Ntr in NN[p]: Failure[p,Ntr]<=DER} Ntr;

param file symbolic = ver & "-out.csv";

let DER := 1e-5;

param log symbolic = "benchmark.log";
param benchmark_rec symbolic;
param benchmark_start;
param benchmark_stop;
let benchmark_start := _ampl_time;
printf "Version %s:\n",ver;

printf ",FEC\n" > (file);
for{m in 1..15}
{
  let M := m;
  print M, DER;
  for{p in P}
  {
    printf "%f,%f\n",p,
      goodN[p]/M
      > (file);
  }
  printf "\n" > (file);
}
close (file);

let benchmark_stop := _ampl_time;
let benchmark_rec := sprintf("%s, %s, done in %f seconds",
  sub(ctime(),"^[^ ]* *",""),ver,benchmark_stop-benchmark_start);
print benchmark_rec;
print benchmark_rec >> (log);
close (log);
```

Codeword overheads as functions from random loss probability are plotted for various numbers of media packets in the block:



The smallest values of *N* at which the decoding failure probability is still below $DER = 10^{-5}$ for all *M* in {1..15}.

The orange curve represents the overhead of an ideal communication that reached its Shannon limit. Our curve will ultimately reach the Shannon limit when *M* is infinitely large. But since the streaming buffer is limited by real-time media constraints, *M* cannot grow too large and therefore the real time media streaming is necessarily separated from the Shannon limit by a certain gap.

V. Interpolation

Instead of an integer codeword length function of *p*, we would like to have a curve crossing smoothly also the non integer values between two $N-1$ and *N*.

Let us remind that *N* is chosen such that the failure probability at *N* is below DER and at $N-1$ is above:

$$I_p(N-1-M+1, M) \geq DER \geq I_p(N-M+1, M)$$

The failure probabilities for *N* and $N-1$ are known and already computed.

We know that the lower bound of $I_p(N-M+1, M)$ is $M \cdot p^N$ (when $M=1$, it is an exact value). Considering the shape of the lower bound $M \cdot p^N$ as sufficiently good approximation, we thus will chose a similar function for finding the real value *x* between two integers $N-1$ and *N*, at which "length" the failure probability is supposedly equal exactly to *DER*.

For two failure probabilities at *N* and $N-1$ we may say

$I_p(N-1-M+1, M) = M' \cdot p'^{N-1}$

$$I_p(N-M+1,M)=M'\cdot p'^N$$

where M' and p' are some accordingly chosen values for the two probabilities at $N-1$ and N

Precisely the values of M' and p' are the following:

$$p'=\frac{I_p(N-M+1,M)}{I_p(N-1-M+1,M)}$$

$$M'=\frac{I_p(N-M+1,M)}{p'^N}$$

But as we will see below we will not need to refer to the values of M' and p':

The real value $N-1\leq x\leq N$ is computed to respect the same interpolation:

$$M'\cdot p'^{N-1}=I_p(N-1-M+1,M)$$

$$M'\cdot p'^x=DER$$

$$M'\cdot p'^N=I_p(N-M+1,M)$$

$$\begin{aligned}
x&=\log_{p'}\frac{DER}{M'}\\
&=\log_{p'}\frac{DER}{I_p(N-M+1,M)}\cdot p'^N\\
&=N+\log_{p'}\frac{DER}{I_p(N-M+1,M)}\\
&=N+\log_{p'}\frac{DER}{I_p(N-M+1,M)}\\
&=N-\frac{\ln DER-\ln I_p(N-M+1,M)}{\ln I_p(N-1+M+1,M)-\ln I_p(N-M+1,M)}\\
&=N-1+\frac{\ln I_p(N-1+M+1,M)-\ln DER}{\ln I_p(N-1+M+1,M)-\ln I_p(N-M+1,M)}
\end{aligned}$$

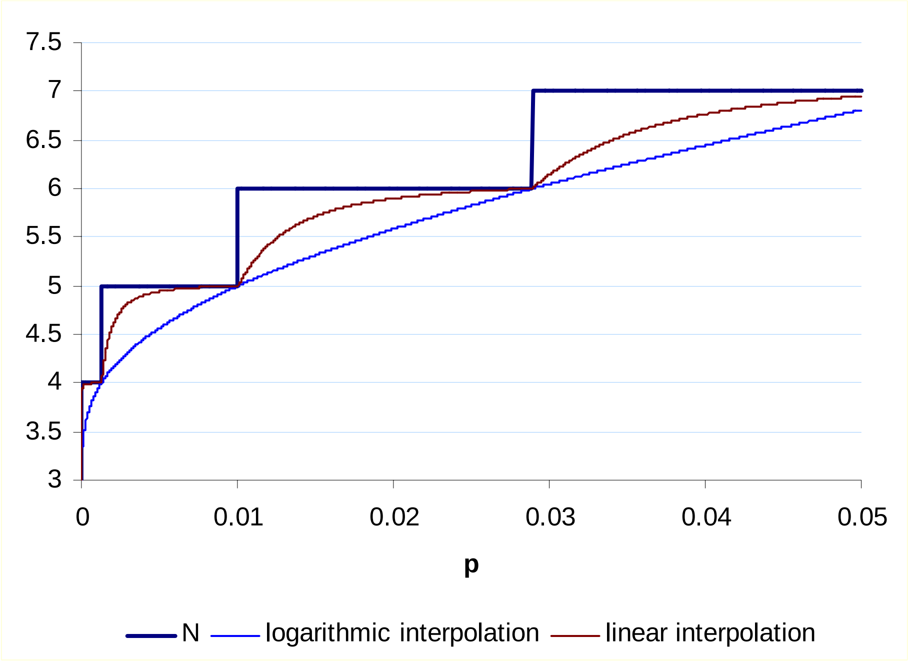
Here is the corresponding AMPL plug-in to be added to the previous script:

