Lecture 23

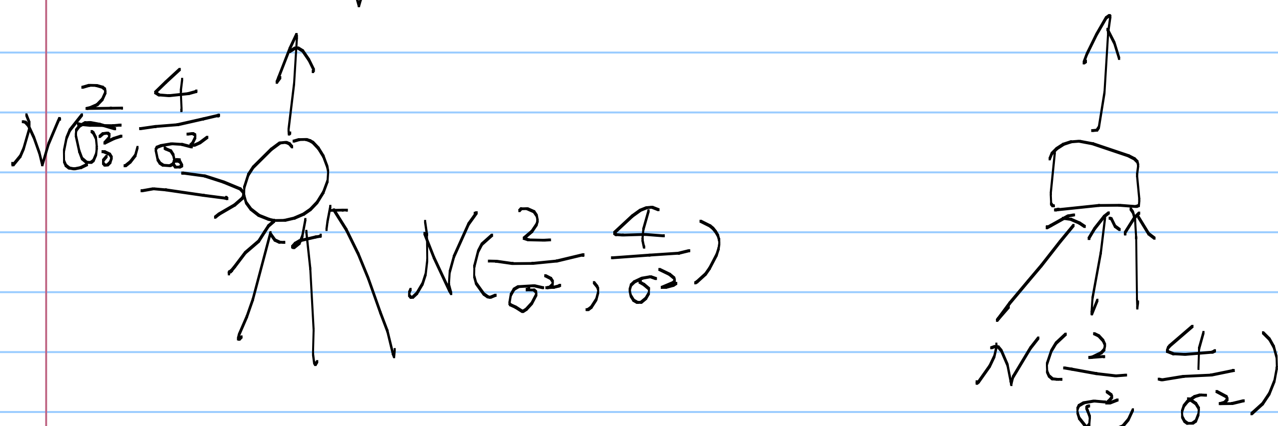
Note Title 4/4/2012

* EXIT Chart for LDPC + Gsn channel.

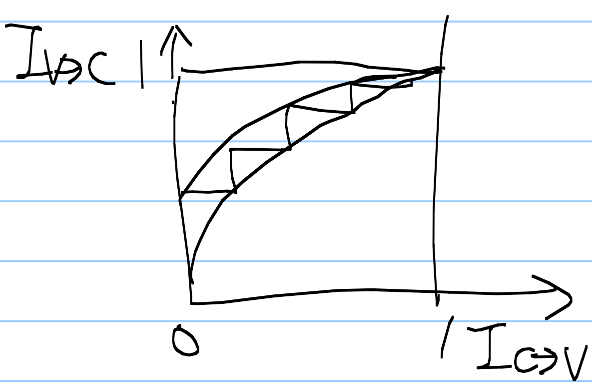
For any channel other than BECs, EXIT chart is only a good approximation tool.

* $I(X;Y) = \int_{-\infty}^{\infty} \log\left(\frac{2e^m}{e^m+1}\right) dP_{m|X=0}$

