Optimizing Flash based Storage Systems

UCLA

- Lifetime
- Reliability
- Latency
- Throughput

Projects

- Reliability/Latency/Throughput: Error Correction Code (ECC) Parallelization and Incremental Redundancy
- Lifetime: Channel Estimation and Write Voltage Optimization

Projects

- Reliability/Latency/Throughput: Error Correction Code (ECC) Parallelization and Incremental Redundancy
- Lifetime: Channel Estimation and Write Voltage Optimization

UCLA

Approaching Capacity Using Incremental Redundancy Without Feedback

Haobo Wang, Sudarsan V. S. Ranganathan, and Richard Wesel

Motivation/Application for Storage

- (Latency/Throughput) How to accelerate ECC for Flash?

Use parallel short codes to replace a long codeword!

- (Reliability) How to recover the data from a failed codeword more efficiently?

Lower the rate of the codeword adaptively!

Outline

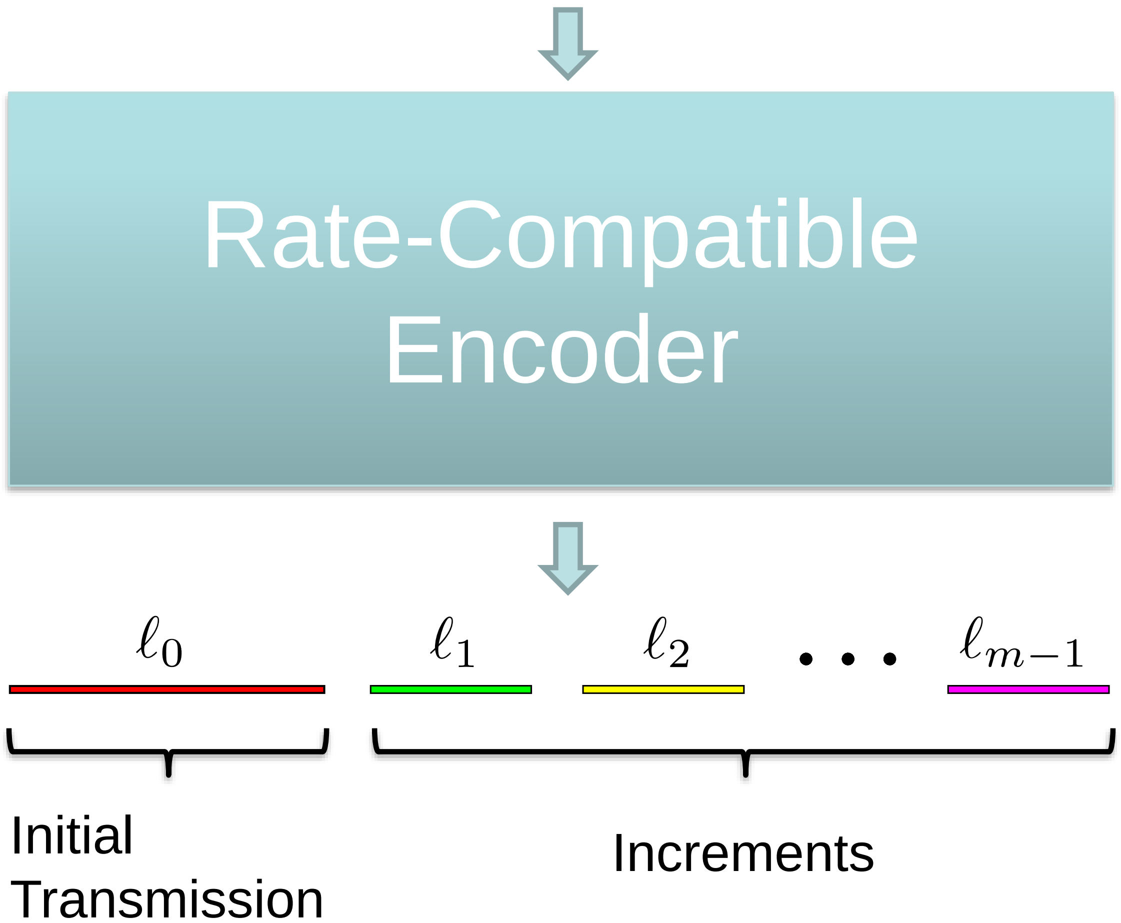
- **Previous work**: Approaching Capacity with Short Blocklengths using Incremental Redundancy and ACK/NACK Feedback [Vakilinia et al. TCOM 2016]
- **New Idea**: Approaching Capacity using Many Short-blocklength Codes with Incremental Redundancy in Parallel *Without Feedback*
  - Concept
  - Design methods and design examples

Outline

- **Previous work**: Approaching Capacity with Short Blocklengths using Incremental Redundancy and ACK/NACK Feedback [Vakilinia et al. TCOM 2016]
- **New Idea**: Approaching Capacity using Many Short-blocklength Codes with Incremental Redundancy in Parallel *Without Feedback*
  - Concept
  - Design methods and design examples

