Density evolution of LDPC ***by Andrew W. Eckford***

Implementation:

This package contains MATLAB scripts which implement Richardson and Urbanke's density evolution technique to find the ultimate performance of LDPC codes in memoryless channels.

Implementation content:

Quantification The density evolution of regular LDPC and irregular LDPC, when the quantization step size is infinitely small, that is, the density at the output of the Bi-AWGN channel evolves.

Example of use:

Example 1: Binary Symmetric Channel, Transition Probability 8.394%, Regular LDPC

```
% quantized channel input, input amplitude is divided into 6001 equal parts
    chan = zeros(1,6001);
 % chan(3000) is the zero mean point, so chan(3240) is the correct probability, and chan(2762) is the error transition probability.
    chan(3240) = 91.606;
    chan(2762) = 8.394;
 % quantizes the input data format. This format maps the input to the [-30,30] interval and quantizes the 6000 equal parts.
    ext = [-30 0.01 6001];
 % Quantification of the complex function of the intermediate check node, which will be studied later
    mapping = [-10 0.0002 50000];
% variable node weight
    dv = 3;
% check node weight
    dc = 6;
% number of iterations
    iter = 5;
% target ber (changing the DE condition constantly, making the output threshold close to the target)
    stop_pe = 1e-5;
% function call
```

```
19.     result_pe = de_regular(chan,iter,ext,mapping,stop_pe,dv,dc)
20.
21. % output
22.     result_pe =
23.
24.     0.0839    0.0788    0.0756    0.0730    0.0710
```

Example 2: Two-input symmetric AWGN output channel, regular LDPC

```
1.      chan = zeros(1,6001);
2.      ext = [-30 0.01 6001];% input quantization
3.      mapping = [-10 0.0002 50000];% quantizaiton for check node values
4.      dv = 3;
5.      dc = 6;
6.      iter = 20;
7.      stop_pe = 1e-5;
8.   % squre root of variance for Guass
9.      noise=1.0;
10. % Gaussian channel quantization, using the complement error function erfc() to calculate the quantized value
11.     chan=chan_mess(ext,noise);
12.     result_pe = de_regular(chan,iter,ext,mapping,stop_pe,dv,dc)
```

Example 3: Two-input symmetric AWGN output channel, irregular LDPC

```
ext = [-30 0.01 6001];
mapping = [-10 0.0002 50000];
iter = 100;
stop_pe = 1e-5;
% degree distribution polynomial
vard(1,:)=[0 0.2895    0.3158 0 0  0.3947];
chkd(1,:)=[ 0 0 0 0 0 0 0.9032  0.0968];
% squre root of variance for Guass
% noise from low to high, the step size is ns_stp, and gradually try out the threshold that matches the stop_pe
ns_str_low=0.9;
ns_str_high=0.94;
ns_stp=0.01;
% includes the statistics of the channel information threshold=threshold_tst(vard,chkd, ns_str_low, ns_str_high, ns_stp,
ext, mapping, iter, stop_pe);
```

General decoding and density evolution process analysis:

Density evolution is the transfer of PDF, and its threshold is the integration of the iterative PDF, and the judgment after the BER is determined whether the set BER requirement is reached. So the value of the intermediate iteration is a PDF function, which is the mean of the statistical variables. The variable of the rule code and the number of nodes participating in the check node update step are equal, so the number of convolutions of the PDF in the update is the same; the number of the irregular code and the number of nodes participating in the check node update step are not equal, so The degree distribution polynomial is used to find the average after iteration.