* Gsn approximation

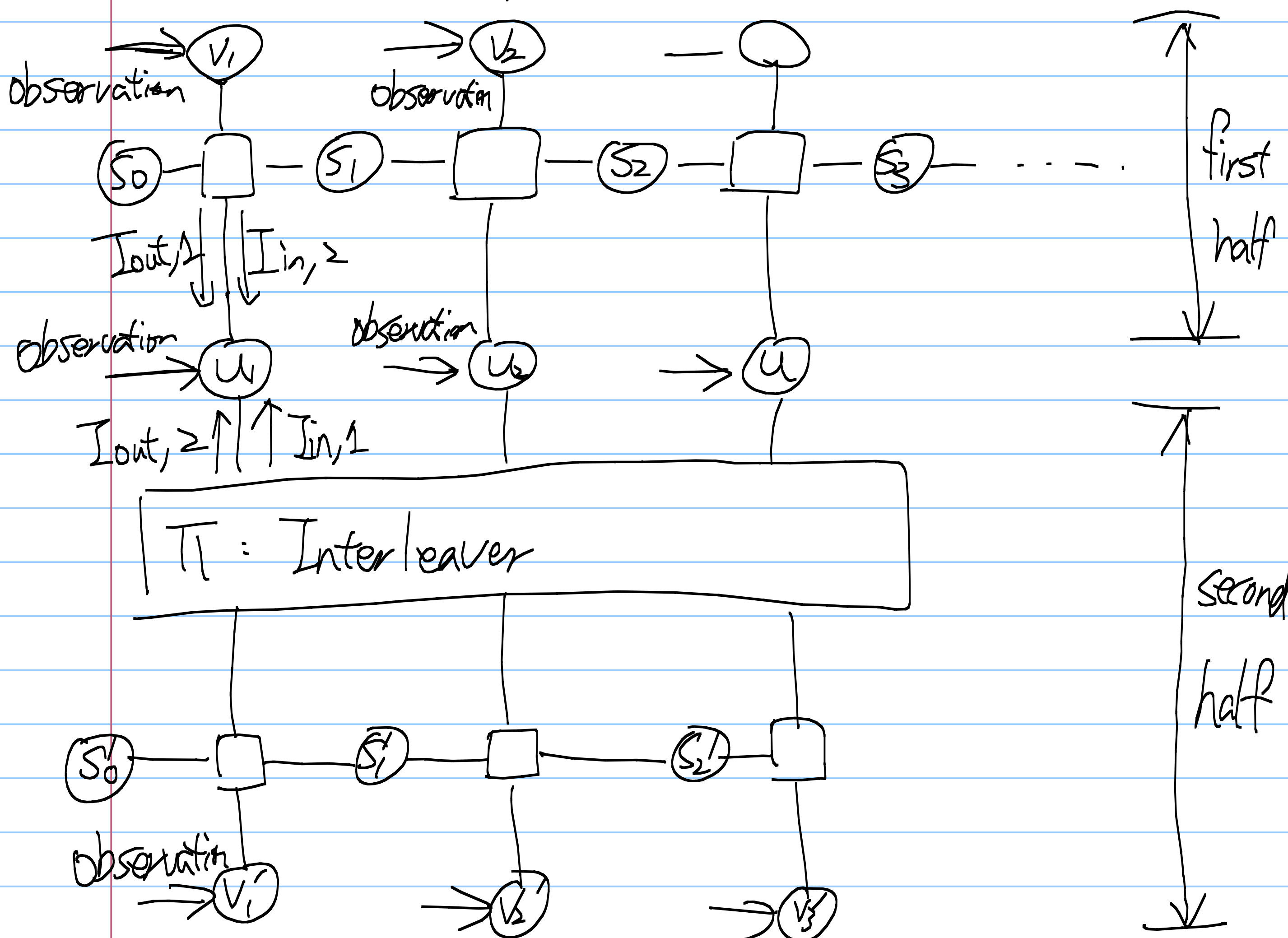


*



* In essence, it is no different than a Gsn approximation of the density by matching the mutual information value

EXIT Chart for turbo codes with Gsn channel. (The very first application of the EXIT Chart analysis when it was first derived.)