```
param codeword{p in P} =
 if goodN[p]-1 in NN[p] and Failure[p,goodN[p]-1]>DER
   then
     goodN[p]-1 +
       (log(Failure[p,goodN[p]-1]) - log(DER)) /
       (log(Failure[p,goodN[p]-1]) - log(Failure[p,goodN[p]]))
   else goodN[p];
```

The full text of the AMPL script is here.

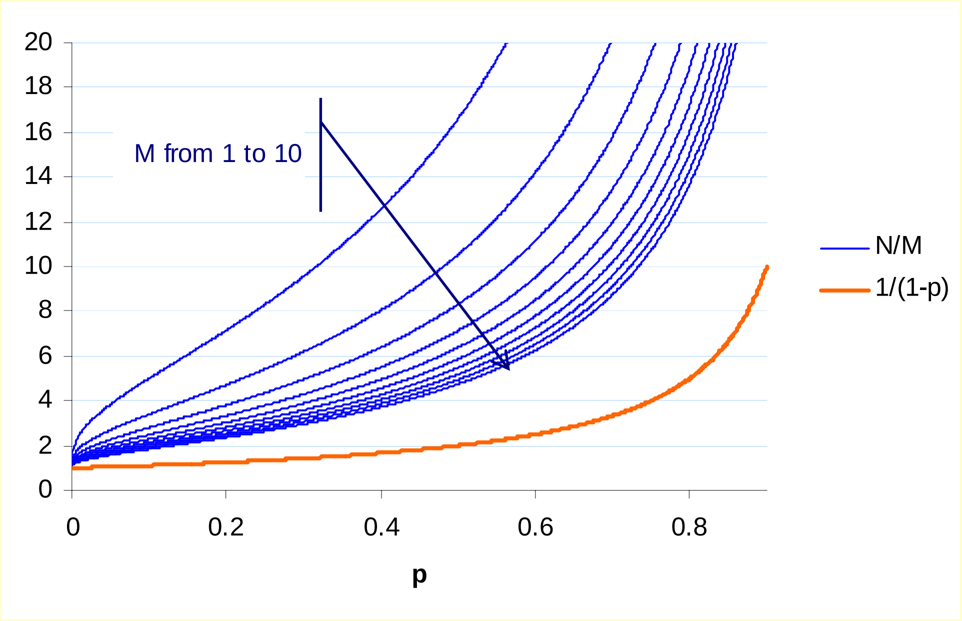
Just for a comparison, the below chart compares the above obtained logarithmical interpolation with this the following linear formula for *x*, whose curve has much worst shape.

$$x = N - 1 + \frac{I_p(N-1+M+1, M) - DER}{I_p(N-1+M+1, M) - I_p(N-M+1, M)}$$



Codeword size to be transmitted for satisfying $DER = 10^{-5}$ with $M = 3$

Below are smoothened curves of required codeword sizes for various *M*. The orange curve is the theoretical lower bound resulted from the Shannon limit.



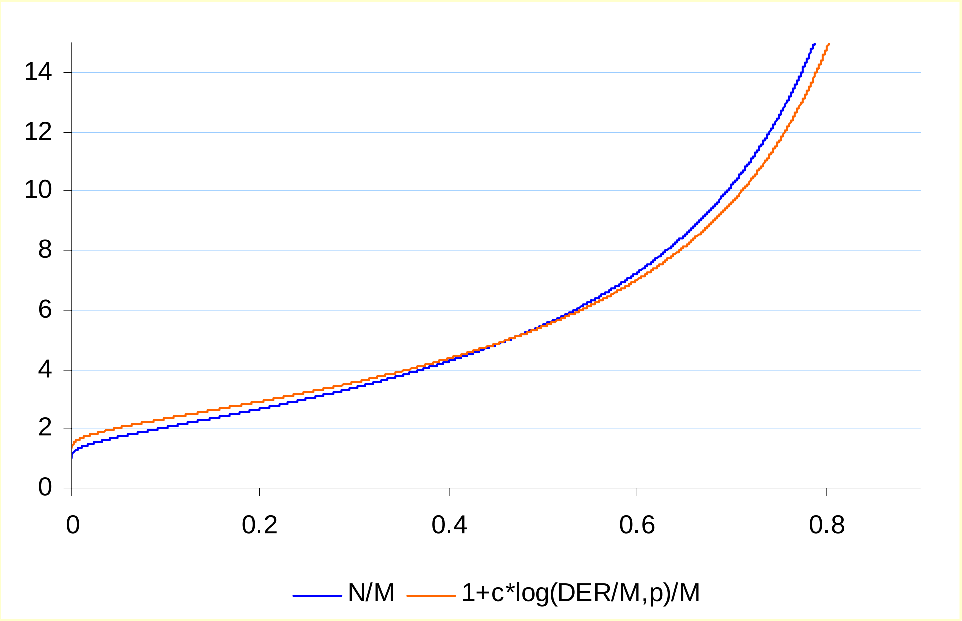
Codeword overhead $\frac{N}{M}$ for $DER = 10^{-5}$ with $M \in \{1..10\}$

VI. A coarse approximation

The codeword overhead $\frac{N}{M}$ can be quite well approximated with

$$1 + c \cdot \frac{\log_p\left(\frac{DER}{M}\right)}{M}$$

where *c* is some constant (see an excel file)



Codeword overhead $\frac{N}{M}$ for $DER = 10^{-5}$ with $M = 7$ and $c = 1.6$

*   *   *

US – Mirror
CH – Mirror