Density evolution process

---

(1) Setting parameters:

- Given the current codeword, density evolution is**Acyclic assumption**Sparse check matrix**Degree distribution**Passed by**PDF**Iteration, so the code word is distributed by degree (d_v, d_c) It is determined that the rule code is represented by a degree distribution constant, and the irregular code is represented by a degree distribution polynomial;
- set a goalber^*, that is, the current codeword is designed in its working environment;
- Channel parameterα, may be the deletion probability of the BEC, may be the transition probability of the BSC, may be the noise variance of the AWGN, and determine the channel parameter threshold under the ber that meets the requirement by adjusting the channel parameters;

(2) Iterative steps:

Continuous density evolution derivation

LDPC iteration **（Sum-Product Algorithm）**	**Density iteration (original Fourier transform version)**
a. Initialization: The channel information of the initial input decoder is a log likelihood ratio: $L_J^0 = \log \frac{P(y\|x=+1)}{P(y\|x=-1)} = \frac{2y}{\sigma^2}$ And initialize the variable node:$L_{j\to i}^0 = L_j^0$ The number of iterations k=1.	<Channel codes classical and modern, chapter5 and chapter 9> Consider the binary input AWGN,$x \in \{+1, -1\}$, the output is y=x+n, where$n \sim \mathcal{N}(0, \sigma^2)$。 Obviously because$y \sim \mathcal{N}(1, \sigma^2)$,and so $L_j^0 \sim \mathcal{N}(\frac{2}{\sigma^2}, \frac{4}{\sigma^4} * \sigma^2)$, ie var=2mean. ***If the input is non-AWGN, the distribution of the LLR needs to be derived separately.*** This way we can find the PDF of the LLR information. L_j^0 PDF is marked as p_c^0
b. VN update: $L_{j\to i}^k = L_j^0 + \sum_{i'(j)-\{i\}} L_{i'\to j}^{k-1}$	This step is because the random variables are summed, andL_j^0 Obey the initial PDF,$L_{i'\to j}^{k-1}$For the random variable after iteration k-1 times, the PDF is different from the initial PDF, but because it is a

	ringless assumption, the distribution of each external node variable is assumed to be the same, that is, the PDF update formula can be written as (The PDF after the summation of random variables is the convolution of their respective PDFs) : $p_v^k = p_c^0 * [p_c^{k-1}]^{*(d_v-1)}$, * is convolution. The d_v of the rule code is the same, and the update formula of the rule code is: $p_v^k = p_c^0 * \sum_{d=1}^{d_v} \lambda_d (p_c^{k-1})^{*(d_v-1)}$ It can be seen that the update formula of the irregular code is a step of increasing the averaging. Its fast calculation can replace the convolution with FFT: $p_v^k = \mathcal{F}^{-1}\{\mathcal{F}\{p_c^0\}][\mathcal{F}\{p_c^{k-1}\}]^{d_v-1}\}$(rule code)
c. CN update: **Original multiplication formula:** $L_{i\to j}^k = 2tanh^{-1}(\prod_{j'\in N(i)-\{j\}} tanh(\frac{1}{2}L_{j'\to i}^{k-1}))$ **The simplification formula ([multiplication on the real number field] becomes [symbol field multiplication + continuous addition on the real field]):** $L_{i\to j}^k = \prod_{j'\in N(i)-\{j\}} sign(L_{j'\to i}^{k-1}) \times \phi(\sum_{j'\in N(i)-\{j\}} \phi(\lvert L_{j'\to i}^{k-1}\rvert))$	The derivation of the check node is also based on the original SP formula, here for the simplified formula, $L_{i\to j}^k = \prod_{j'\in N(i)-\{j\}} sign(L_{j'\to i}^{k-1}) \times \phi(\sum_{j'\in N(i)-\{j\}} \phi(\lvert L_{j'\to i}^{k-1}\rvert))$ The first consecutive multiplication term on the right is symbol-multiplied, which is not conducive to deriving PDF, so $s_{j'\to i}^{k-1} = log_{-1} sign(L_{j'\to i}^{k-1})$, Then the contingency is changed to the summation term $L_{i\to j}^k = mod(\sum_{j'\in N(i)-\{j\}} s_{j'\to i}^{k-1}, 2) \times \phi(\sum_{j'\in N(i)-\{j\}} \phi(\lvert L_{j'\to i}^{k-1}\rvert))$

among them, $\phi(x) = -ln[tanh(\frac{x}{2})] = ln\frac{e^x+1}{e^x-1}$

At this point, it is natural to change the LLR into a symbol (where the symbol is true, the sign is a base-to-logarithm, 0 or 1) and the amplitude:
$L_{i\to j}^k = [s^k, |m^k|]$