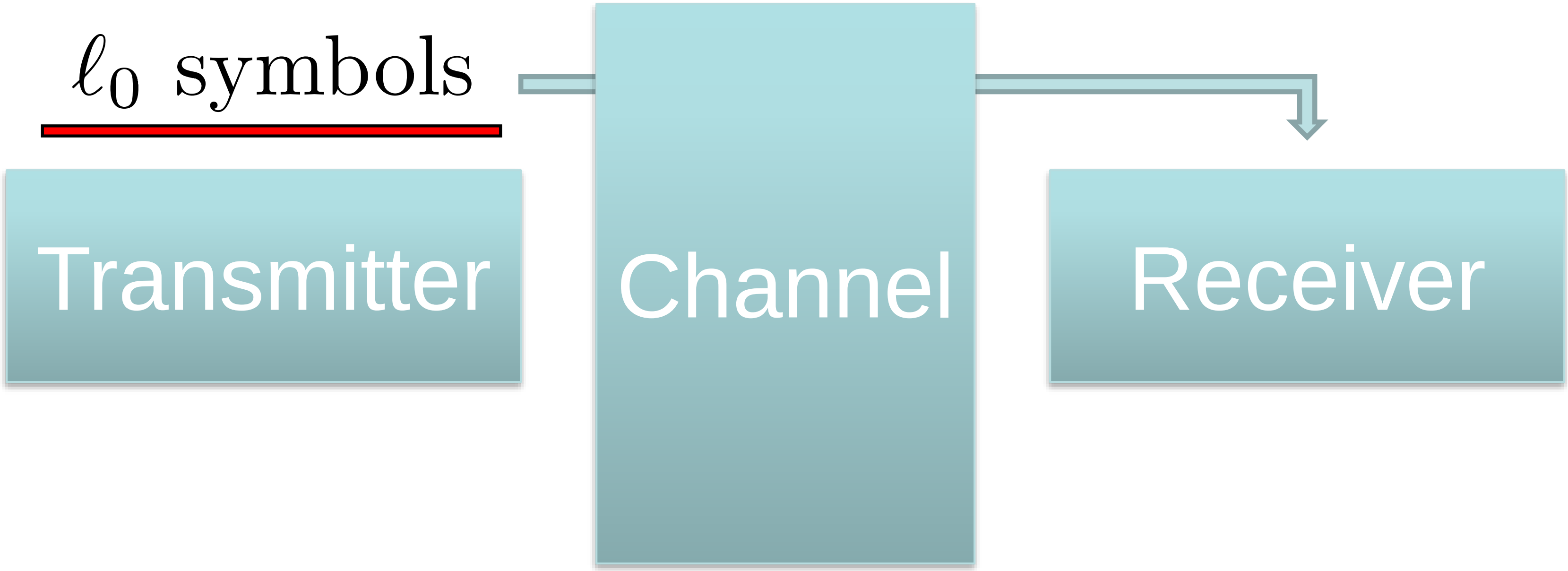
A Rate-compatible Encoder

k bits of user information

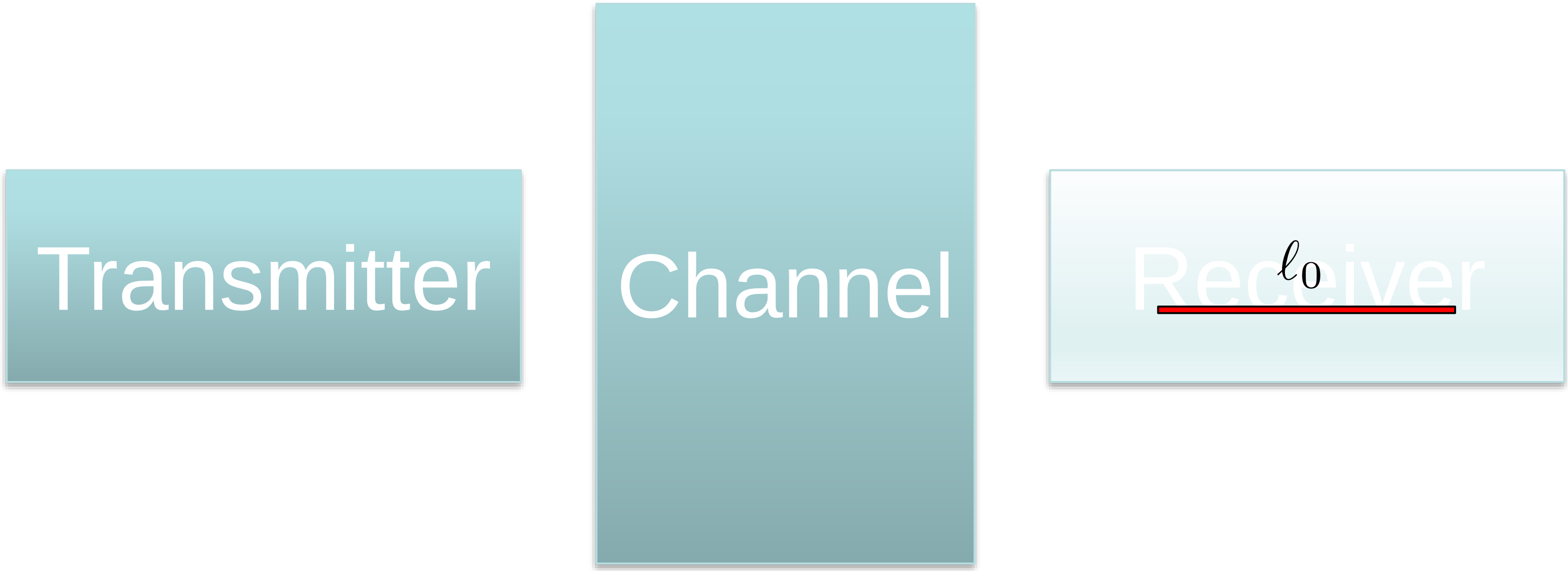


UCLA

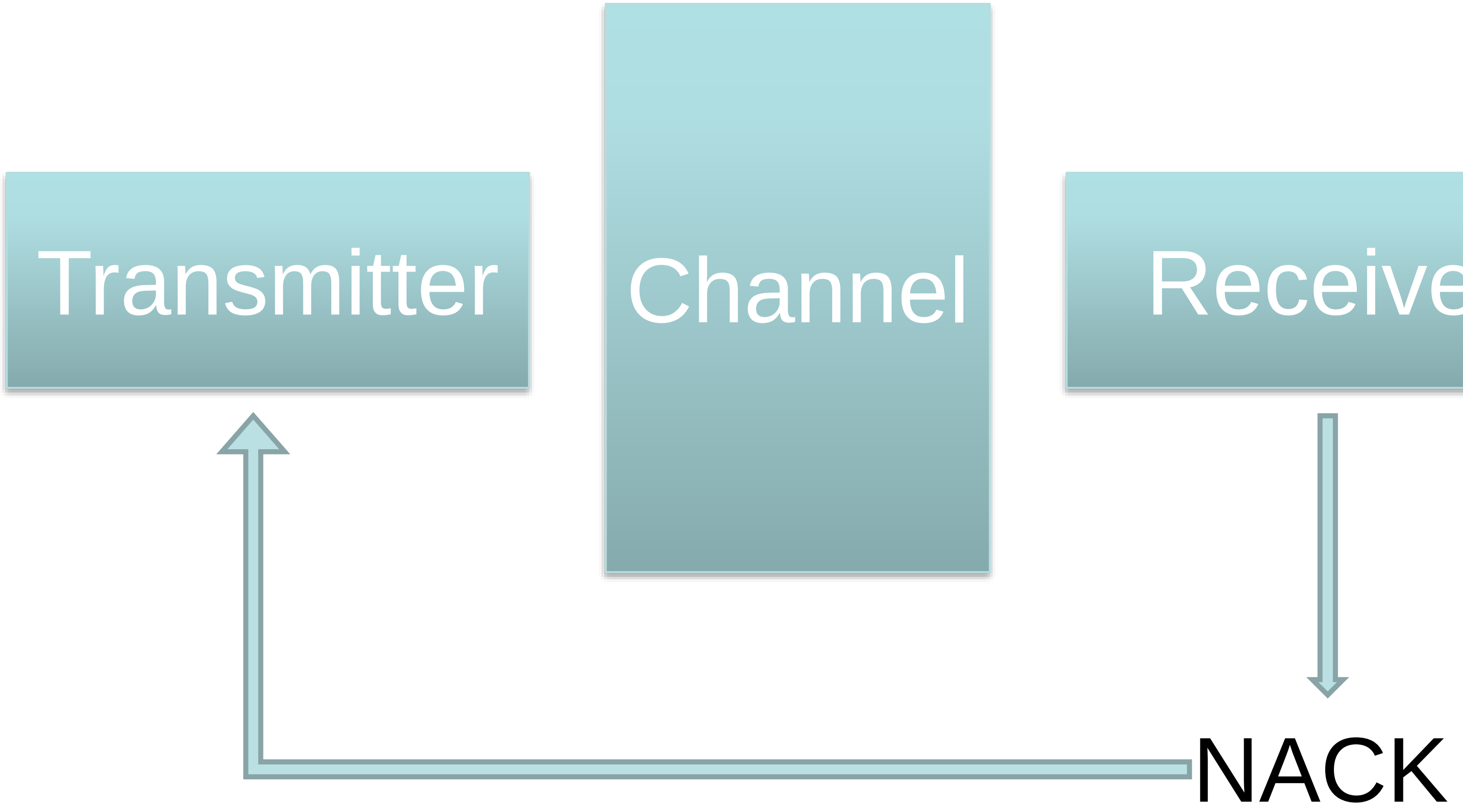
Incremental Redundancy with Feedback



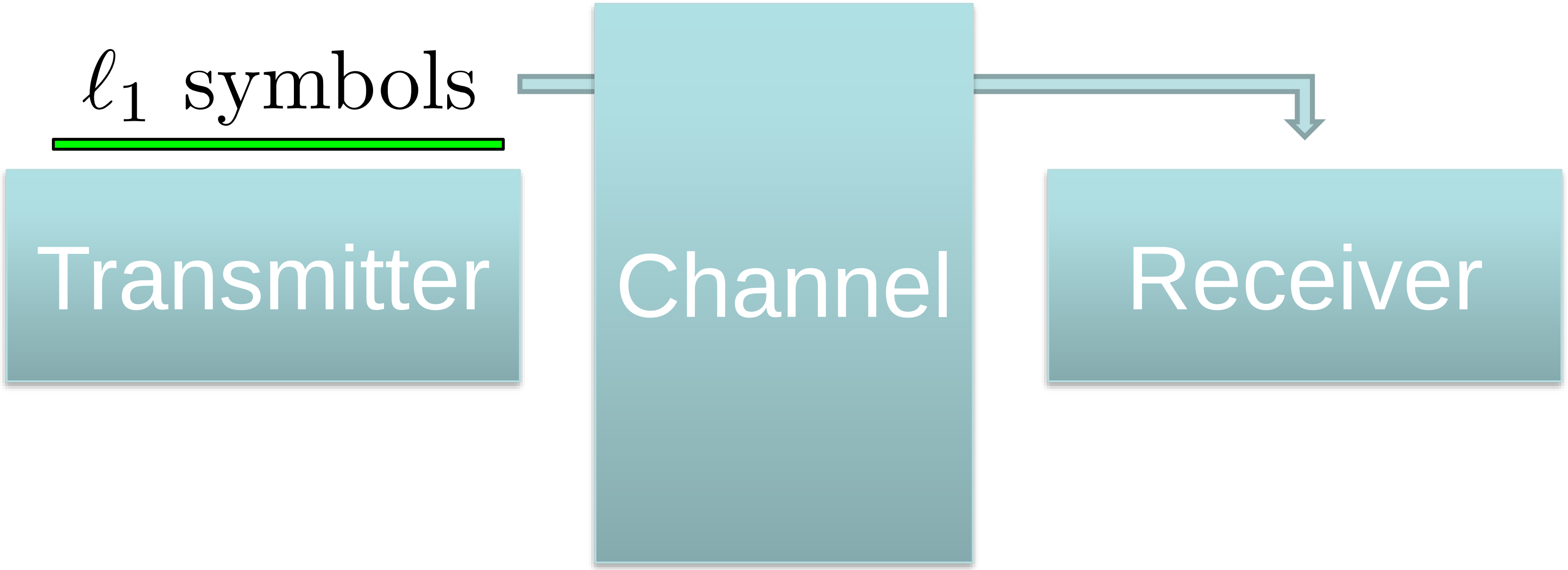
Incremental Redundancy with Feedback



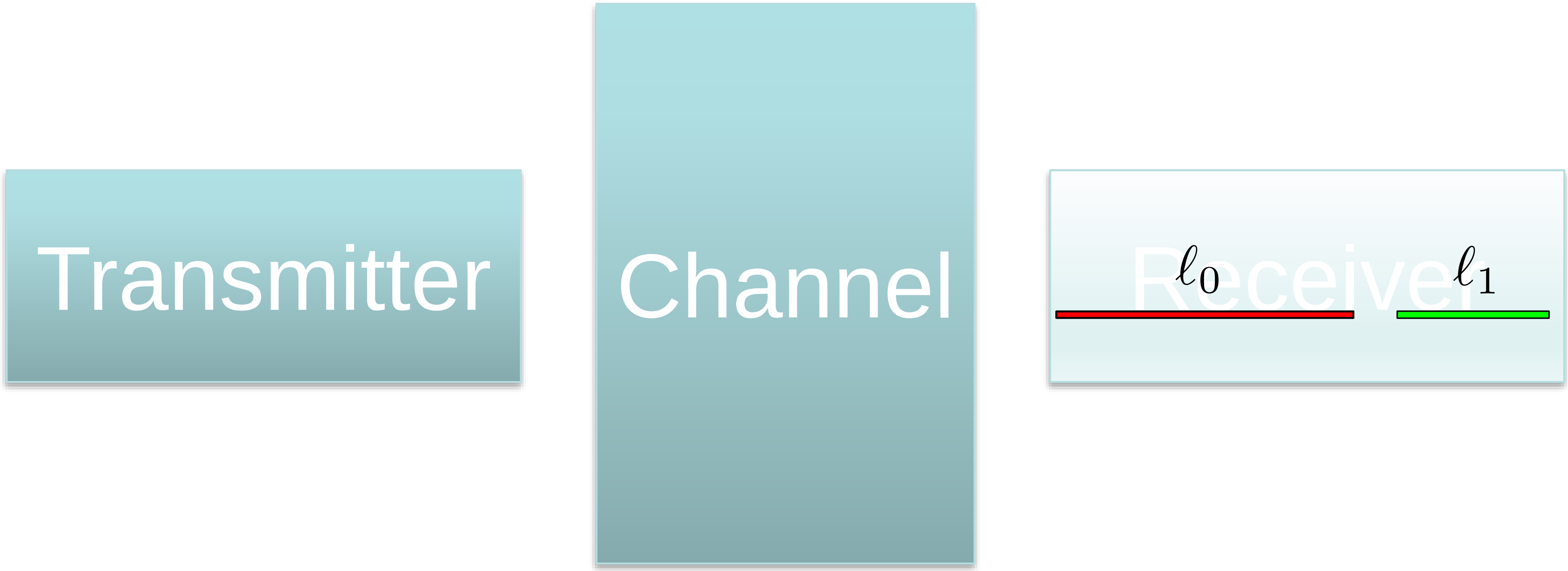
Incremental Redundancy with Feedback



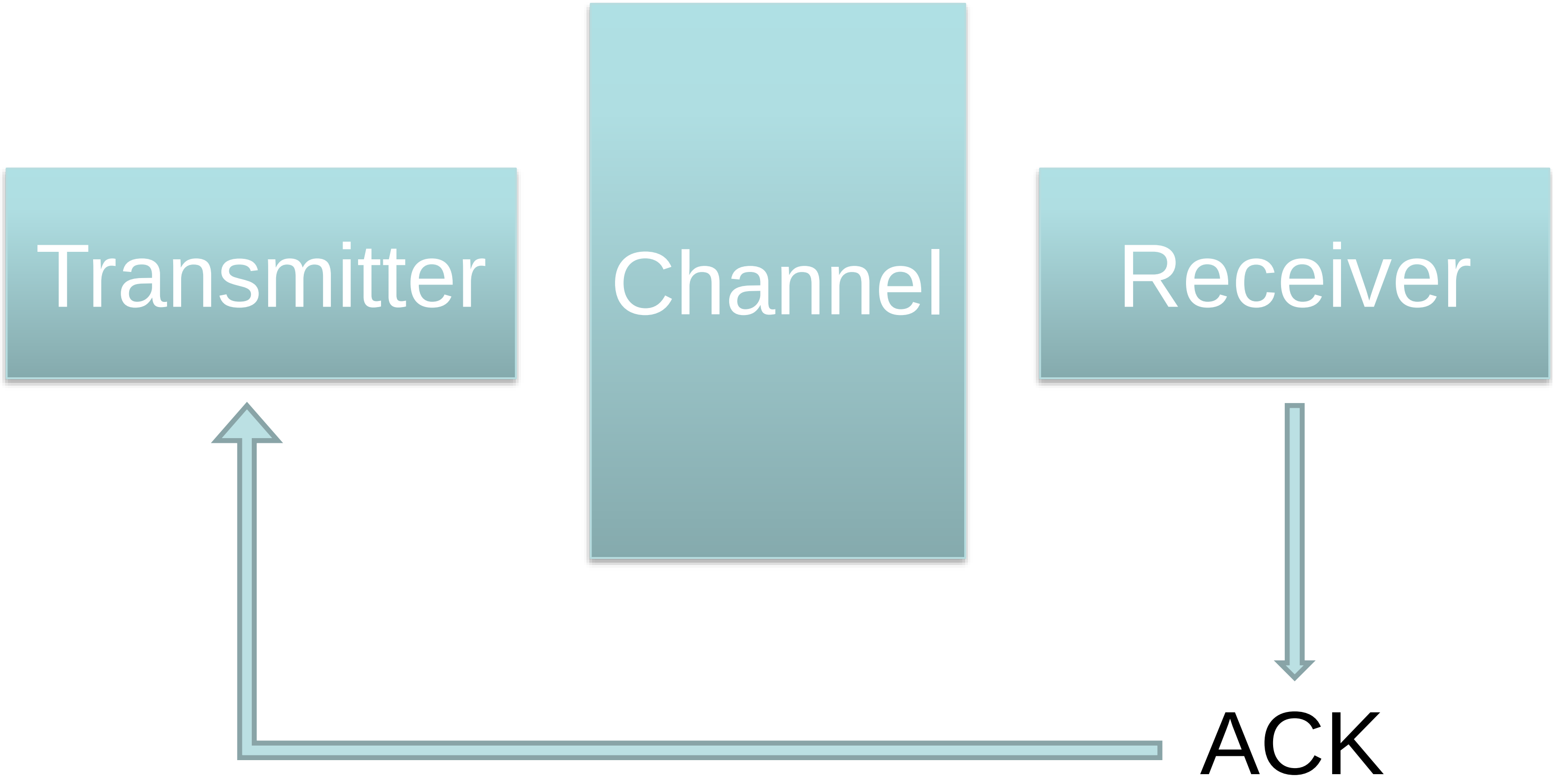
Incremental Redundancy with Feedback



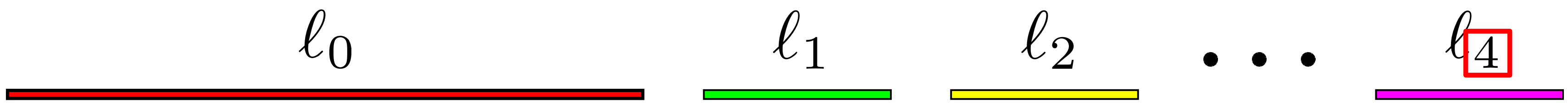
Incremental Redundancy with Feedback



Incremental Redundancy with Feedback



Variable-length Code Parameter in This Work



$$\ell_1 = \ell_2 = \ell_3 = \ell_4 = \ell_\Delta$$

$$R_t^{(FB)} = \frac{k(1 - \epsilon_{FB})}{l_0 + \beta_{FB}\ell_\Delta}$$

In this presentation, we will compare our feedback-free design against corresponding constant-increment-size feedback codes.

Keep in mind

- In general, a VL error correction code (ECC) with feedback has a higher rate than a feedforward ECC at comparable block length.
- We want to approach the rate of the feedback scheme without feedback.

Outline

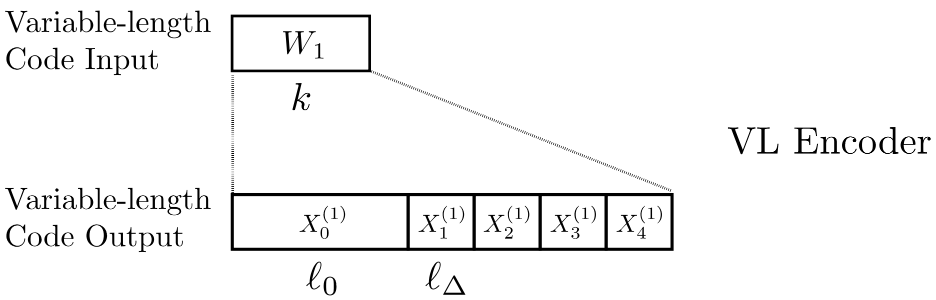
- **Previous work**: Approaching Capacity with Short Blocklengths using Incremental Redundancy and ACK/NACK Feedback

- **New Idea**: Approaching Capacity using Many Short-blocklength Codes with Incremental Redundancy in Parallel *Without Feedback*
  - Concept
  - Design methods and design examples

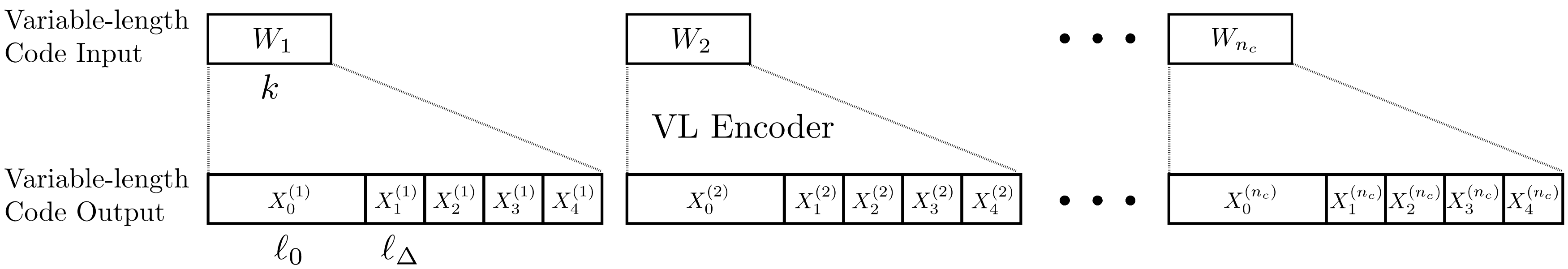
Principal Concept

- We use many VL codewords in parallel.
- Send the highest-rate part of each VL codeword. Some VL codewords need increments.
- From ergodicity, we know the total amount of redundancy needed by all the codewords.
- We use ***inter-frame coding [Zeineddine et al. JSAC 2016]*** to linearly encode the increments
  - Deliver exactly the right amount of redundancy for each VL codeword.
  - We expand the analysis to any point-to-point channel, and design actual codes.

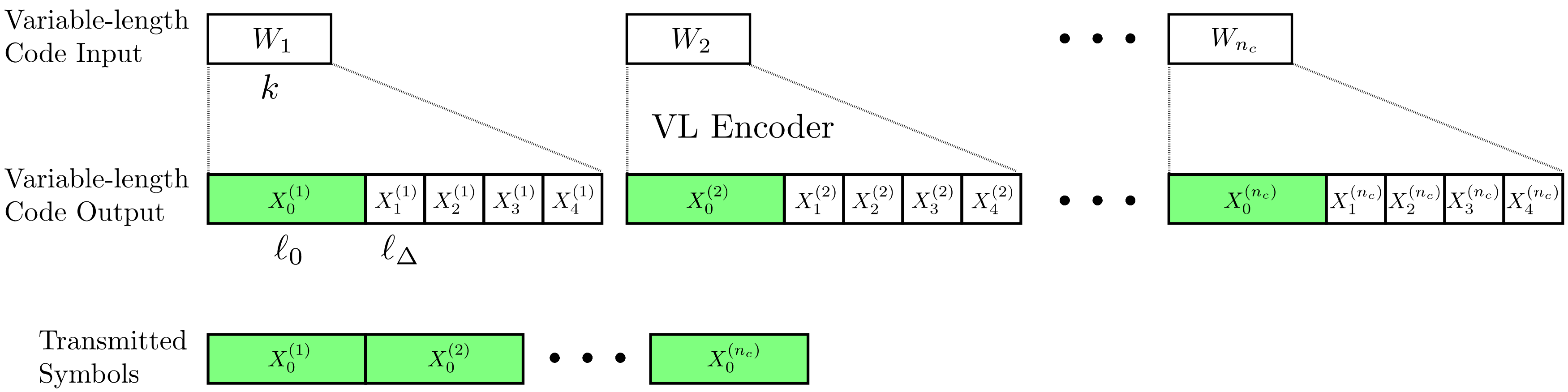
Inter-frame Code [Zeineddine et al. JSAC 2016]



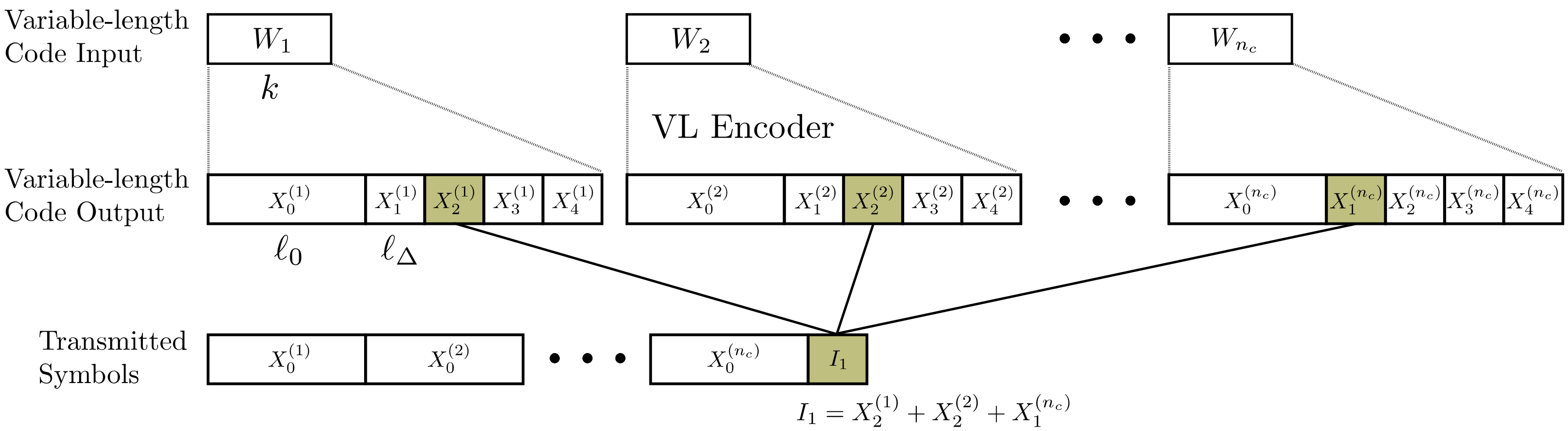
Inter-frame Code [Zeineddine et al. JSAC 2016]



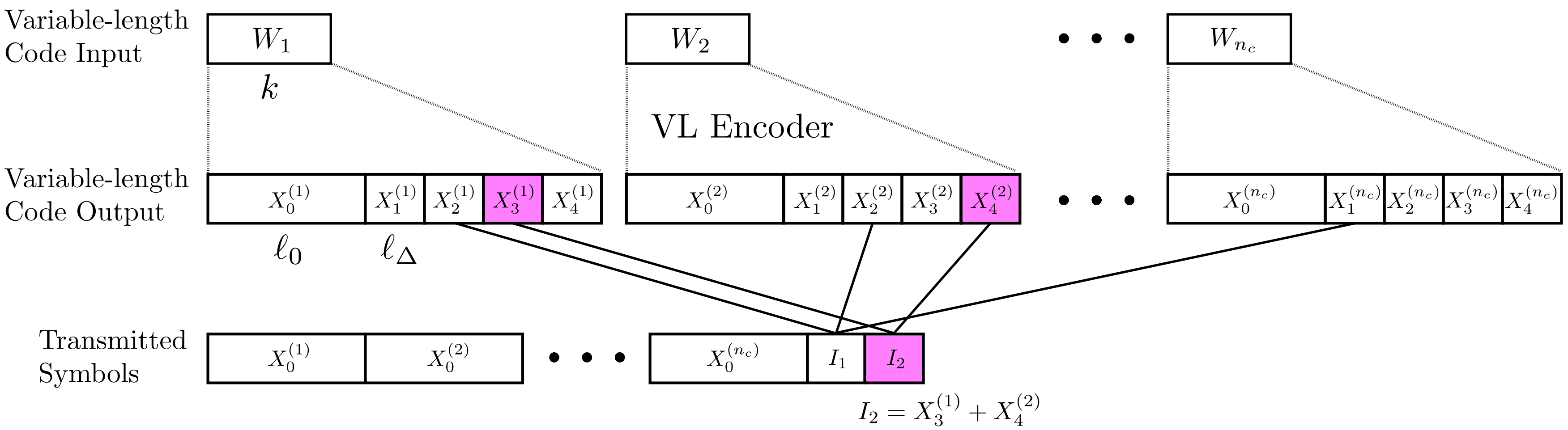
Inter-frame Code [Zeineddine et al. JSAC 2016]



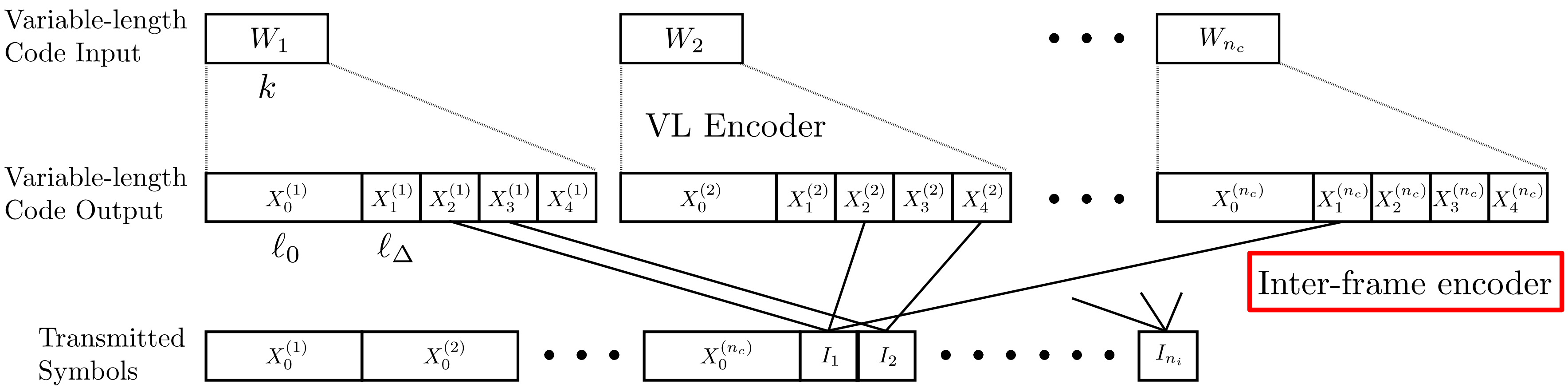
Inter-frame Code [Zeineddine et al. JSAC 2016]