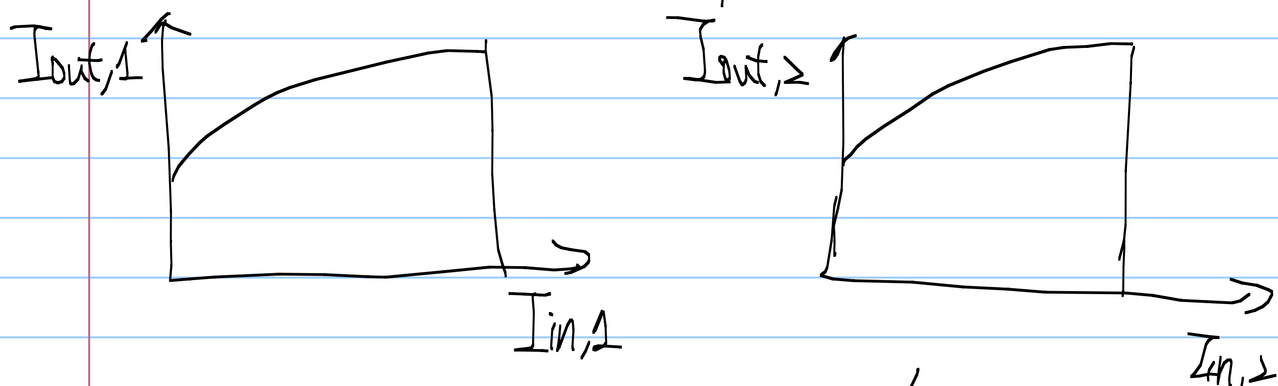


EXIT Chart: ① Separating the turbo code into two components

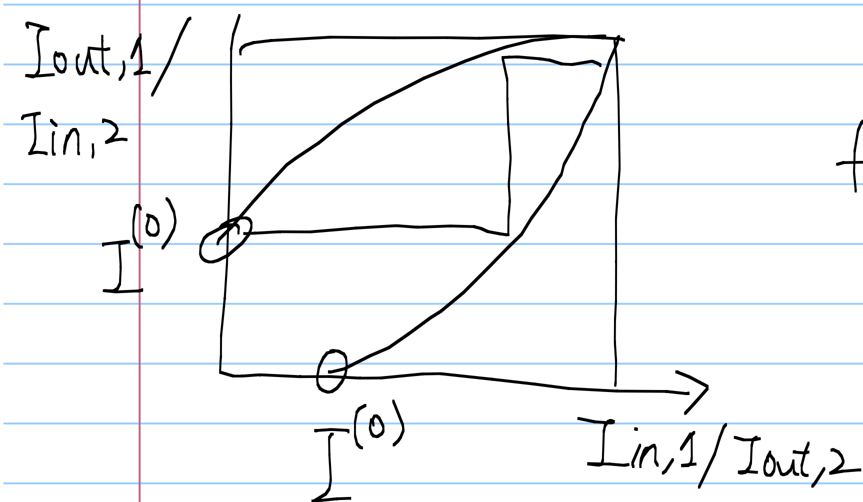
② Gsn approximation with matched mutual information

③ Monte-Carlo-simulation-based computation

① With the right separation, the EXIT Chart curve will plot



flipped



Remark:

① If both RSCs are identical, then the two curves are symmetric images

② The quanties $I_{in,1}$, $I_{out,1}$, $I_{in,2}$, $I_{out,2}$ are indeed the "extrinsic information", which gives the name of the EXIT Chart analysis.

③ Again, the open tunnel argument holds.

How to compute the EXIT curves?

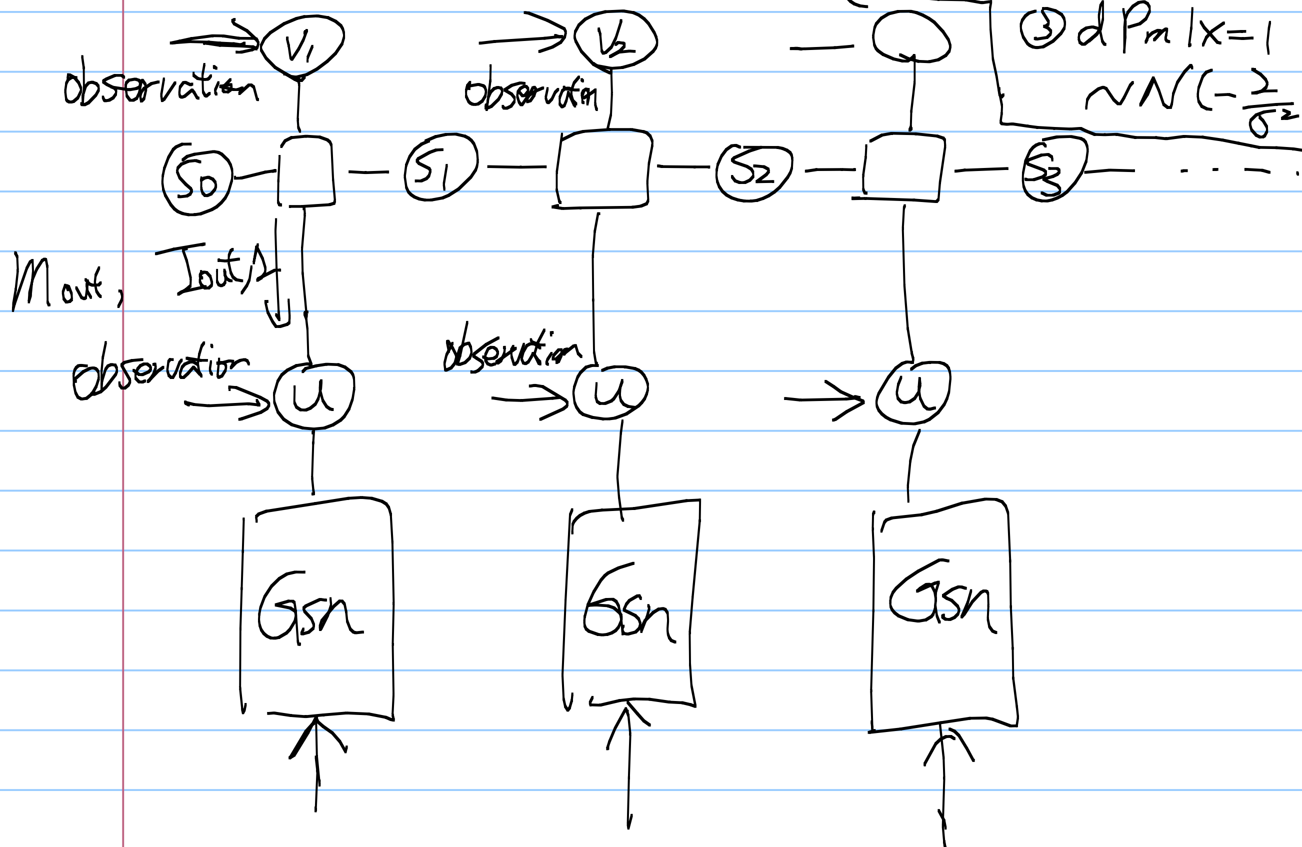
Step 0: Randomly choose one codeword.