Start deriving variable node PDF calculation
Need knowledge: "LDPC code base and should, He Heyun compiled" p.159
Let the random variable X have a probability density function f_x(x), and let g(x) be continuous and have g'(x)>0 (or g'(x)<0), then y=g(x) is a continuous random variable, x = h (y) is the inverse of y = g (x), and its probability density function is:
$f_Y(y) = f_X(h(y))|h'(y)|,$
$min(g(-\infty), g(+\infty)) < y < max(g(-\infty), g(+\infty))$

STEP 1: Find the second sum in the right $\phi(|L_{j'\to i}^{k-1}|)$ PDF
(1) When $s_{j'\to i}^{k-1} = 0$ **Time,** $L_{j'\to i}^{k-1} > 0$
$g(x) = y = \phi(L_{j'\to i}^{k-1}) = -lntanh(\frac{L_{j'\to i}^{k-1}}{2})$
$g'(x) = \phi'(L_{j'\to i}^{k-1}) = \frac{1}{sinh(L_{j'\to i}^{k-1})}, g'(x) < 0$
,
$L_{j'\to i}^{k-1}$ PDF for p_v^{k-1}
because $\phi(x)$ of **Inverse function** Symmetry, ie $\phi(x) = \phi^{-1}(x)$ Or write as $y = \phi(x), x = \phi(y)$,
We can

$g(x) = y = -lntanh(\frac{L_{j'\to i}^{k-1}}{2})$

get:

$h(y) = L_{j'\to i}^{k-1} = -lntanh(\frac{y}{2})$

$h'(y) = \frac{1}{sinh(y)}$

then:

$y = \phi(|L_{j'\to i}^{k-1}|)$The probability density function is:

$$P(0,y) = p_v^{k-1}(-\ln\tanh(\frac{y}{2}))\frac{1}{\sinh(y)}, L_{j'\to i}^{k-1} > 0$$

same,

(2) When $s_{j'\to i}^{k-1} = 1$**Time,** $L_{j'\to i}^{k-1} < 0$

get

then:

$y = \phi(|L_{j'\to i}^{k-1}|)$The probability density function is:

$$P(1,y) = p_v^{k-1}(-\ln\tanh(\frac{-y}{2}))\frac{1}{\sinh(-y)}, L_{j'\to i}^{k-1} < 0$$

STEP 2: Calculation$w = \sum \phi(|L_{j'\to i}^{k-1}|) = \sum y$PDF

Using the rule of variable node update, the random variable is summed as PDF convolution, but the information at this time is divided into positive and negative cases, so when using Fourier transform, it needs to be performed by two-dimensional Fourier transform. One is a binary domain transform {0, 1}, and the second is a real domain transform [0, +\infty).

	convolution:$p(w) = p(y)^{*(d_c-1)}$ Fourier transform: First of all $\mathcal{F}\{p(y)\}_{(0,w)} = \mathcal{F}\{p(0,y)\}_w + \mathcal{F}\{p(1,y)\}_w$ $\mathcal{F}\{p(y)\}_{(1,w)} = \mathcal{F}\{p(0,y)\}_w - \mathcal{F}\{p(1,y)\}_w$ then $\mathcal{F}\{p(w)\}_{(0,w)} = [\mathcal{F}\{p(y)\}_{(0,w)}]^{d_c-1}$ $\mathcal{F}\{p(w)\}_{(1,w)} = [\mathcal{F}\{p(y)\}_{(1,w)}]^{d_c-1}$ At last $p(w) = \mathcal{F}^{-1}\{p(w)\}$ **STEP 3:**Calculation $m^k = \phi^{-1}(w) = \phi^{-1}(\sum y) = \phi^{-1}(\sum \phi(\lvert L_{j'\to i}^{k-1}\rvert))$PDF Similar to the derivation of STEP 1, $P(0,m^k) = p_v^{k-1}(0,y)\frac{1}{\sinh(m^k)}\lvert y = \phi(m^k), s = 0$ $P(1,m^k) = p_v^{k-1}(1,y)\frac{1}{\sinh(-m^k)}\lvert y = \phi(-m^k), s = 1$ The Fourier transform mentioned above has not been studied in depth. It is only a matter of combining and understanding the knowledge in the book with its own understanding. The discrete density evolution and the continuous situation are very different.
d. LLR summation: $L_j^{total} = L_j^0 + \sum_{i\in N(j)} L_{i\to j}^{k-1}$	Density evolution actually only corresponds to the above iterative process, and each iteration is obtained.p_v^kAfter that, by calculating the integral and comparing it with the set ber