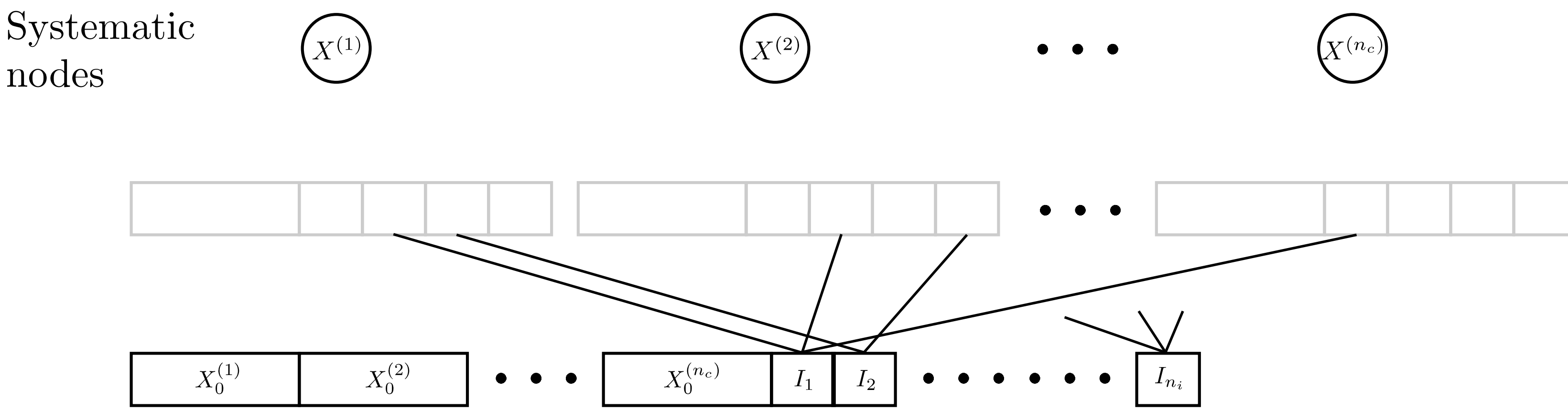
Inter-frame Code [Zeineddine et al. JSAC 2016]



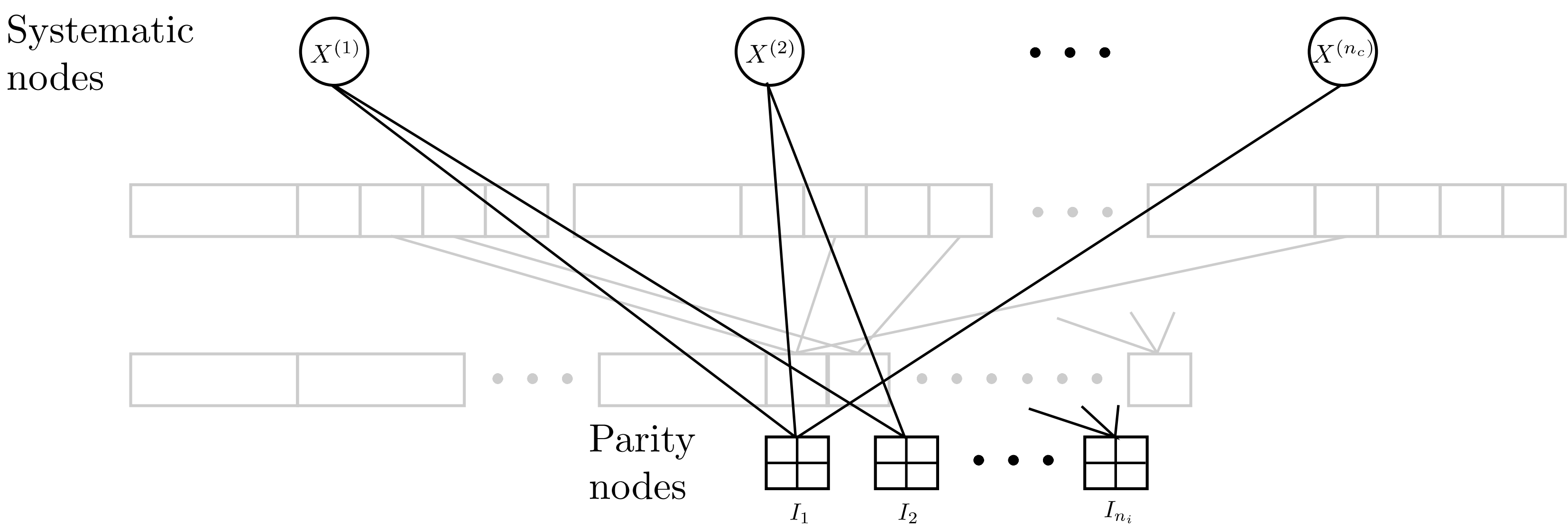
Inter-frame Code [Zeineddine et al. JSAC 2016]



Low Density Generator Matrix (LDGM) [Cheng et al. Allerton 1996] Code

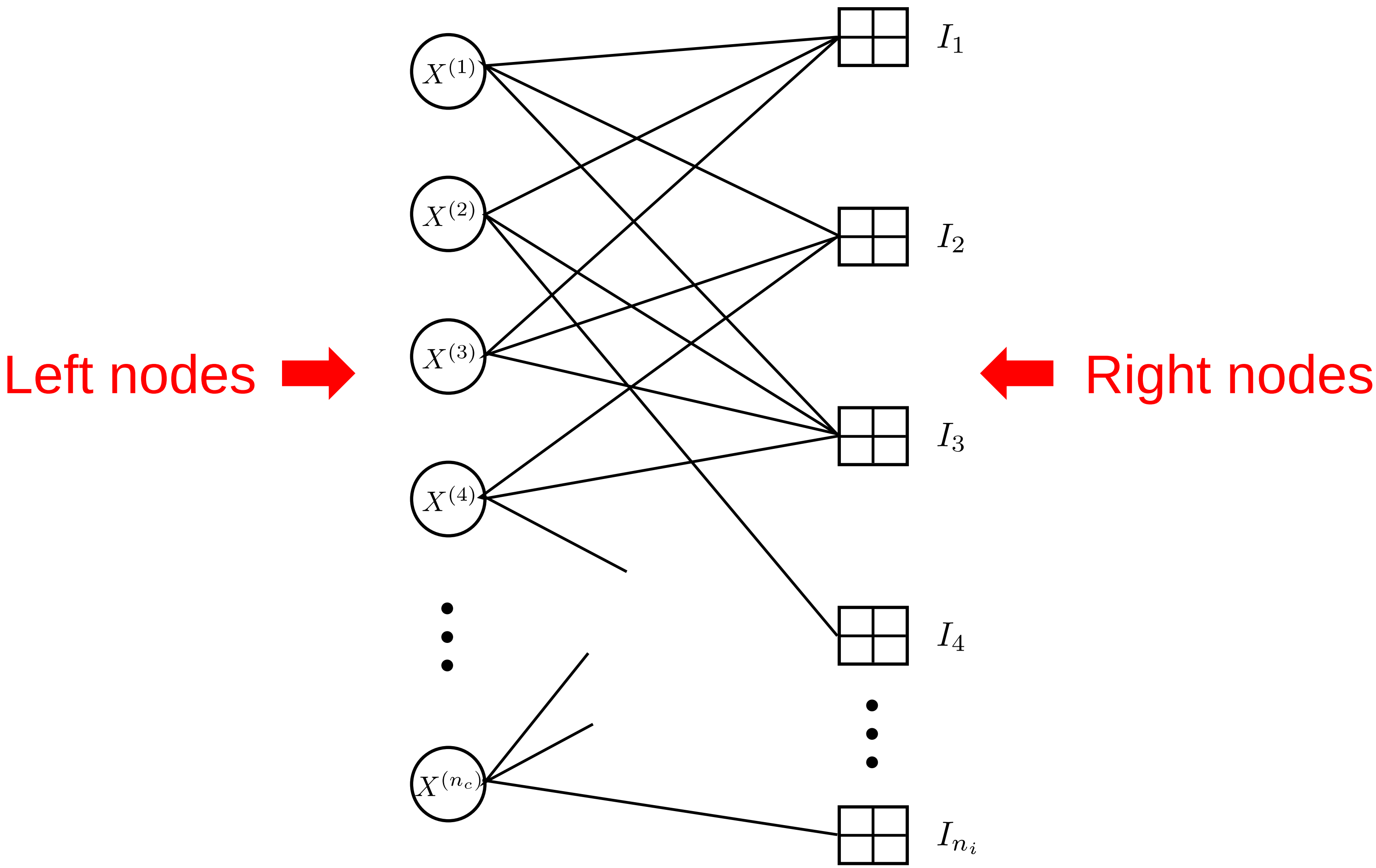


Low Density Generator Matrix (LDGM) [Cheng et al. Allerton 1996] Code

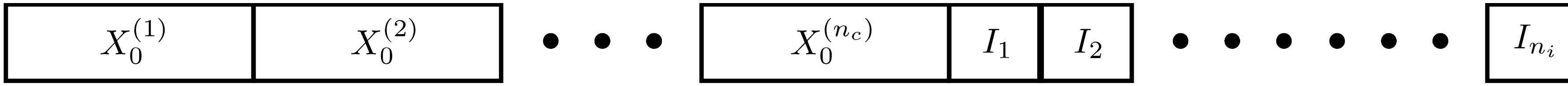


UCLA

LDGM Code



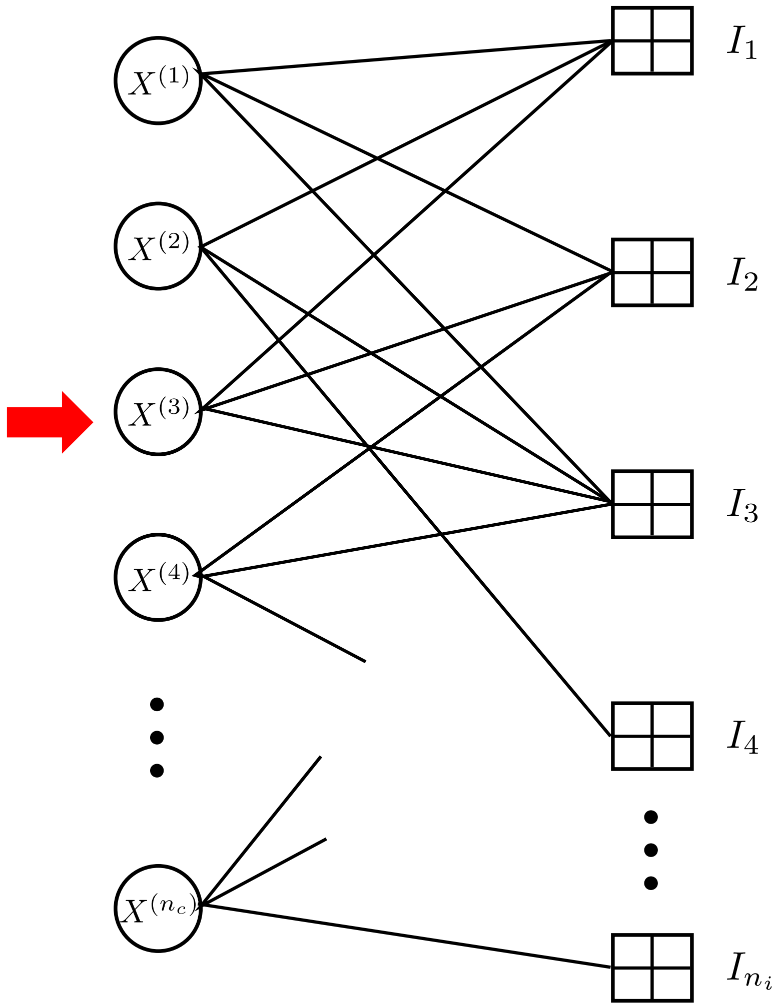
Inter-frame code at the Decoder



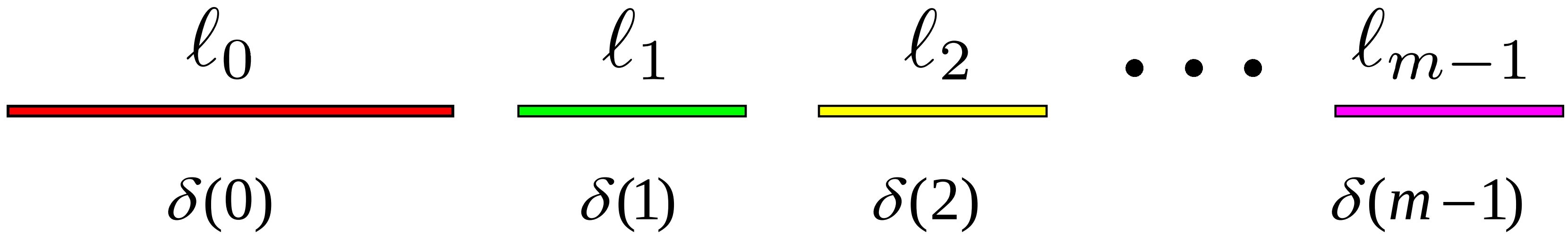
+ Noise

Decoder structure

VL decoders initialized with $X_0^{(i)}$.



Statistics of VL Code in Inter-frame Code Analysis



$$\delta = \{\delta(0), \delta(1), \ldots, \delta(m-1), \delta(m)\}$$

$\delta(n),\ n < m$ is the probability of decoding correctly for the first time after $n+1$ transmissions.

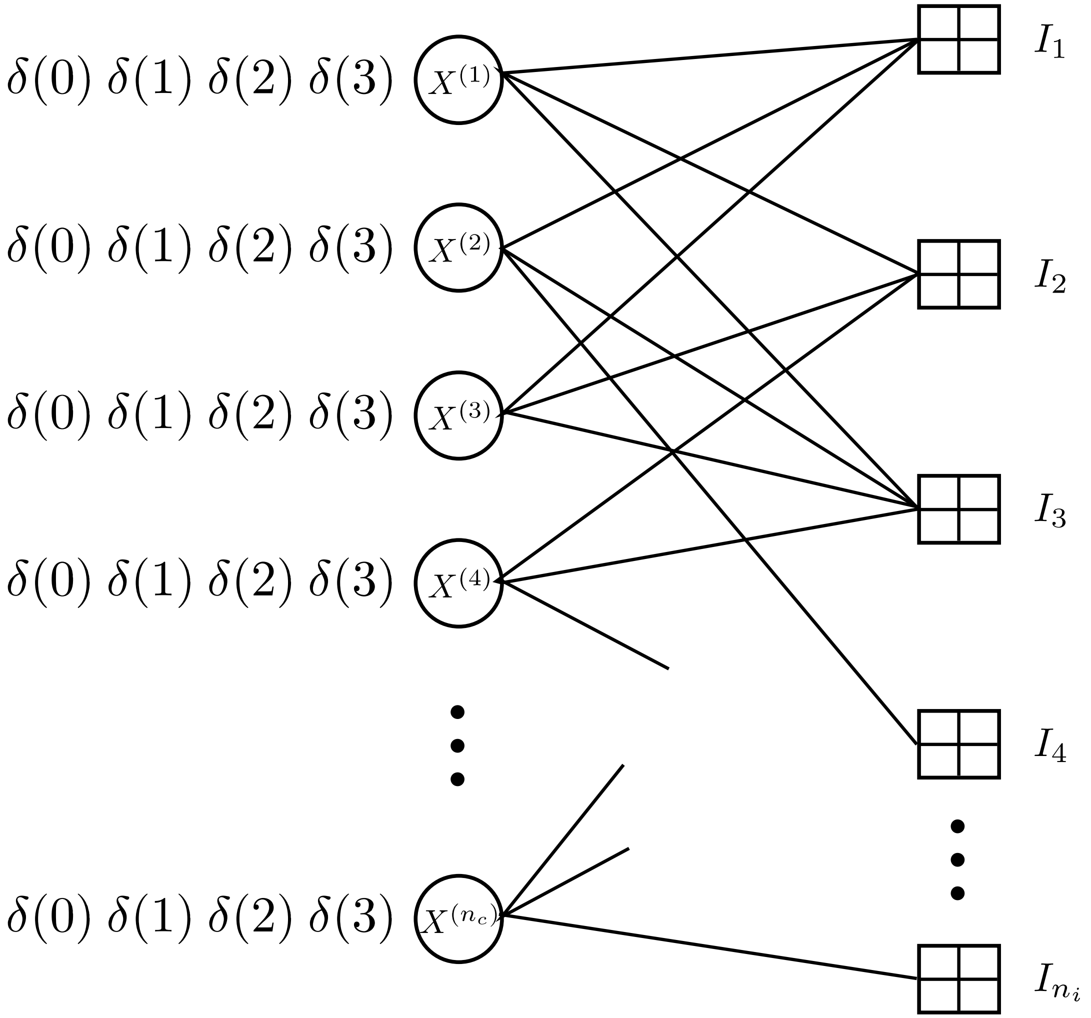
UCLA

Inter-frame Code – Peeling Decoder

- Every systematic node has a degree of 3 ($m = 3$).