Ans: Step 1: Given the scalar $I_{in,1}$, replace the messages by messages coming from independent binary-input Gsn channel with matched $I_{in,1}$.

① $dP_{m|x=0} \sim N\left(\frac{2}{\sigma^2}, \frac{4}{\sigma^2}\right)$

② $I_{in,1} = \int \log\left(\frac{2e^m}{e^m+1}\right) dP_{m|x=0}$,

③ $dP_{m|x=1} \sim N\left(-\frac{2}{\sigma^2}, \frac{4}{\sigma^2}\right)$



Even though the original messages are <u>neither Gsn</u>, nor <u>independent</u>.

Step 2: Run DE to compute the density of M_{out}. Then use

$$I(X;Y) = \int_{m=-\infty}^{\infty} \log \frac{2}{1+e^{-m}} \, dP_{m|x=0}$$

to compute $I_{out,1}$, where $P_{m|x=0}$ is the marginal density of m

* However, density evolution is too complicated for turbo codes.

⇒ We use Monte-Carlo simulation instead

Step 2.0: Randomly choose the info bits & encode it by the turbo encoder.

Step 2.1 generate the extrinsic messages from i.i.d Gsn w. $\boxed{\text{matched } I_{in,2}.}$

Assume $\vec{X}$ is transmitted, and for any given σ.

∴ Each extrinsic incoming message is chosen with distribution $N\left(\frac{2\cdot(-1)^{x_i}}{\sigma^2}, \frac{4}{\sigma^2}\right)$ depending on

on whether the underlying bit is 0 or 1 respectively. This naturally circumvents the potential non-symmetry of the system

Step 2.2: Generate the observation messages from i.i.d Gsn with $\sigma^{(0)}$. $\leftarrow$ The actual noise of the observation channel

Again messages $\sim \mathcal{N}\left(\frac{2(-1)^{y_j}}{(\sigma^{(0)})^2}, \frac{4}{(\sigma^{(0)})^2}\right)$

where y_j is the j-th parity bit.