	$\int_{-\infty}^{0} p_v^k(x)dx \leq ber^*$ To decide whether to continue iterating or adjust channel parametersα
e. Judgment: • $\hat{v} = 1, if L_j^{total} < 0; \hat{v} = 0, other.$ • if $\hat{v}H^T = 0$, exit and output; otherwise k++ and goto **step b**.	

Discrete density evolutionary derivation

LDPC iteration **（Box-plus Algorithm）** This decoding algorithm is to solve $\phi(x) = -ln[tanh(\frac{x}{2})] = ln\frac{e^x+1}{e^x-1}$ The function causes a problem that is difficult to fit, that is, there is a performance loss when using a lookup table, so a box addition operation is employed. The most important step is to replace it with a box plus operation. $L_{i\to j}^k = 2tanh^{-1}(\prod_{j'\in N(i)-\{j\}} tanh(\frac{1}{2}L_{j'\to i}^{k-1}))$, Box plus box-plus see CN node update.	**Density iteration (quantized version)** <Channel codes classical and modern, chapter5 and chapter 9> The code is not the way to use this, you can find out.
a. Initialization: The channel information of the initial input decoder is a log	Find the PDF of the LLR information: L_j^0PDF is marked asp_c^0 The PDF is quantized, that is, the PDF in the quantization interval is integrated, and the probability of occurrence of

likelihood ratio: $L_J^0 = \log \frac{P(y\|x=+1)}{P(y\|x=-1)} = \frac{2y}{\sigma^2}$ Correct L_J^0 Quantify, And initialize the variable node: $L_{j\to i}^0 = 0$ The number of iterations k=1.	the quantized signal, that is, the probability mass function PMF, also known as the probability spectral density, is obtained.
b. VN update: $L_{j\to i}^k = L_j^0 + \sum_{i'(j)-\{i\}} L_{i'\to j}^{k-1}$	Consistent with the non-quantized version, this step is because of the random variable summation, for the regular LDPC code $p_v^k = p_c^0 * [p_c^{k-1}]^{*(d_v-1)}$, * is a discrete convolution. Similarly, its fast calculations can use FFT instead of convolution: $p_v^k = \mathcal{F}^{-1}\{\mathcal{F}\{p_c^0\}][\mathcal{F}\{p_c^{k-1}\}]^{d_v-1}\}$
c. CN update: **Original multiplication formula:** $L_{i\to j}^k = 2tanh^{-1}(\prod_{j'\in N(i)-\{j\}} tanh(\frac{1}{2}L_{j'\to i}^{k-1}))$ **After adding the box to simplify the formula:** $L_{i\to j}^k = \boxplus_{j'\in N(i)-\{j\}} L_{j'\to i}^{k-1}$ among them, $(a,b) = \mathcal{B}(a,b) = 2\tanh^{-1}(\tanh(a/2)\tanh(b/2))$ The formula is recursively multiplied. Box-plus can be used as a lookup table for calculations. And you can make special arrangements for the CN node calculation	This step is different from the original SP algorithm, continuous density evolution derivation, box-plus operation $m = \mathcal{B}(m_1, m_2) = m_1 \boxplus m_2$ The probability distribution function can be expressed as: $P_m[k] = \sum_{(i,j):k\Delta=\mathcal{B}(i\Delta,j\Delta)} P_{m_1}[i]P_{m_2}[j]$ among them Δ To quantify the step size. The PMF operation can be written as a box addition operation of the PMF: $P_m = P_{m_1} \boxplus P_{m_2}$ Then the continuous box addition operation on the left can be written as the PMF box plus index. $P_c^k = (P_v^{k-1})^{\boxplus(d_c-1)}$ Bring the convolution in the VN update to the above