Initialization:

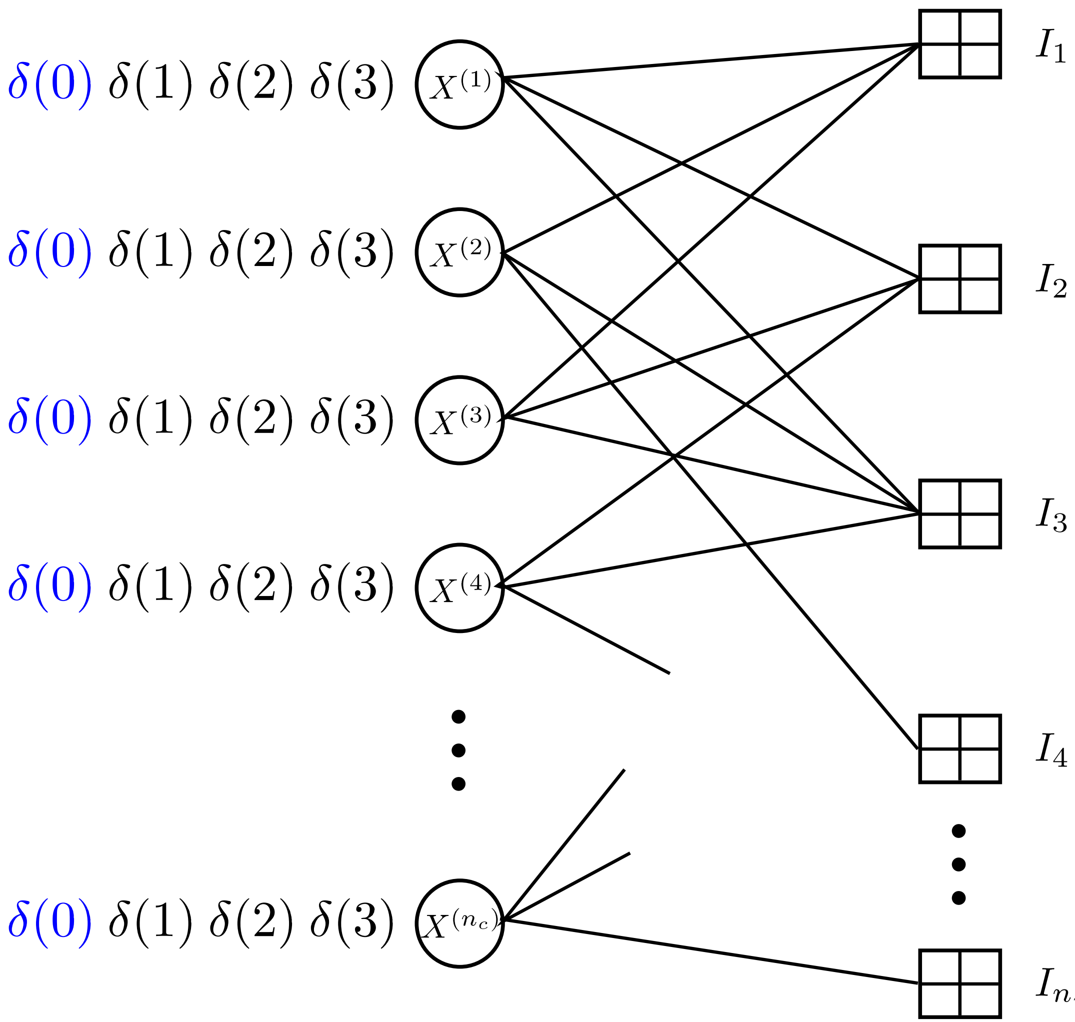
- Every VL decoder observes its noisy highest-rate codeword $X_0^{(i)}$.
- The parity nodes are, likewise, received from the channel.



Inter-frame Code – Peeling Decoder

Iteration 1 (left):

- The systematic nodes (VL decoder) attempt to decode with their highest-rate codewords.
- Each systematic node succeeds with probability $\delta(0)$.

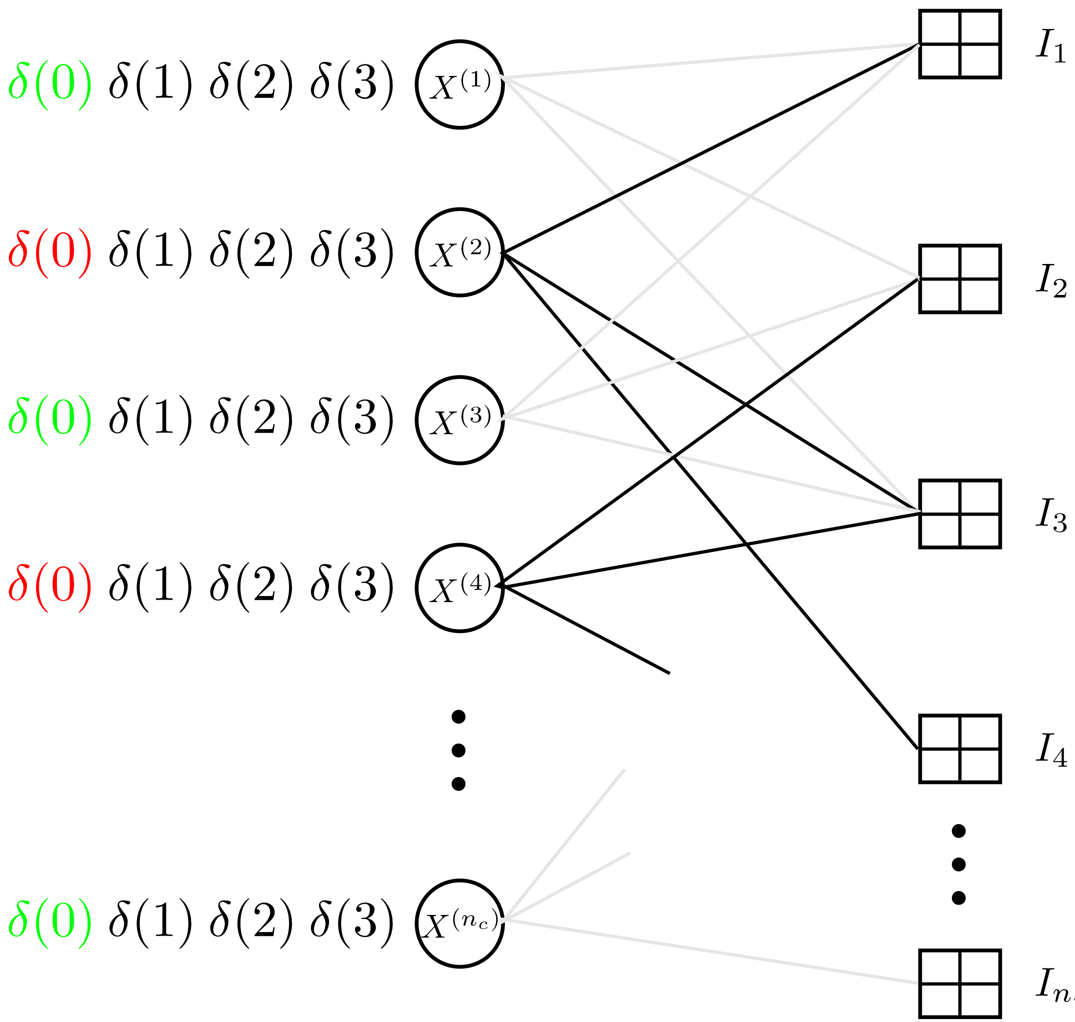


UCLA

Inter-frame Code – Peeling Decoder

Iteration 1 (left):

- The ones that succeed
  - can compute all their increments.
  - can remove effect of their increments from parities.

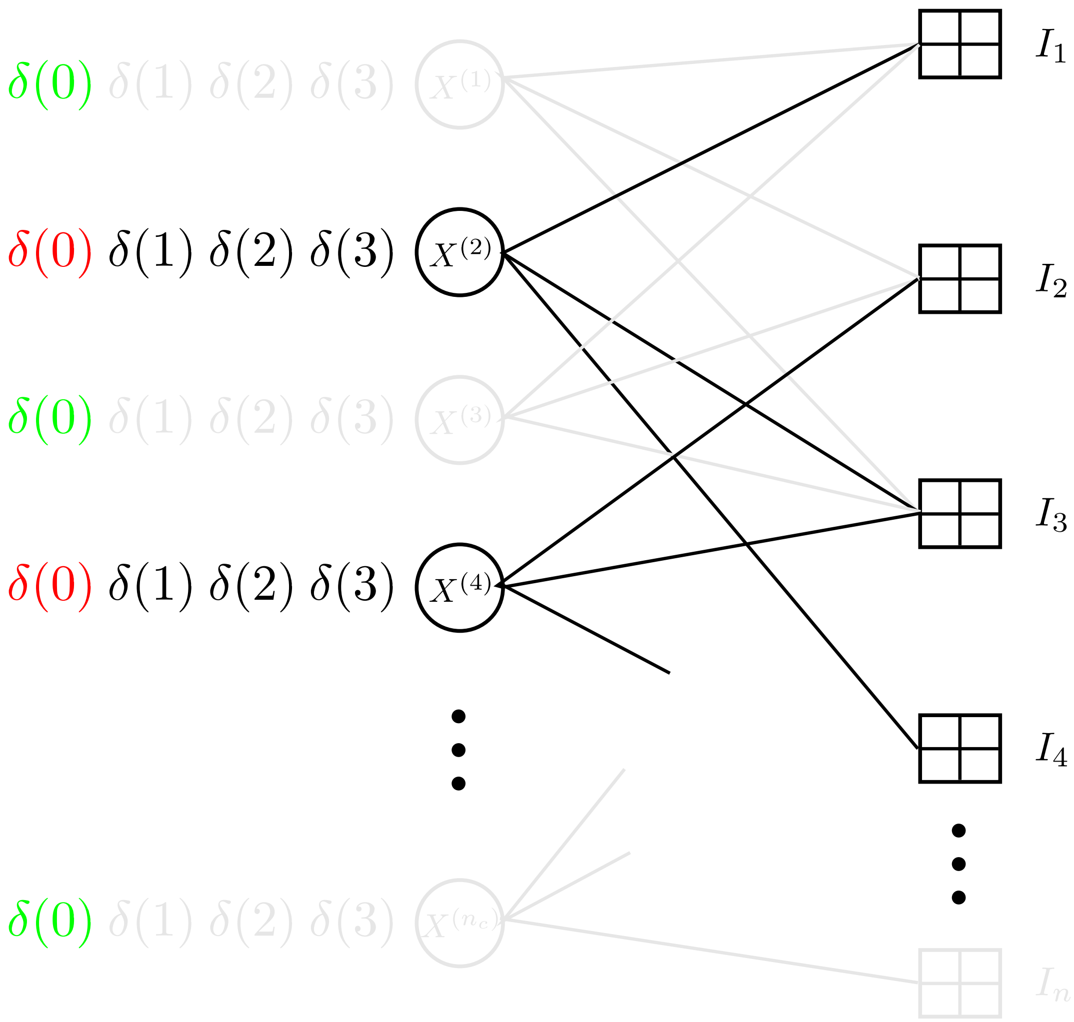


UCLA

Inter-frame Code – Peeling Decoder

Iteration 1 (left):

- The ones that succeed
  - can compute all their increments.
  - can remove effect of their increments from parities.

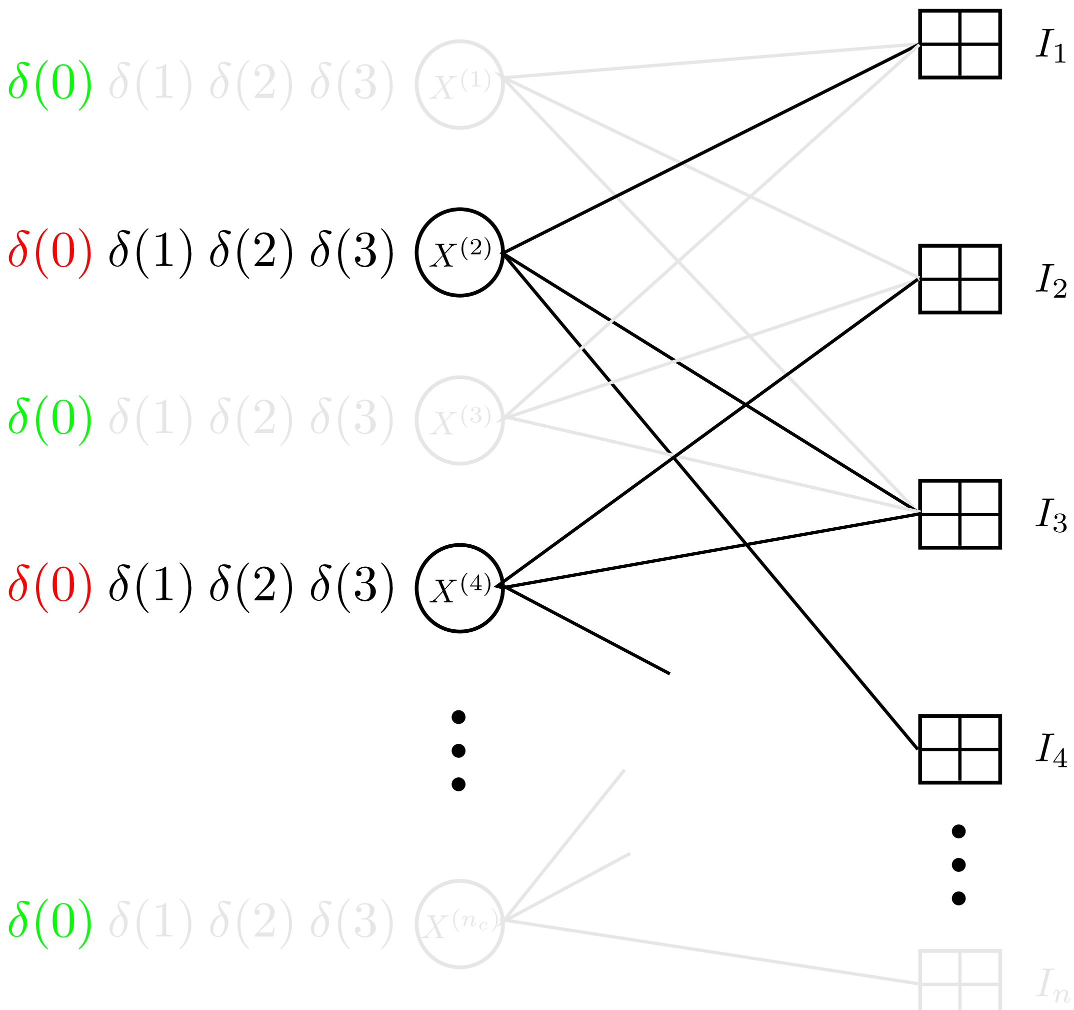


UCLA

Inter-frame Code – Peeling Decoder

Iteration 1 (left):

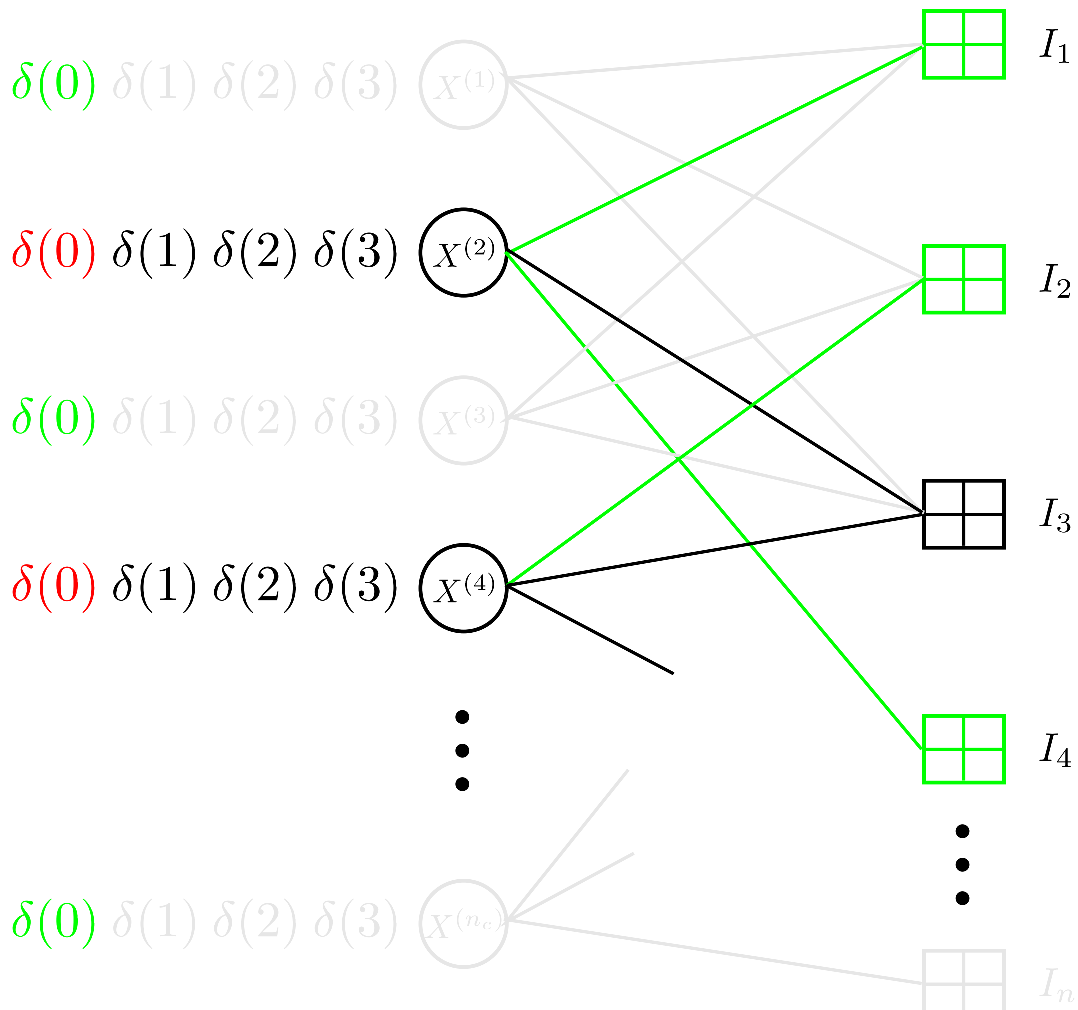
- The ones that fail
  - are retained.
  - when additional increments become available, they can attempt decoding again.



Inter-frame Code – Peeling Decoder

Iteration 1 (right):

- If all but one edge are deactivated, the parity node can become a known increment to a systematic node.