Step 2.3: Fix all the generated messages, run BCJR & compute the <u>extrinsic</u> LLR messages ~~of~~ <u>m_{out}</u>. This is a deterministic step

Step 2.4: Use Monte-Carlo simulation to estimate the mutual information (of the marginal distribution)

$$I(X;Y) = P(X=0)\int_{m=-\infty}^{\infty} \log \frac{2e^m}{e^m+1}\, dP_{m|X=0}$$

$$+P(X=1)\int_{m=-\infty}^{\infty} \log \frac{2}{e^m+1}\, dP_{m|X=1}$$

$$\approx \frac{\# x_i=0}{k} \cdot \left(\frac{1}{\# x_i=0} \sum_{\forall i,\, x_i=0} \log \frac{2e^{m_i}}{e^{m}+1}\right) +$$

$$\frac{\# x_i=1}{k}\left(\frac{1}{\# x_i=1} \sum_{\forall i\, x_i=1} \log \frac{2}{e^{m}+1}\right)$$

If we have k $= 10000$ information bits, then one round of BCJR computation gives us $k = 10{,}000$ Sample points of the marginal distribution $\Rightarrow$ precision is high. However, sometimes we still need to repeat Steps 2.0 – 2.4 to achieve the desired precision.

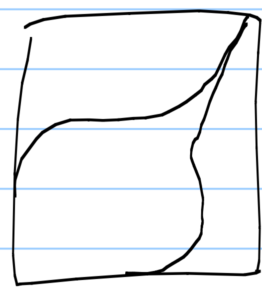
The computed output $I_{out,1}$ vs. the the input $I_{in,1}$ gives one point of the EXIT curve. Choose different σ, we can have the desired EXIT curve.

* The computation for $I_{out,2}$ vs. $I_{in,2}$ can be carried out in a similar way.

* The benefits of EXIT chart analysis on turbo codes (in addition to it visualization benefits.)

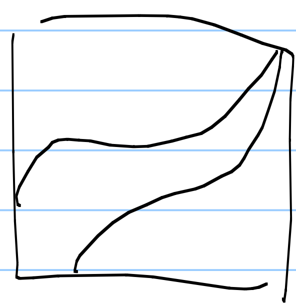
① The design of two RSCs can be conducted independently.

In general, using two identical RSCs has (slightly worse) performance

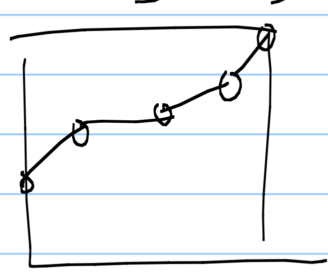


(hard to match the curves)

With different RSCs, we might have



② The computation of the EXIT curves is very efficient.



We do not need too many rounds of Monte-Carlo simulation as each round gives us "k" samples of the marginals. When k is large, we have good accuracy using a small # of rounds when computing

$$I(X;Y) \approx \frac{\#\, x_i=0}{k} \cdot \left(\frac{1}{\#\, x_i=0} \sum_{\forall i,\, x_i=0} \log \frac{2e^{m_i}}{e^{m}+1} \right) + \frac{\#\, x_i=1}{k} \left(\frac{1}{\#\, x_i=1} \sum_{\forall i\, x_i=1} \log \frac{2}{e^{m}+1} \right)$$

& we only need a few points in the EXIT curves

③ EXIT Chart is thus a clever combination of the asymptotic, cycle-free assumption, iterative decoding behavior, $\underbrace{\text{decoding behavior,}}_{\text{Gsn approximation}}$ & Monte-Carlo simulation

See the extracted page from tenBrink's paper.