order to reduce the number of operations and reduce the complexity (please check the literature yourself, tired, and do not shoot).	formula to get: $P_c^k = (p_c^0 * [p_c^{k-1}]^{*(d_v-1)})^{\boxplus(d_c-1)}$ At this time, with continuous density evolution derivation P_v^kDifferent formulas, just askP_c^kThat's enough, because under the assumption that only +1 is sent, the error probability integration results of the two are the same.
d. LLR summation: $L_j^{total} = L_j^0 + \sum_{i \in N(j)} L_{i \to j}^{k-1}$	Also perform ber calculations, but discrete $\sum_{i<0} p_c^k[i](x)] \leq ber^*$ Decide whether to continue iterating or adjust channel parametersα
e. Judgment: • $\hat{v} = 1, if L_j^{total} < 0; \hat{v} = 0, other.$ • if $\hat{v}H^T = 0$, exit and output; otherwise k++ and goto **step b**.	

Let's see how the code is implemented (subfunctions)

1. chan_ini=chan_mess(ext,deta) ;

According to the quantization method ext, the standard deviation deta of Gaussian noise is used to calculate the quantized LLR PDF, and the output can be seen by plot(chan_ini).

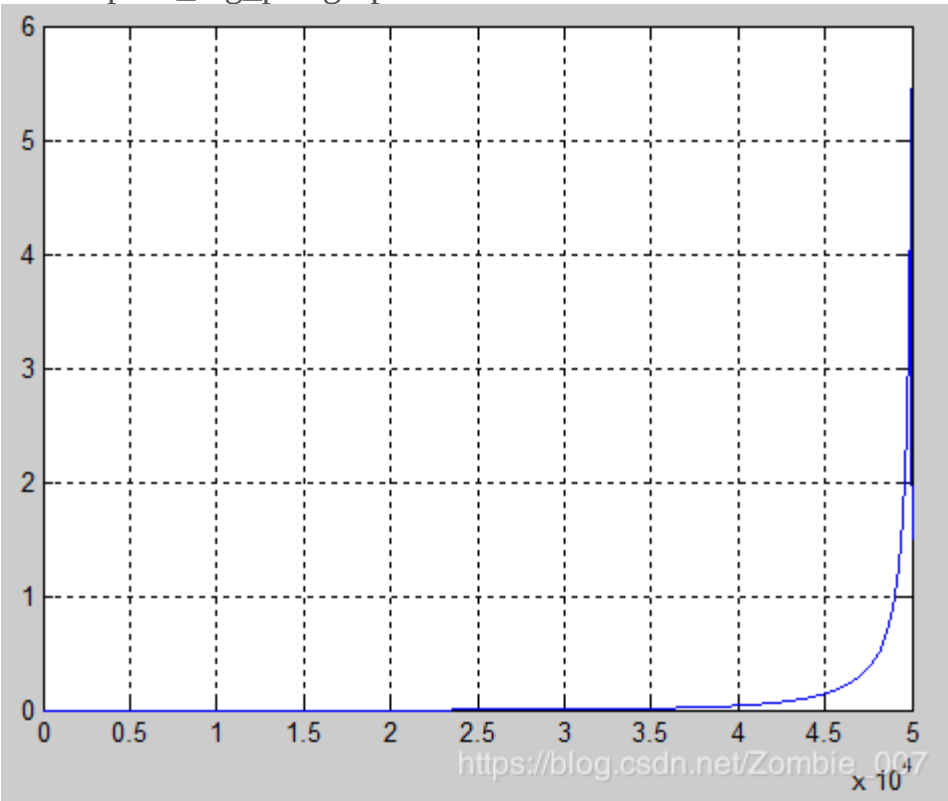
2. y = new_xchk(ext, z, dc-1, mapping);

Check the node update process, focusing on the key points.