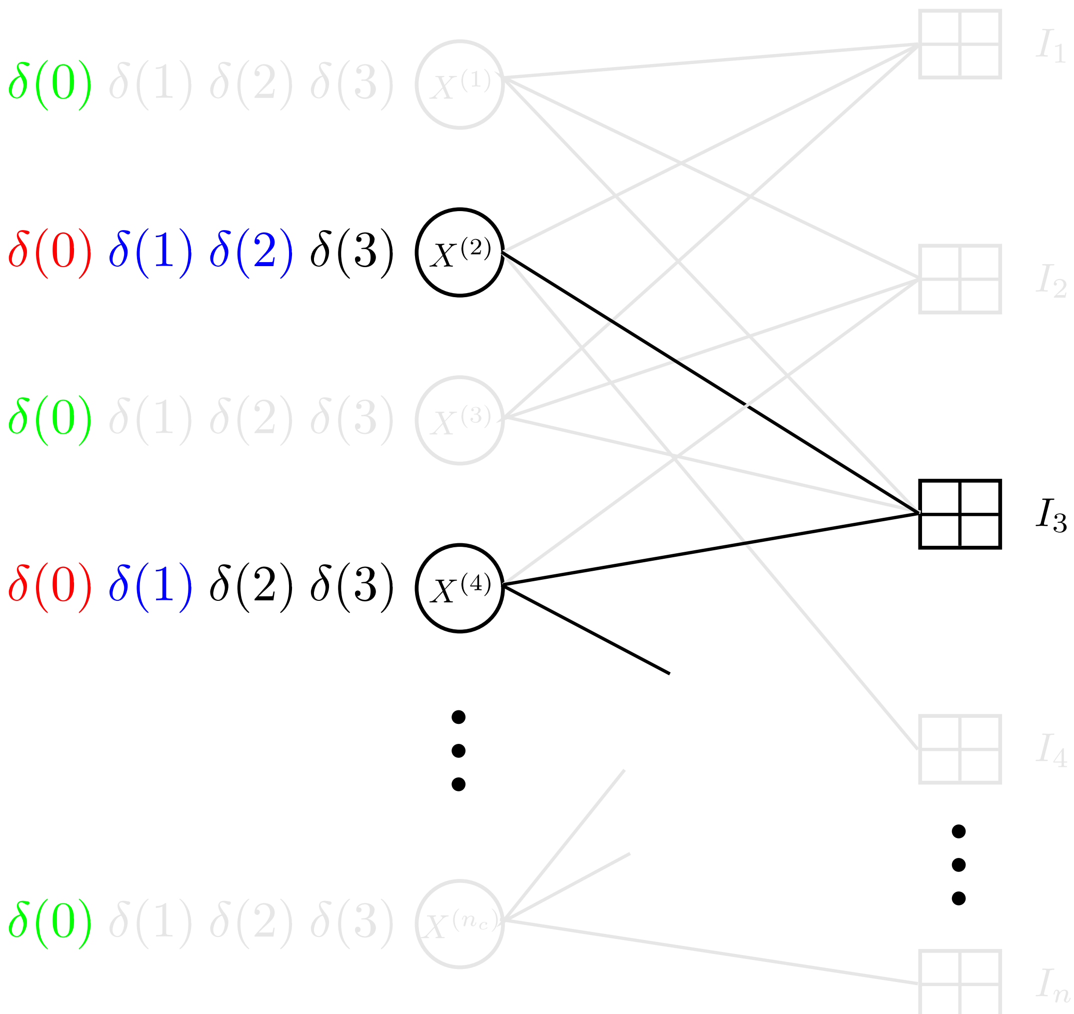


UCLA

Inter-frame Code – Peeling Decoder

Iteration 1 (right):

- Systematic nodes append available increments to lower their rate.

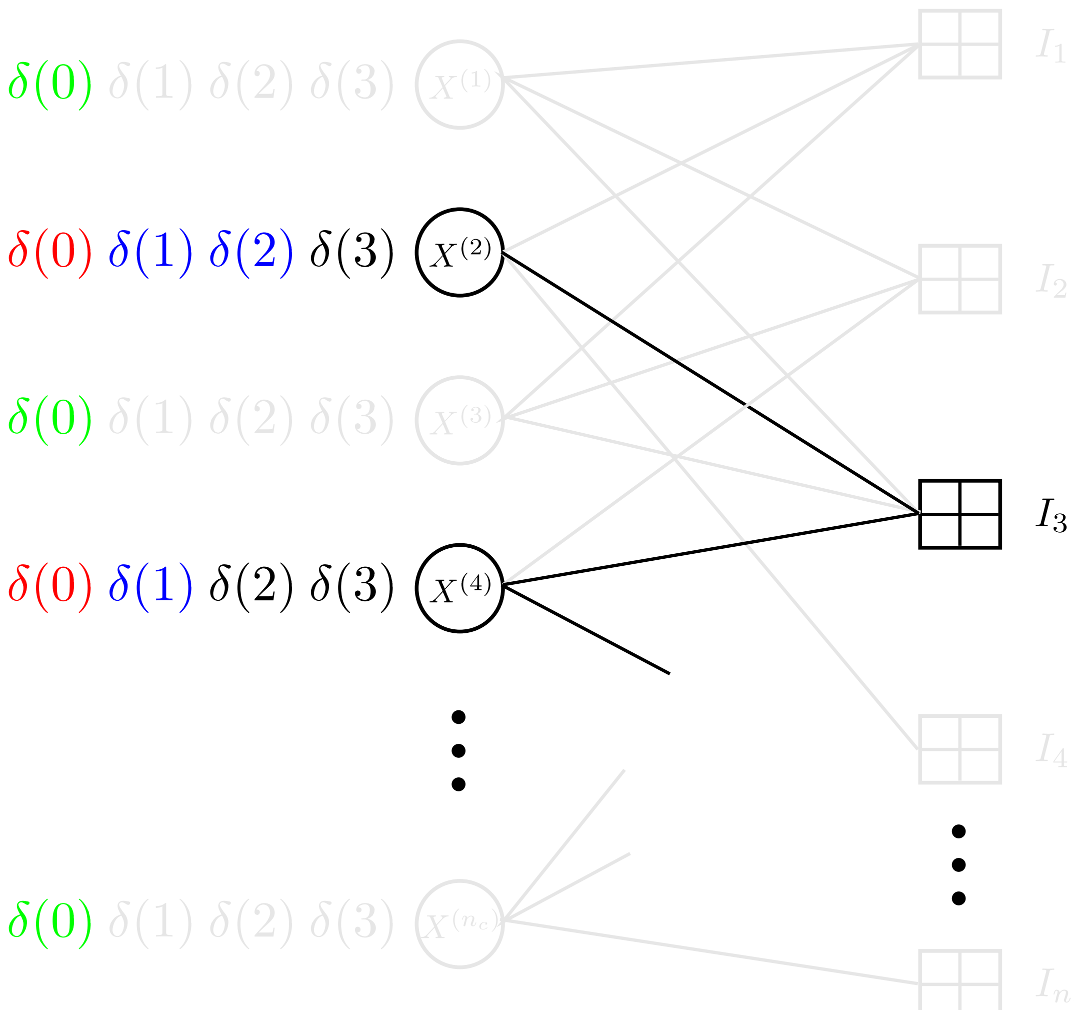


UCLA

Inter-frame Code – Peeling Decoder

Iteration 2:

- The systematic nodes (yet to decode) decode again if new increments are available to them.

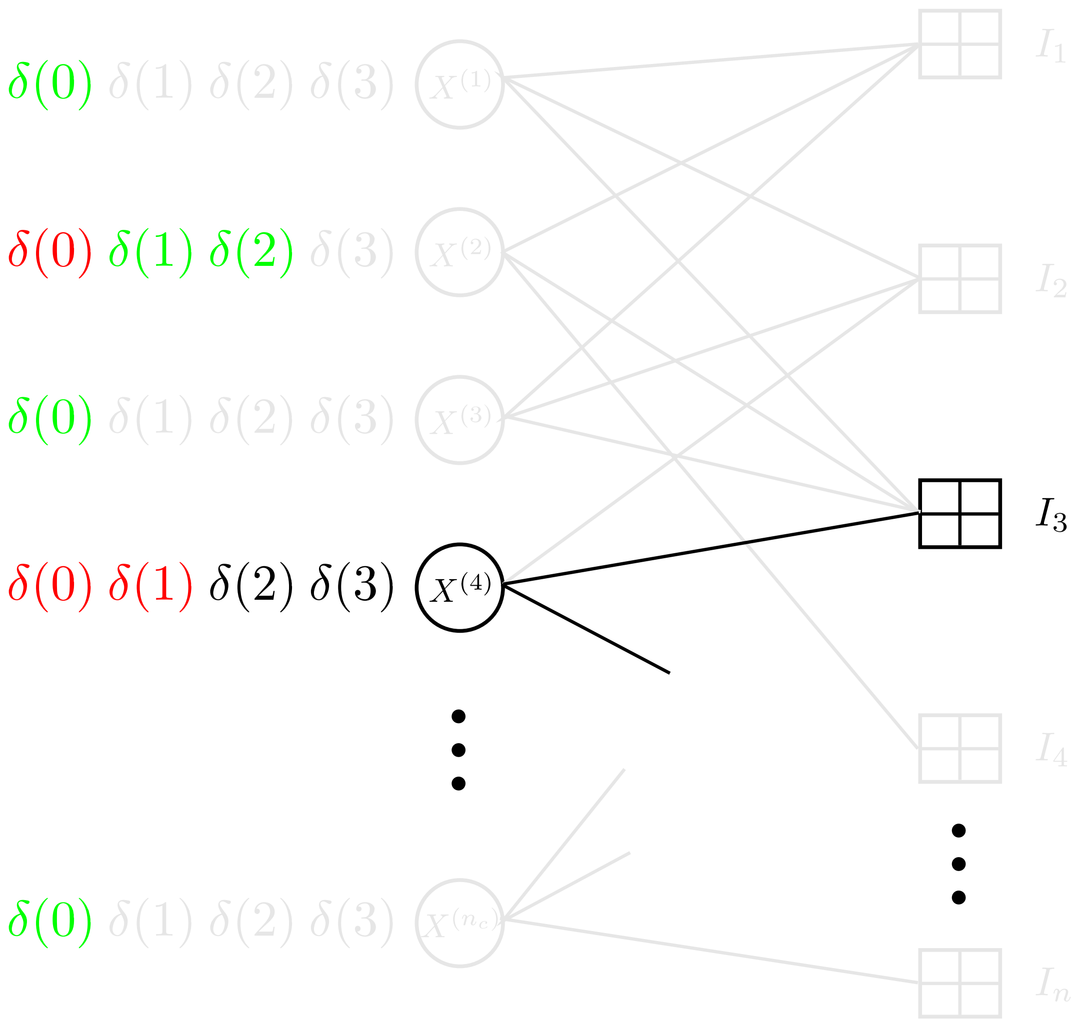


UCLA

Inter-frame Code – Peeling Decoder

Iteration 2:

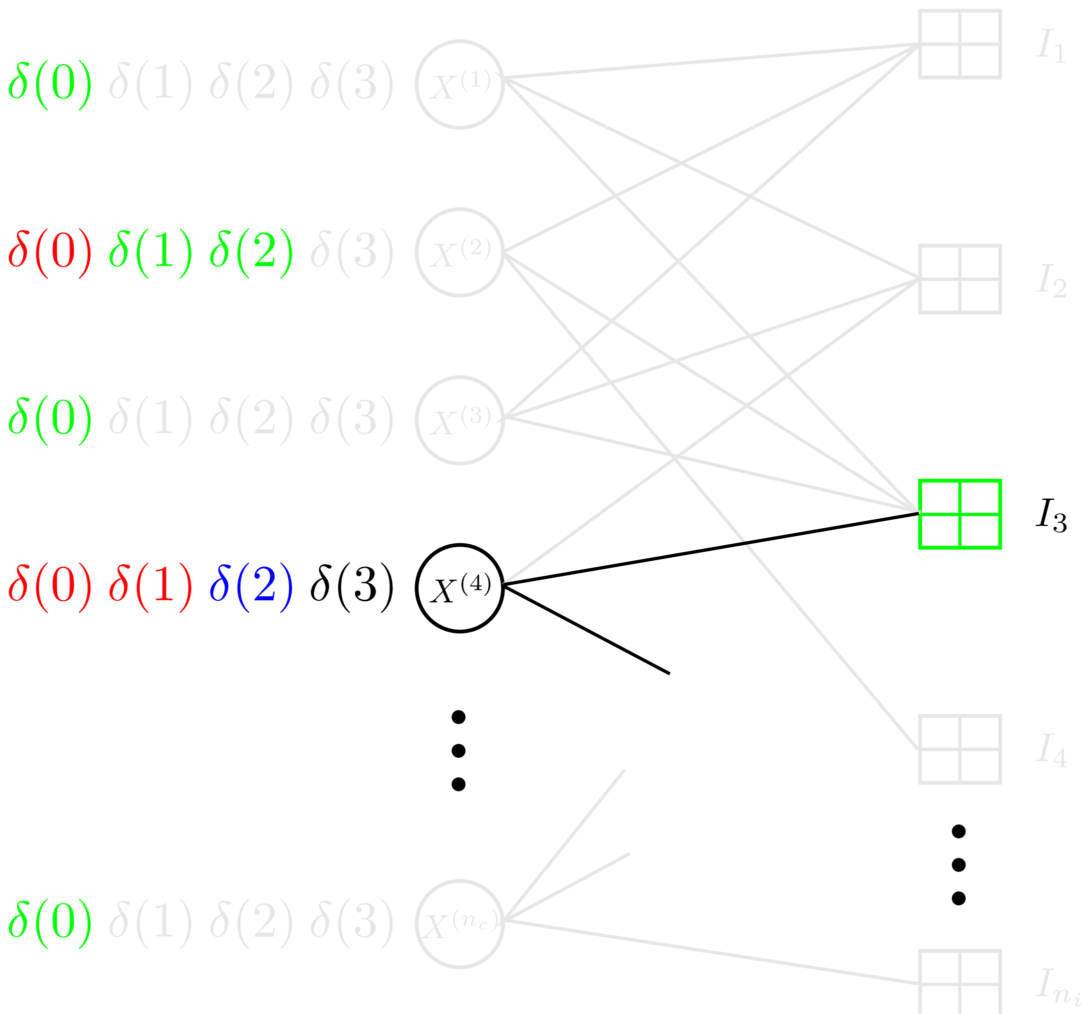
- The ones that successfully decode can be removed from the graph along with all their edges.



UCLA

Inter-frame Code – Peeling Decoder

The process continues until no more systematic nodes can be recovered.



UCLA

Outline

- **Previous work**: Approaching Capacity with Short Blocklengths using Incremental Redundancy and ACK/NACK Feedback

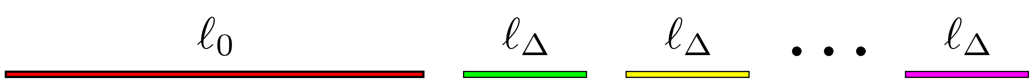
- **New Idea**: Approaching Capacity using Many Short-blocklength Codes with Incremental Redundancy in Parallel *Without Feedback*
  - Concept
  - Design methods and design examples
    - Differential evolution for degree distribution
    - Quasi-regular heuristic for degree distribution

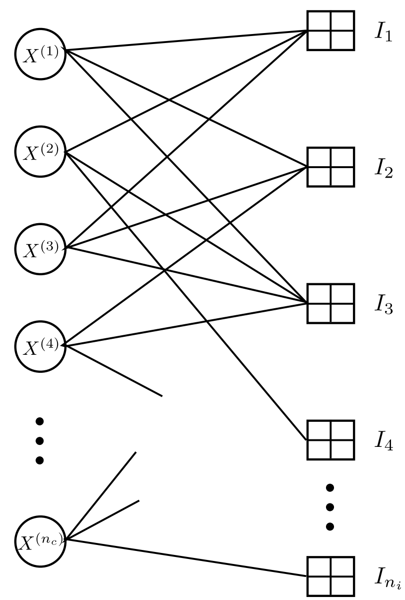
What need to be designed in an inter-frame system?

- Choose a VL code and a maximum number of transmissions ($m = 5$) allowed.

What need to be designed in an inter-frame system?

- Given a VL code with a fixed number of transmissions ($m = 5$) allowed, there are 3 parts to design.
  - The initial transmission length (ℓ_0) and increment length (ℓ_Δ)
  - The degree distributions of the inter-frame code
  - The bipartite graph (parity matrix) of the inter-frame code





What need to be designed in an inter-frame system?

- Given a VL code with a fixed number of transmissions ($m = 5$) allowed, there are 3 parts to design.
  - The initial transmission length (ℓ_0) and increment length (ℓ_Δ):
    through brute-force search or sequential differential optimization (SDO) [Vakilinia et al. TCOM 2016]
  - The degree distributions of the inter-frame code
  - The bipartite graph (parity matrix) of the inter-frame code

UCLA

What need to be designed in an inter-frame system?

- Given a VL code with a fixed number of transmissions ($m = 5$) allowed, there are 3 parts to design.
  - The initial transmission length (ℓ_0) and increment length (ℓ_Δ):
    through brute-force search or sequential differential optimization (SDO) [Vakilinia et al. TCOM 2016]
  - The degree distributions of the inter-frame code
  - The bipartite graph (parity matrix) of the inter-frame code:
    through progressive edge growth (PEG) [Hu et al. IT 2005]

What need to be designed in an inter-frame system?