EXIT Chart

Summary: ① Arbitiarily encode your information

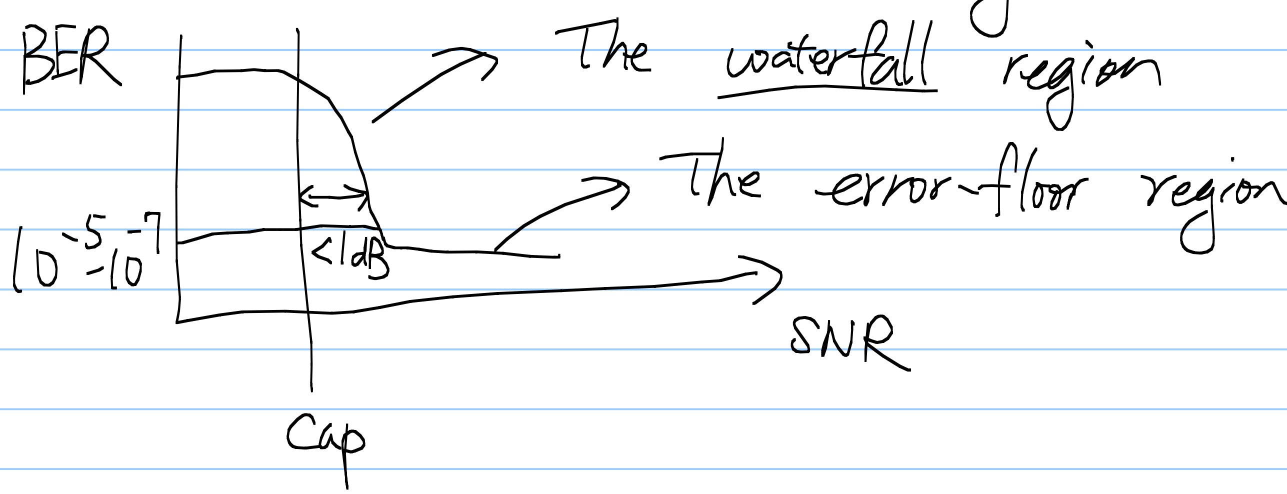
② Compute its extrinsic info by carefully distinguishing the X values & the conditional distibutions

③ Change different input I_{in} & use Gsn approixmation + Monte-Carlo to generate the results.

④ It holds for any decoder as we simply use the right decoder to generate M_{out} from the MC simulation

* We will wrap up our discussion on capacity-approaching error correction codes & move to network coding, Another exciting combination of graph-based studies & coding

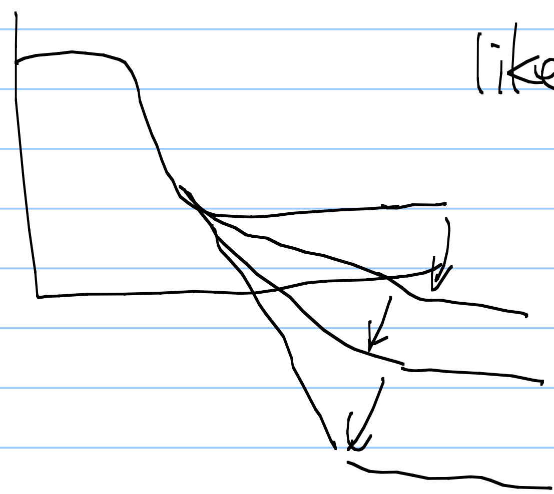
* LDPC codes & turbo codes are extremely efficient. with codeword length $10^4 - 10^5$, one can easily get a BER curve like the following



What's next? In addition to more efficient implementation, there are at least three directions that worth investigation.

① Codes that have performance arbitrarily close to the capacity. Example: can we derive codes that has the threshold within 0.00001 dB of the cap?

② Bend down the error-floor (lower the error floor) for some applications



like satellite comm.
optical comm.
storage comm.

③ Design codes that have high performance even for smaller codeword length. $10^4 - 10^5$ is too long for commercial packet-based applications. $10^3 - 10^4$ is more practical. Can we maintain the performance with short codes?
For LDPC codes, it can be done from

the new structure & design of the interleaver. It's harder for turbo code design.

* For the following, we briefly discuss ①, ② and some existing results.

* BEC Capacity - Approaching irregular LDPC code ensemble.

Remark: Even though EXIT Chart & DE have led to many LDPC code ensembles that have close-to-cap performance. One remaining question is whether we can be "arbitrarily" close to the capacity. Or are we bounded away strictly from the capacity.

Our goal: Construct a series of (λ, ρ) ensembles, denoted by (λ^m, ρ^m) for $m = 1, \cdots, \infty$. s.t. all (λ^m, ρ^m) have the same rate R. & the BEC threshold $\varepsilon^m \nearrow 1-R$ when $m \to \infty$.

(Oswald, Shokrollahi 02)

The derivation there is complicated.

In this lecture, let's consider an alternative goal of deriving the cap-approaching LDPC codes.

Goal: Construct a series of $(\lambda^{(m)}, \rho^{(m)})$ LDPC codes, such that they can all decode BEC of erasure prob ε & $R^{(m)} \searrow 1-\varepsilon$ when $m \to \infty$

$\forall$ Note: each λ^m, ρ^m can only have finite max degrees