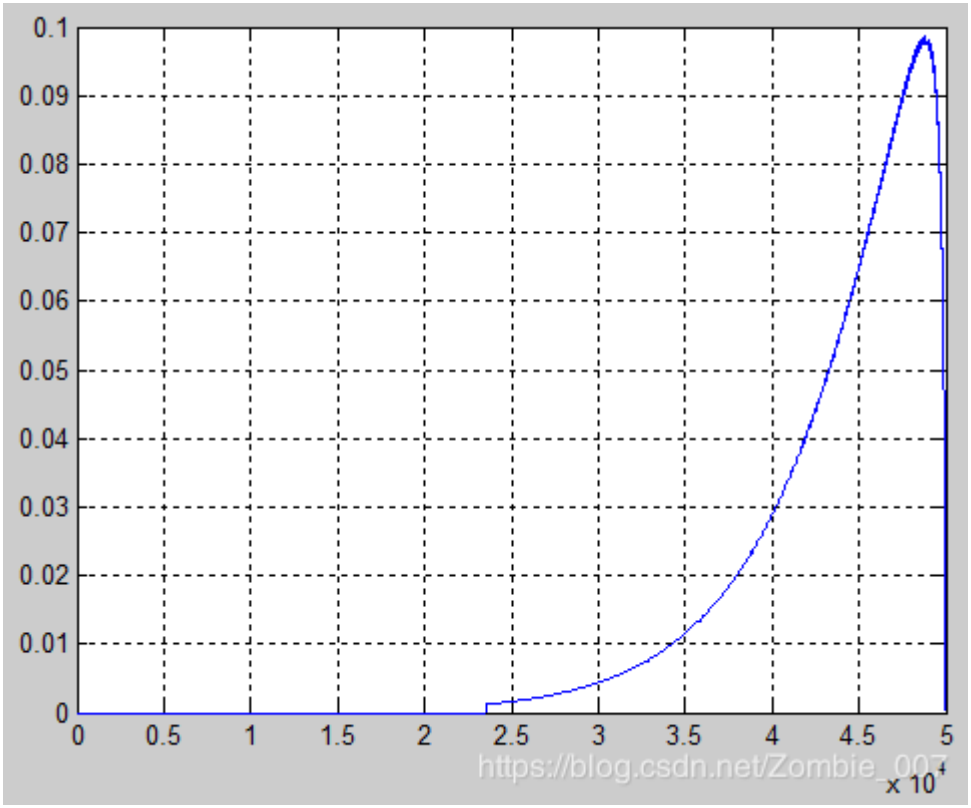
The main method is to map the discrete PDF of the LLR (through the look-up table of the limiting range) to$\phi(x)$, log (tanh (x / 2)) function, then fft convolution operation, and finally changed back to the LLR PDF. Therefore, the calculation of discrete density evolution is performed based on the derivation of the original density evolution in Table 1 introduced earlier.

2.1 [f_log_pos,f_log_neg,excess] = new_wrap2flog(ext, f_ext, mapping);

in conclusion The function of this function is to convert the LLR into a form of log(tanh(x/2)), keeping the symbol information unchanged.
Wrap: standard LLR
Flog: log(tanh(x/2)) function, ie$\phi(x)$At this point, it is basically possible to determine that the function should be doing one.Function lookup table,andPrimitive Gaussian density derivation method。
The output f_log_pos graphic:



The output f_log_neg graphic:



Mapping: consistent with the form of ext, [base increment length]