- Given a VL code with a fixed number of transmissions ($m = 5$) allowed, there are 3 parts to design.
  - The initial transmission length (ℓ_0) and increment length (ℓ_Δ): through brute-force search or sequential differential optimization (SDO) [Vakilinia et al. TCOM 2016]
  - The degree distributions of the inter-frame code
    - Differential evolution
    - Quasi-regular heuristics
  - The bipartite graph (parity matrix) of the inter-frame code: through progressive edge growth (PEG) [Hu et al. IT 2005]

Outline

- **Previous work**: Approaching Capacity with Short Blocklengths using Incremental Redundancy and ACK/NACK Feedback

- **New Idea**: Approaching Capacity using many Short-blocklength Codes with Incremental Redundancy in Parallel *Without Feedback*
  - Concept
  - Design methods and design examples
    - Differential evolution for degree distribution
    - Quasi-regular heuristic for degree distribution

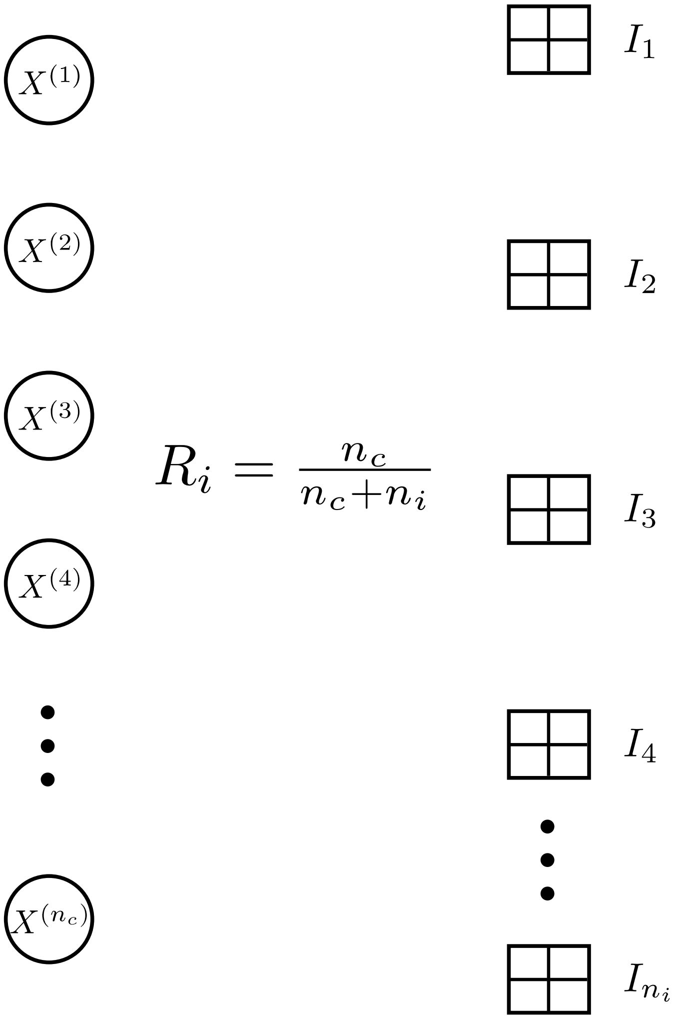
Design Degree Distributions using Differential Evolution

- Given a $\boldsymbol{\delta} = \{\delta(0), \delta(1), \dots, \delta(m)\}$ and m, the objective is to find $\lambda(x), \rho(x)$.
- The peeling decoder can be analyzed using density evolution.
- For a given $\lambda(x), \rho(x)$ pair, density evolution equations can predict the residual systematic-node error rate after a large number of iterations.
- Differential evolution method can then be used to find a $\lambda(x), \rho(x)$ pair with a low-enough failure rate.

Differential Evolution

- Differential evolution is a type of genetic algorithm that optimizes a problem by iteratively improving randomly generated candidates regarding a metric.
- The candidates in our problem are randomly generated $\lambda(x)$,$\rho(x)$ pairs that have a certain LDGM rate R_i.
- The metric is for the codeword (systematic nodes) failure probability ϵ_{FF} to be as small as possible, and below 10^{-3}.
- The LDGM rate R_i is a meta-parameter that are chosen to be as high as the optimization produces valid results.

Inter-frame Code – Rate of LDGM Code



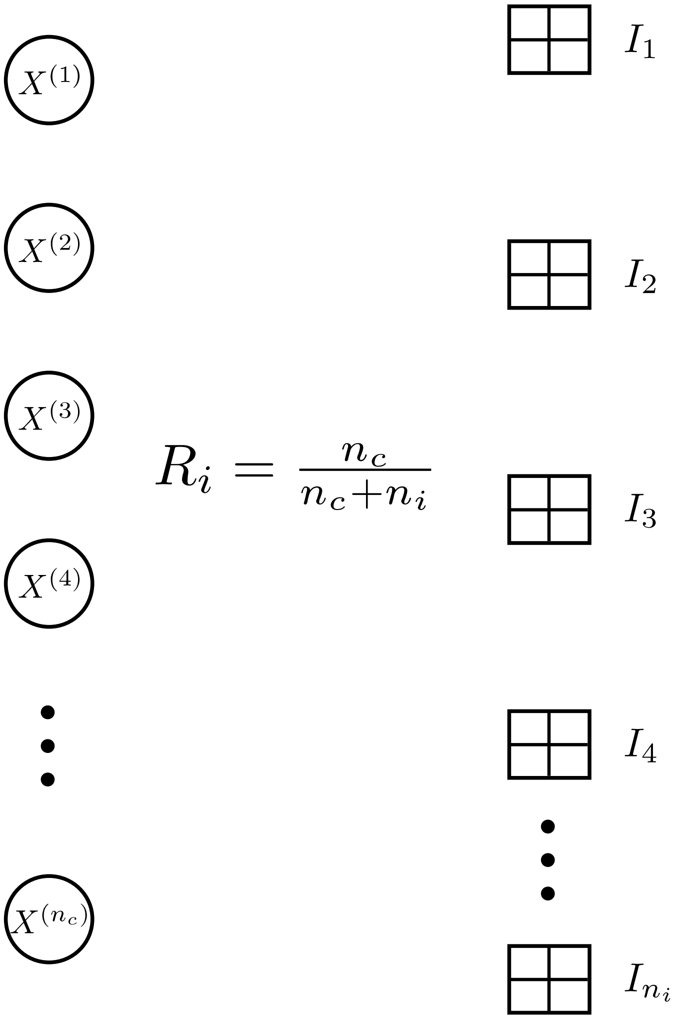
$$R_i = \frac{n_c}{n_c + n_i}$$

UCLA

Inter-frame Code – Throughput Rate

$$R_t^{(FF)} = \frac{n_c k(1-\epsilon_{FF})}{n_c \ell_0 + n_i \ell_\Delta}$$

$$R_t^{(FF)} = \frac{k(1-\epsilon_{FF})}{\ell_0 + (R_i^{-1}-1)\ell_\Delta}$$



$$R_i = \frac{n_c}{n_c + n_i}$$

Predict the Failure Probability of the Peeling Decoder

- For a pair of $\lambda(x), \rho(x)$, the codeword failure rate ϵ_{FF} can be calculated using density evolution directly.
- An analytical characterization of ϵ_{FF} can also be used in differential evolution.
- Luby et al. proposed using differential equations or the and-or tree approach to analyze the decoding process of the peeling decoder.
- We extend Luby et al.'s analysis to the inter-frame code by using direct probabilistic arguments.

Peeling Algorithm for Decoder Analysis

- Initially, each VL decoder is assigned a generalized erasure state drawn according to PMF $\boldsymbol{\delta}$.
- Remove all the left nodes that decode, and their incident edges.
- **WHILE** right-degree-one edges (i.e. available increments) remain in the graph
  - Randomly select **one** right-degree-one edge Q_t.
  - Remove Q_t (and its incident right node).
  - Reduce the generalized erasure state of its incident left node by 1.
  - **IF** the left node can decode (the generalized erasure state is 0)
    - Remove the left node and its remaining incident edges.
  - **ENDIF**
- **ENDWHILE**

Characterize the Peeling Process

- As long as there is an edge connects to a degree-1 right node, the peeling process continues.
- Peeling process metric $r_1(x)$: the probability that a randomly picked edge in the initial bipartite graph has not been removed after t iterations, and connects to a degree-1 right node.
- We use $r_1(x)$ to predict ϵ_{FF}.

Definition of x

- Define $x(t)$ or simply x as the probability that a randomly selected edge in the initial graph that is not in the set $\{Q_1, \ldots, Q_t\}$.
- In [Luby et al. IT 2001], x is defined with a differential equation to solve differential equations to find $r_1(x)$.

Characterize the Peeling Process

- $p_l(x)$: the probability that a randomly selected edge in the initial graph has as its incident left node a VL decoder that cannot decode after $\{Q_1, \ldots, Q_t\}$ have been provided as potential increments by all the other edges connecting to that VL decoder.
- $p_r(x)$: the probability that a randomly selected edge in the initial graph has as its incident right node a node with exactly one edge remaining after $\{Q_1, \ldots, Q_t\}$ have been provided as increments to the VL decoders.

Characterize the Peeling Process

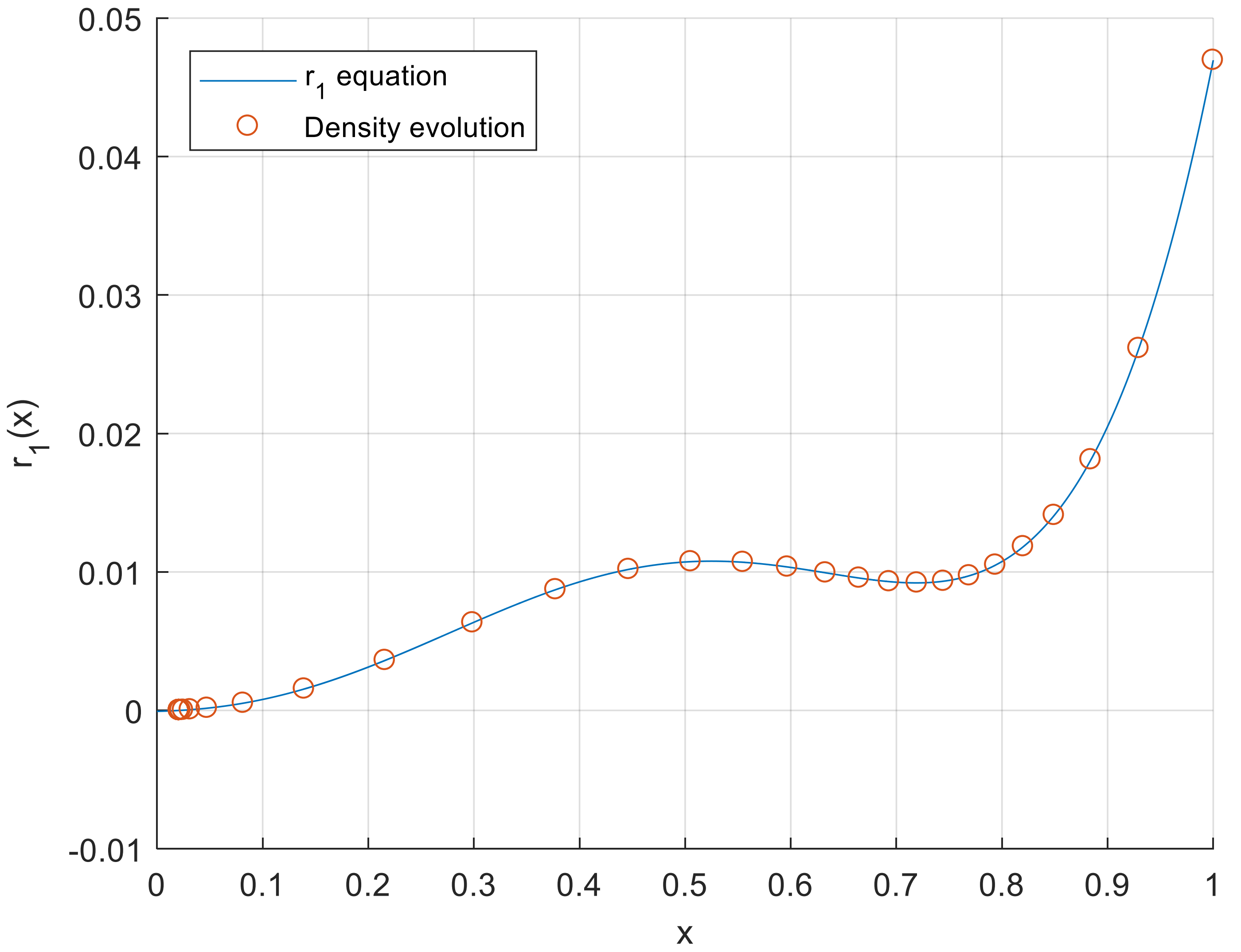
$$r_1(x) = p_l(x)p_r(x) - p_l(x)(1-x)$$

$$p_l(x) = \sum_{\omega=1}^{m} \delta(\omega) \sum_{i=1}^{d_L} \lambda_i \sum_{j=0}^{\min(\omega,i)-1} \binom{i-1}{j} (1-x)^j x^{i-1-j}$$

$$p_r(x) = \rho(1 - p_l(x))$$

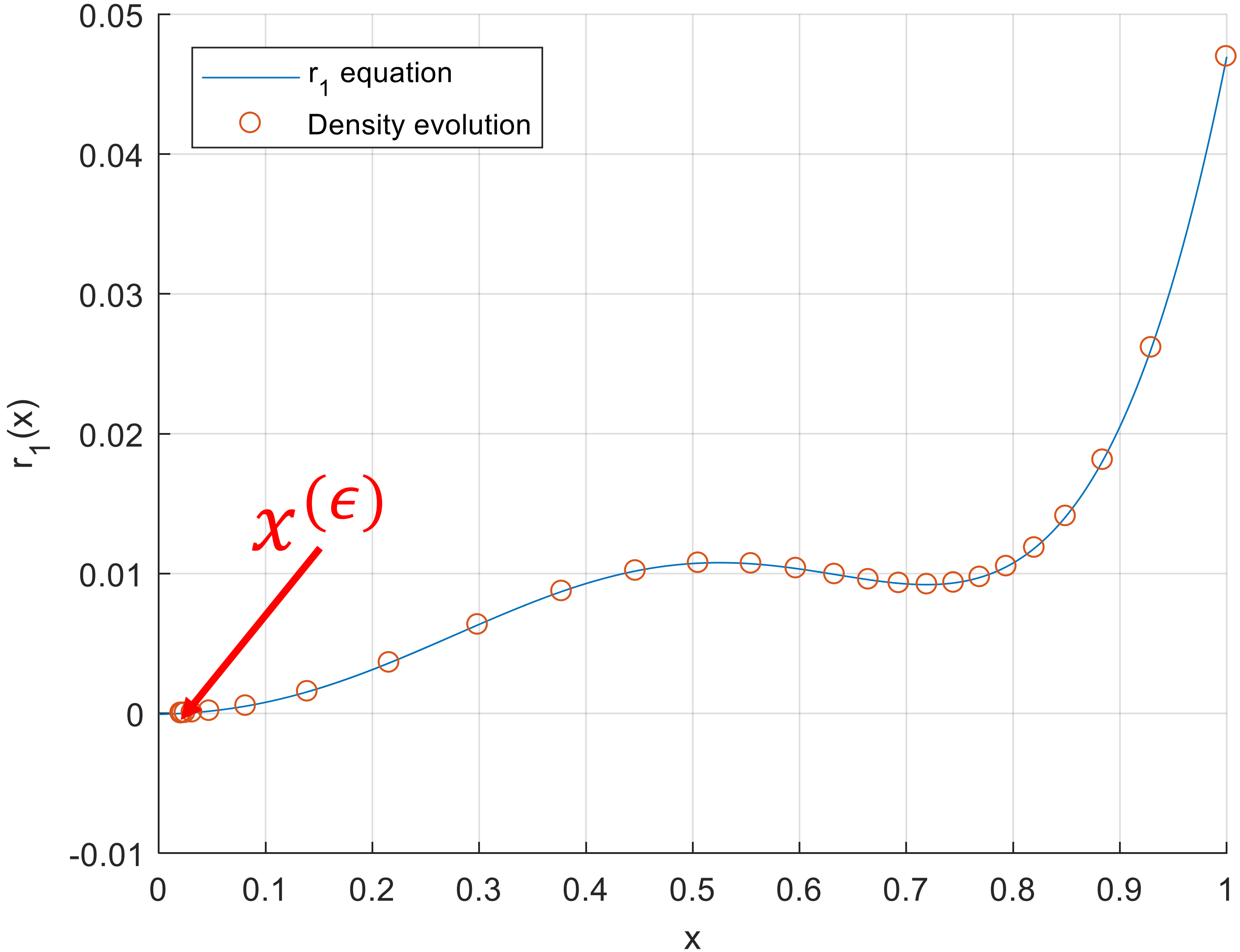
UCLA

Characterize the Peeling Process



UCLA

Characterize the Peeling Process



Probability of Failure ϵ_{FF}

$$\epsilon_{FF} = \sum_{\omega=1}^{m} \delta(\omega) \sum_{i=1}^{d_L} \Lambda_i \sum_{j=0}^{\min(\omega-1,i)} \binom{i}{j} \left(1 - x^{(\epsilon)}\right)^j \left(x^{(\epsilon)}\right)^{i-j}$$

$$\Lambda_i = \frac{\lambda_i / i}{\sum_{j=1}^{d_L} \lambda_i / i}$$

UCLA

Convolutional Code as VL Code [Williamson et al. TCOM 2014]

VL encoder

k=64 bits, Rate-1/3, 1024-state, tail-biting convolutional code (TBCC) with pseudorandom puncturing [Ma et al. TCOM 1986]

Channel

Binary-input AWGN Channel with SNR=2 dB

VL decoder

Tail-biting reliability-output Viterbi algorithm (ROVA) [Raghavan et al. TIT 1998]

Convolutional VL Code parameters with Constant-size Increments

$\boldsymbol{\ell_0}$	**$\boldsymbol{\ell_\Delta}$ for $\boldsymbol{m = 5}$ (four increments)**	**Throughput Rate $R_t^{(FB)}$ with ACK/NACK Feedback**	**Percentage of Capacity of 2dB BI-AWGN**
108 bits	16 bits	0.5208	81.10%

- $\boldsymbol{\delta} = \{0.333, 0.449, 0.182, 0.0316, 0.00402, 0.000505\}$

UCLA

Design Example – Regular LDGM Code

- Systematic node degree: 4
- Parity node degree: 3
- LDGM code design rate $R_i = 0.4286$
- Number of systematic nodes = 100,000

Throughput Rate $R_t^{(FB)}$ with ACK/NACK Feedback	**Throughput Rate $R_t^{(FF)}$ - inter-frame code (regular LDGM)**
0.5208	0.4945

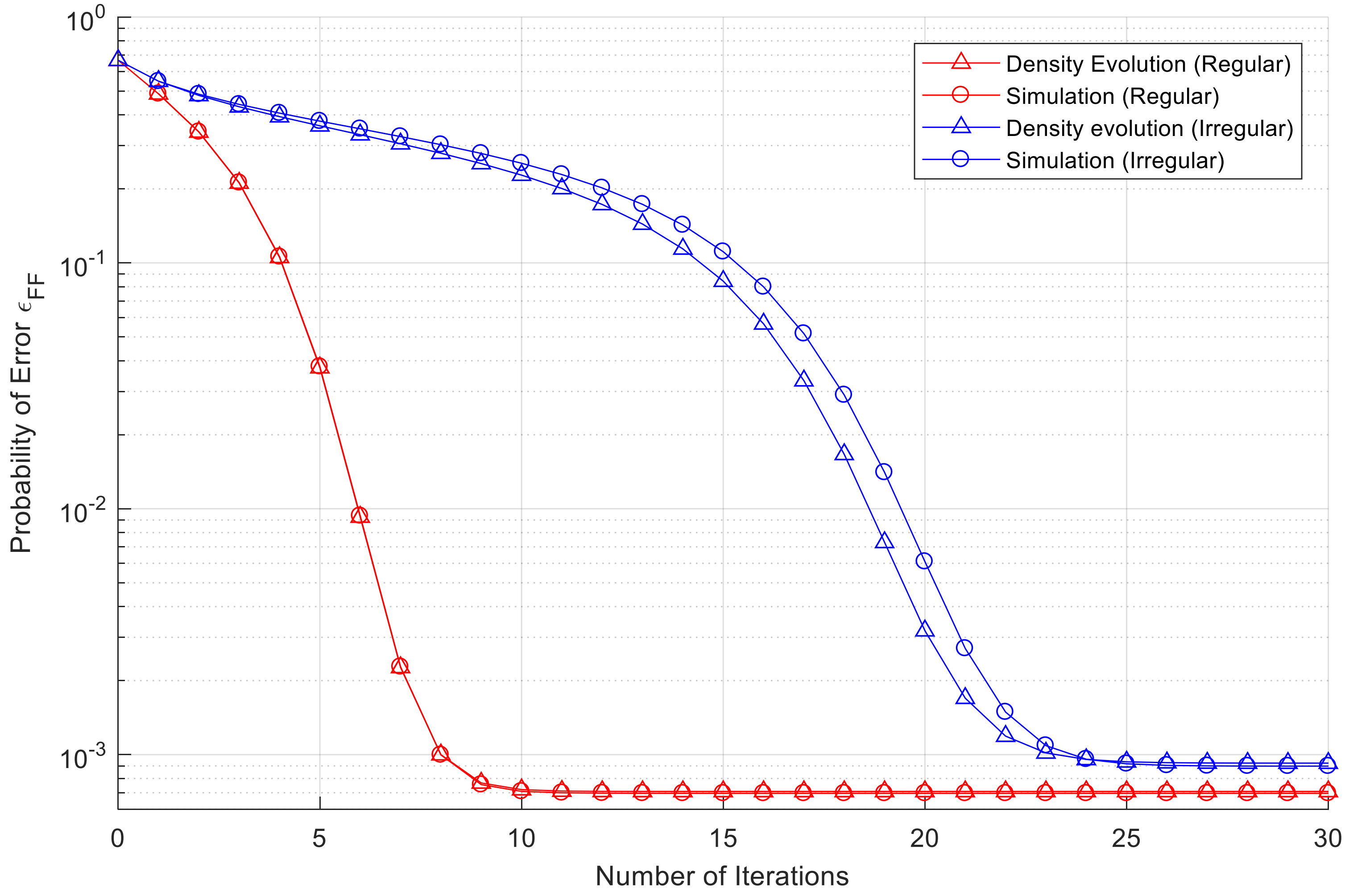
Design Example – Irregular LDGM Code

- Maximum systematic node degree: 4
- Maximum parity node degree: 10
- LDGM code design rate $R_i = 0.48$
- Number of systematic nodes = 100,000

Throughput Rate $R_t^{(FB)}$ with ACK/NACK Feedback	**Throughput Rate $R_t^{(FF)}$ - inter-frame code (regular LDGM)**	**Throughput Rate $R_t^{(FF)}$ - inter-frame code (irregular LDGM)**
0.5208	0.4945	0.5102

UCLA

Probability of Error Characterization of the 100,000 Systematic Node Codes



Design Example – Shorter LDGM Code

- Maximum systematic node degree: 4
- Maximum parity node degree: 10
- LDGM code design rate $R_i = 0.46$
- Number of systematic nodes: 1000

Throughput Rate $R_t^{(FB)}$ with ACK/NACK Feedback	Throughput Rate $R_t^{(FF)}$ - inter-frame code (irregular LDGM)
0.5208	0.5044

UCLA

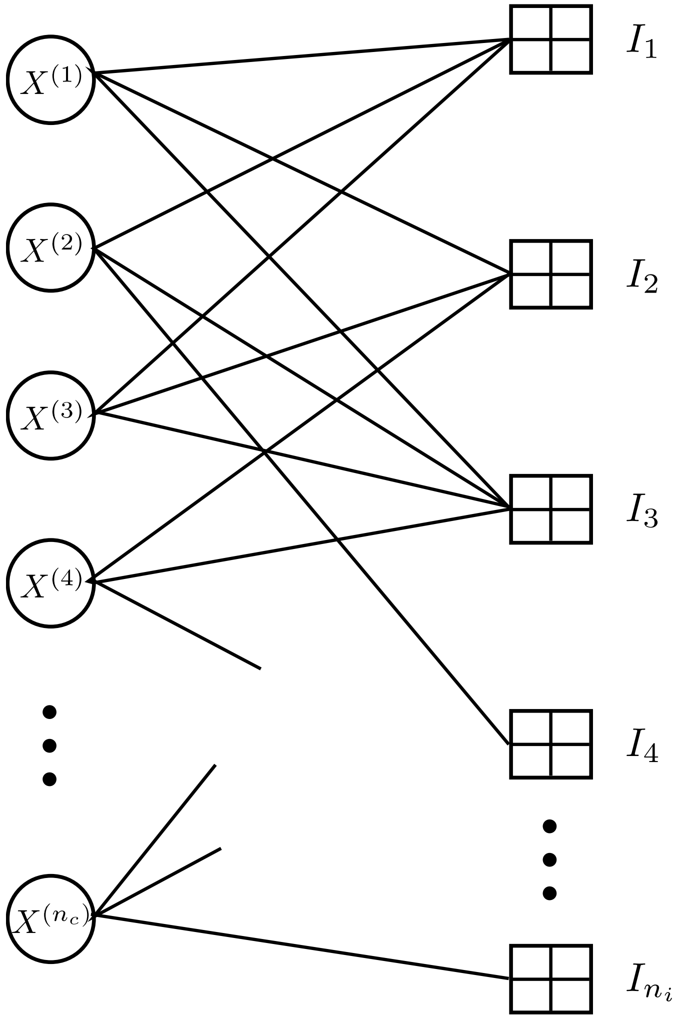
Design Example – Comparison vs Capacity

	VL Code with ACK/NACK Feedback	100,000 systematic nodes		1000 systematic nodes
		Regular LDGM	Irregular LDGM	Irregular LDGM
Throughput rate	0.5208	0.4945	0.5102	0.5044
Percentage of Capacity of 2dB BI-AWGN	81.10%	77.01%	79.45%	78.55%
Percentage of ACK/NACK Feedback Throughput	---	95.0%	98.0%	96.9%

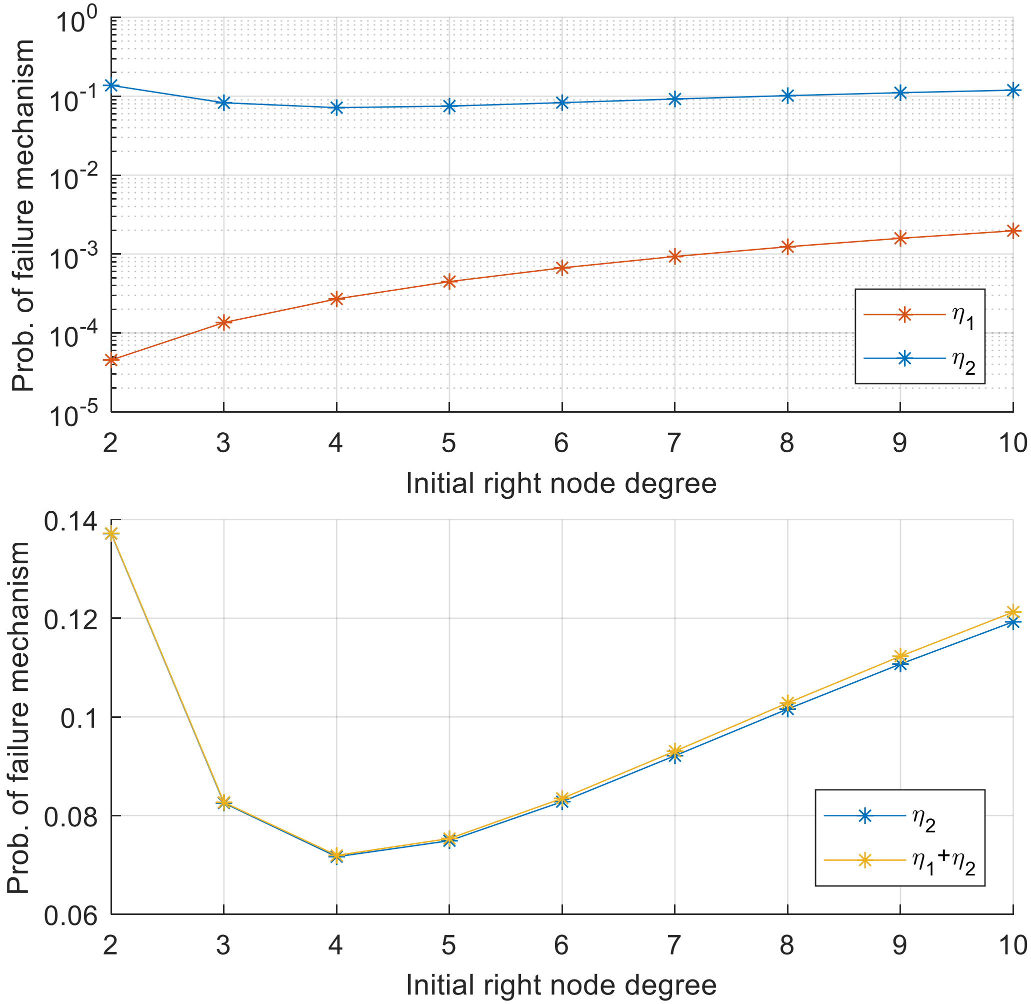
The throughput loss is the result of using more linear combinations of increments (right nodes) than the feedback system.

Three Mechanisms of Throughput Loss

- The degree of the right node of interest (RNOI) never decreases below two. (η_1)
- The degree of the RNOI decreases from two or more to zero in a single iteration of the peeling decoder so that it never provides an increment. (η_2)
- The degree of the RNOI achieves the value of one during an iteration so that it provides an increment to a left node, but other right nodes simultaneously provide the remaining required increments to that left node making the RNOI's increment superfluous. (η_3)



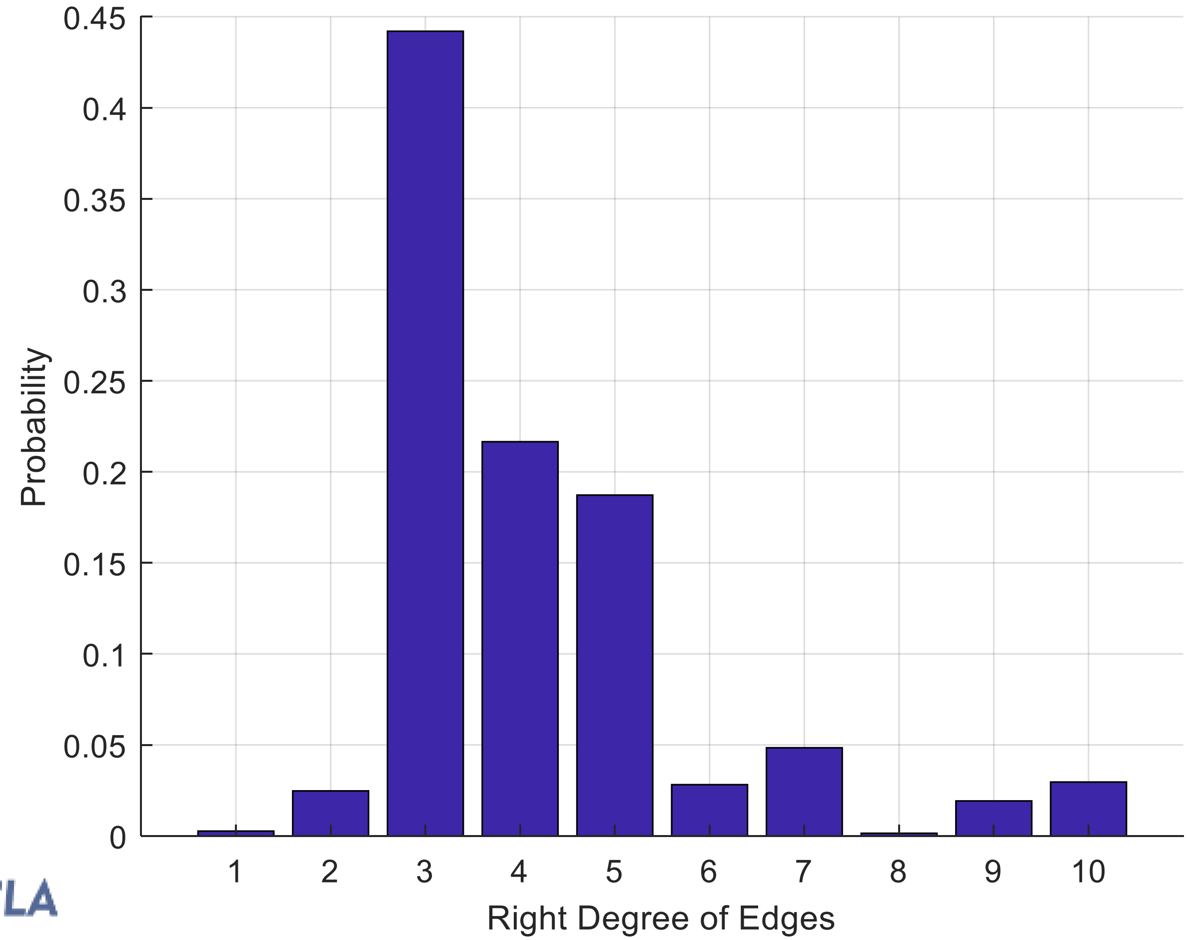
Probability of Failure Mechanisms



Quasi-regular heuristic for the right degree distribution

Right Degree Distribution Example from Differential Evolution

- The right degree distribution of the 100,000-systematic-node irregular LDGM code is:



Quasi-regular!

This is different from the Poisson right degree distribution proposed in [Luby et al. IT 2001] and Zeineddine et al. JSAC 2016] .

UCLA

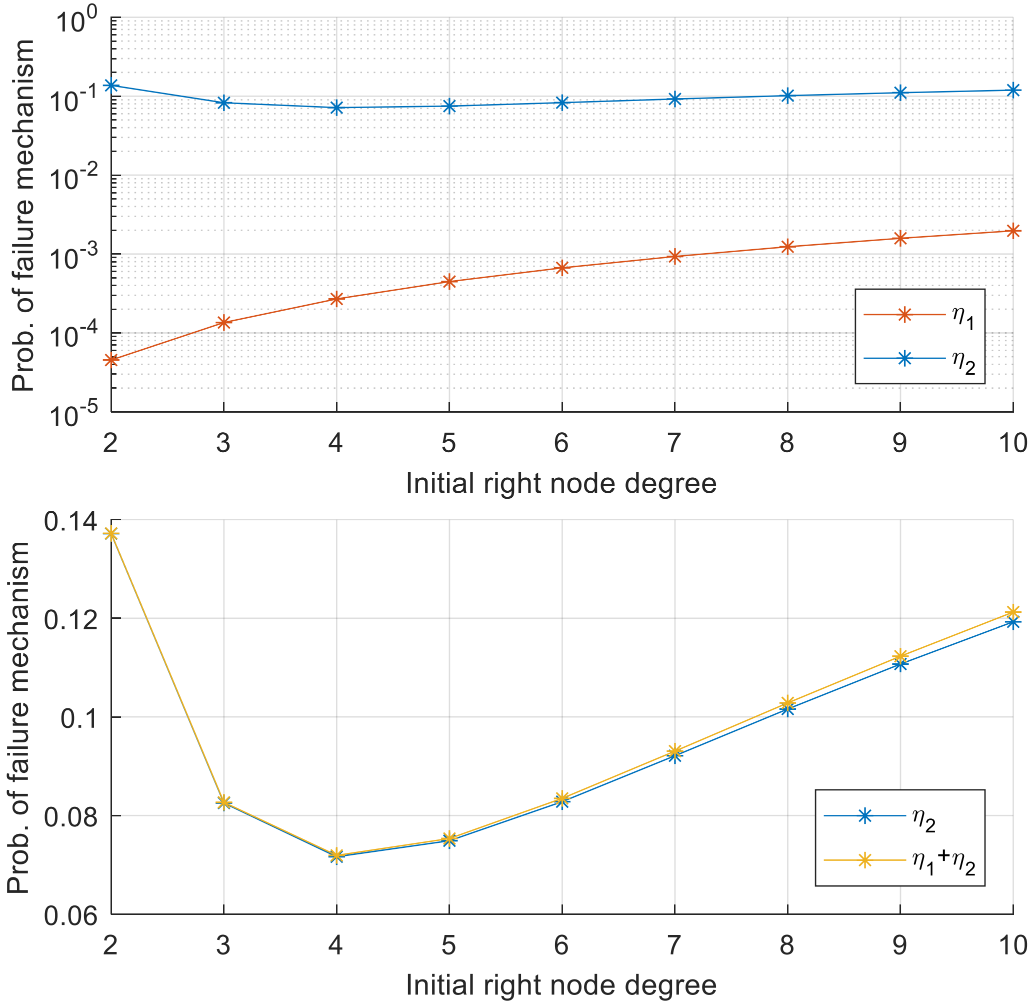
Outline

- **Previous work**: Approaching Capacity with Short Blocklengths using Incremental Redundancy and ACK/NACK Feedback
- **New Idea**: Approaching Capacity using many Short-blocklength Codes with Incremental Redundancy in Parallel *Without Feedback*
  - Concept
  - Design methods and design examples
    - Differential evolution for degree distribution
    - Quasi-regular heuristic for degree distribution

Quasi-regular heuristic for degree distribution

- Given a $\boldsymbol{\delta} = \{\delta(0), \delta(1), \ldots, \delta(m)\}$ and m, the objective is to find $\lambda(x), \rho(x)$.
- Set $\lambda(x) = x^3$ for $m = 5$ so that each left node has the maximum capacity of receiving required increments.
- Select $\rho(x) = \alpha x^2 + (1 - \alpha)x^3$ for example where α is the design parameter.
- Find α that maximizes the throughput and guarantees the target failure probability.