T_mapping: use mapping to generate the entire function abscissa axis

Foo: from 0.01 to 30, except for the first and last tails (foo(1,1) and foo(2,3000)), the intermediate step is 0.005

foo

2x3000 double

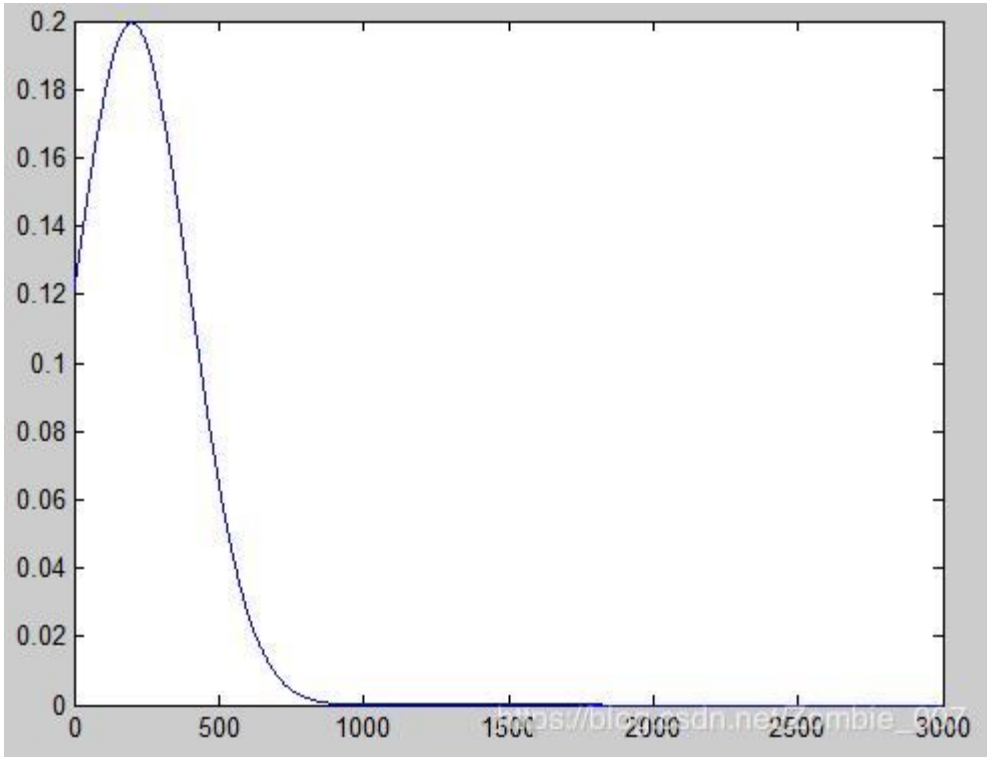
	1	2	3	4	5	6	7
1	0.0100	0.0150	0.0250	0.0350	0.0450	0.0550	0.0650
2	0.0150	0.0250	0.0350	0.0450	0.0550	0.0650	0.0750

2995	2996	2997	2998	2999	3000
29.9450	29.9550	29.9650	29.9750	29.9850	29.9950
29.9550	29.9650	29.9750	29.9850	29.9950	30.0050

The two values in the foo column are a group, and the subsequent up/down overflow judgments work.
Then, according to the quantized bin given by foo, the discrete function value of log(tanh(foo/2)) is obtained.
Bin_convert = log(tanh(foo/2)); Then limit it, ie find the values of overflow and underflow uf.

Calculation $coff = \frac{2e^{t\_mapping}}{1 - e^{2t\_mapping}}$

bar :



Take half of the LLR's PDF.

2.1.1 [f_log_pos,ofl,ufl] = xconvert(wrap, bar, bin_convert, mapping, coeff);
Limit the value of the flog function based on the given LLR quantization range
3. z = new_xvar(chan, y, dv-1, ext, ext);
Update of the variable node. This step mainly uses the fwt and iftr functions that come with matlab to realize the operation between discrete probability mass functions.