Which adjacent degrees to choose?



From a $\lambda(x), \rho(x)$ pair generated by differential evolution.

Which adjacent degrees to choose?

- For the VL code with feedback, define β_{FB} as the expected number of increments required by a VL decoder.

$$\beta_{FB} = \sum_{i=1}^{m-1} i\delta(i) + (m-1)\delta(m) = \mathbb{E}(\boldsymbol{\delta}) - \delta(m)$$

- For an inter-frame code, define β_{FF} as the average number of combined increments per left node. $\beta_{FF} = R_i^{-1} - 1.$

- Lower bound on β_{FF}:

$$\beta_{FF} \geq \beta_{FB}$$

Which adjacent degrees to choose?

- When the left degree distribution is regular, $\lambda(x) = x^{m-1}$, define a_R as the average right node degree.

$$\beta_{FF} = \frac{\int_0^1 \rho(x)\, dx}{\int_0^1 \lambda(x)\, dx} = \frac{m-1}{a_R}$$

$$\beta_{FF} \geq \beta_{FB} \Rightarrow a_R \leq \frac{m-1}{\beta_{FB}}$$

UCLA

Which adjacent degrees to choose?

- For the convolutional code example, $\beta_{FB} = 0.9260$, $m = 5$.

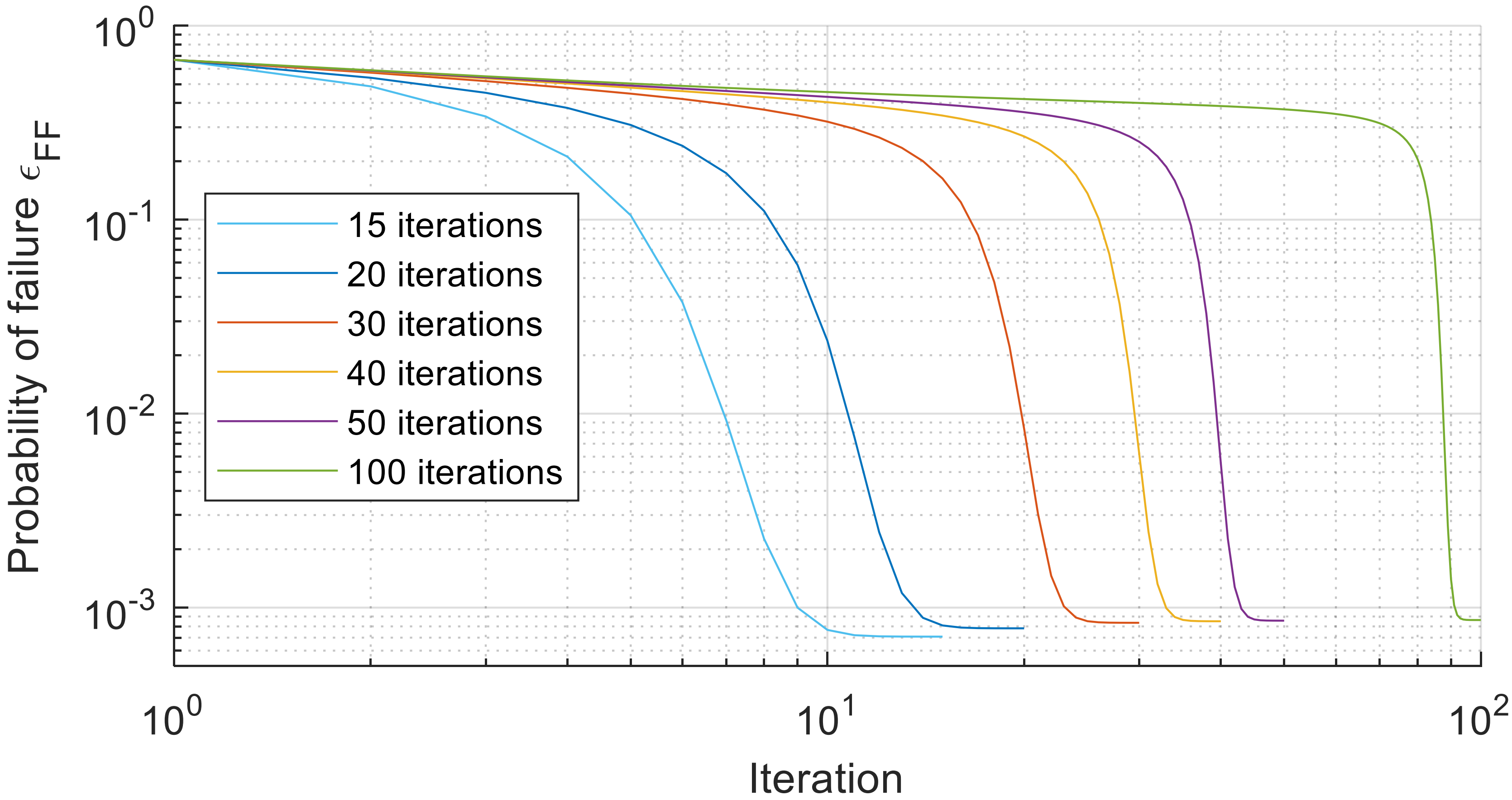
$$a_R \leq \frac{m-1}{\beta_{FB}} = 4.32$$

Asymptotic Performance of 2-degree Quasi-regular Right Degree Distribution from Density Evolution

- $\lambda(x) = x^3, \rho(x) = \alpha x^2 + (1-\alpha)x^3$, assume infinite large bipartite graphs

	α	a_R	β_{FF}	No. Iterations	% $R_t^{(FB)}$	Codeword Error Rate ϵ_{FF}
Regular →	1	3	1.333	15	95.0%	7.09×10^{-4}
Irregular	0.531	3.398	1.177	20	96.8%	7.82×10^{-4}
Irregular	0.244	3.699	1.081	30	98.0%	8.35×10^{-4}
Irregular	0.168	3.788	1.056	40	98.3%	8.50×10^{-4}
Irregular	0.139	3.823	1.046	50	98.4%	8.56×10^{-4}
Irregular	0.108	3.861	1.036	100	98.6%	8.63×10^{-4}

Asymptotic Performance of 2-degree Quasi-regular Right Degree Distribution from Density Evolution



UCLA

Non-binary Low Density Parity Check (NB-LDPC) Code as VL Code [Vakilinia et al. ISIT 2014]

VL encoder

k=24 symbols, n=32, Rate-3/4, GF(256) NB-LDPC [Davey et al. Comm. Lett. 1998] with bit-by-bit incremental redundancy (IR)

Channel

Binary-input AWGN Channel with SNR=2 dB

VL decoder

Fast Fourier transformation based Q-ary sum product algorithm (FFT-QSPA) [MacKay et al. IMA 2000] with IR

NB-LDPC VL Code parameters with Constant-size Increments

-

$\boldsymbol{\ell_0}$	$\boldsymbol{\ell_\Delta}$ **for** $\boldsymbol{m = 5}$ **(four increments)**	**Throughput Rate** $R_t^{(FB)}$ **with ACK/NACK Feedback**	**Percentage of Capacity of 2dB BI-AWGN**
302 bits	36 bits	0.5705	88.85%

- $\boldsymbol{\delta} = \{0.309, 0.464, 0.194, 0.0293, 0.00318, 0.00049\}$

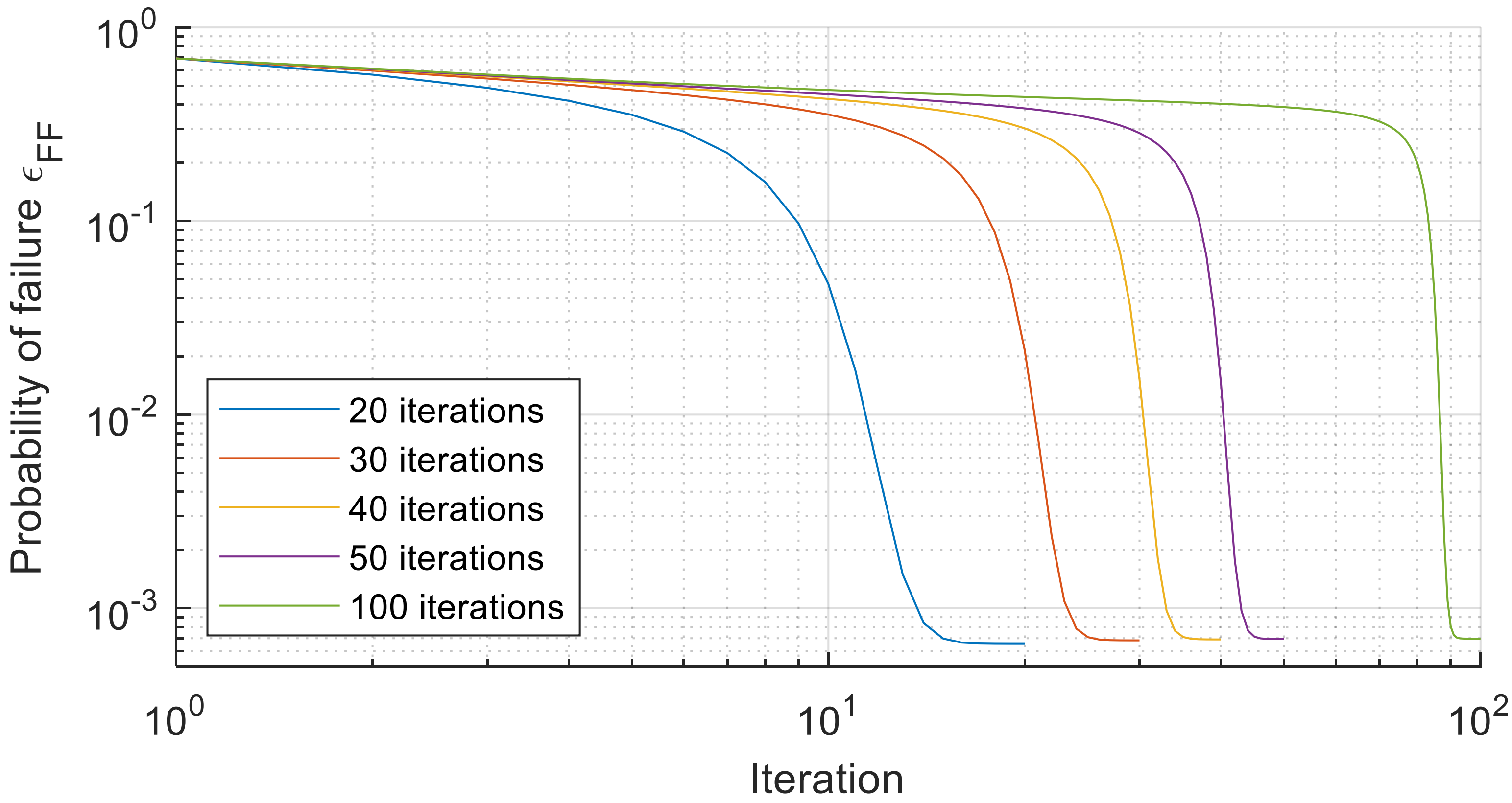
UCLA

Asymptotic Performance of 2-degree Quasi-regular Right Degree Distribution from Density Evolution

- $\lambda(x) = x^3, \rho(x) = \alpha x^2 + (1-\alpha)x^3$, assume infinite large bipartite graph

α	a_R	β_{FF}	No. Iterations	% $R_t^{(FB)}$	Codeword Error Rate ϵ_{FF}
0.597	3.336	1.199	20	97.4%	6.55×10^{-4}
0.341	3.591	1.114	30	98.3%	6.82×10^{-4}
0.273	3.666	1.091	40	98.5%	6.90×10^{-4}
0.246	3.697	1.082	50	98.6%	6.93×10^{-4}
0.217	3.730	1.072	100	98.7%	6.97×10^{-4}

Asymptotic Performance of 2-degree Quasi-regular Right Degree Distribution from Density Evolution



UCLA

Practical Constraints When Designing An Inter-frame Code

- Complexity: Number of systematic nodes n_c
- Error Performance: Probability of error ϵ_{FF}
- Latency: Number of iterations

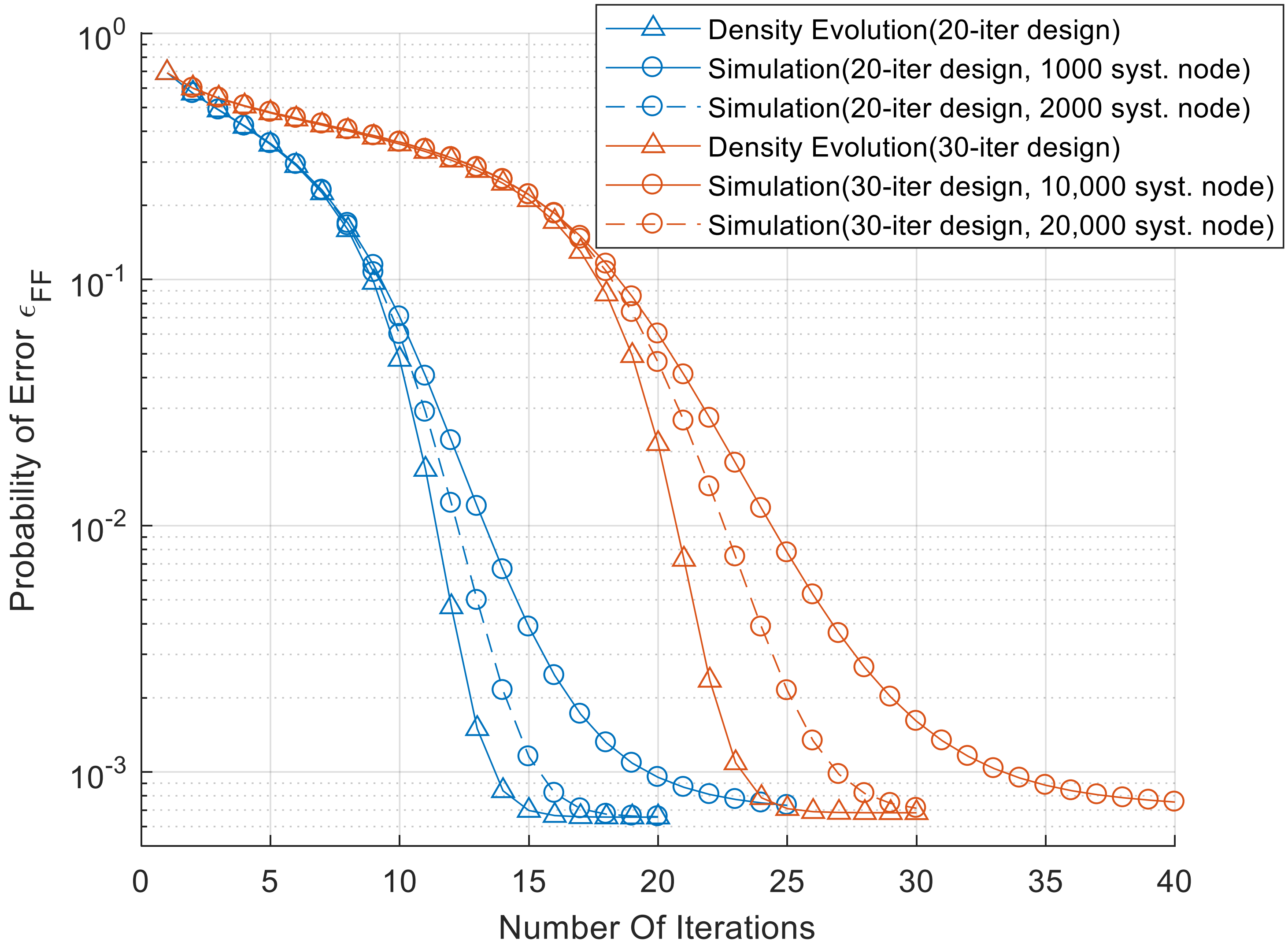
Trade-offs among the constraints!

Number of Systematic Nodes Required of 2-degree Quasi-regular Right Degree Distribution

- $\lambda(x) = x^3, \rho(x) = \alpha x^2 + (1-\alpha)x^3$, 100 inter-frame code iterations

	$\boldsymbol{\alpha = 0.597}$	$\boldsymbol{\alpha = 0.341}$
Throughput rate	0.5559	0.5609
Percentage of Capacity of 2dB BI-AWGN	86.57%	87.35%
Number of Systematic Nodes Needed to Achieve the Designed Throughput Rate	1000	10,000

Probability of Error for Different Designs Requiring Varying Number of Systematic Nodes



UCLA

Conclusions

- VL codes with ACK/NACK feedback can approach capacity with short blocklengths.
- We used many short blocklength VL codes in parallel *without feedback* to achieve 98% of throughput of the underlying VL codes with feedback.
- Inter-frame coding enables a distributed decoding architecture for very high throughputs.

Projects

- Reliability/Latency/Throughput: ECC Parallelization and Incremental Redundancy

- Lifetime: Channel Estimation and Write Voltage Optimization

Histogram-Based Flash Channel Estimation and Dynamic Voltage Allocation

Haobo Wang, Tsung-Yi Chen, Richard D. Wesel

Motivation

- How to reduce Flash memory's wear-out?

Write to lower threshold voltages!

Outline

- Channel Model
- Channel Parameter Estimation
- Dynamic (Write) Voltage Allocation

Channel Model

- We model the NAND flash memory cell data storage process as

$$y = x + n_p + n_w + n_r$$

y : sensed programmed state threshold voltage

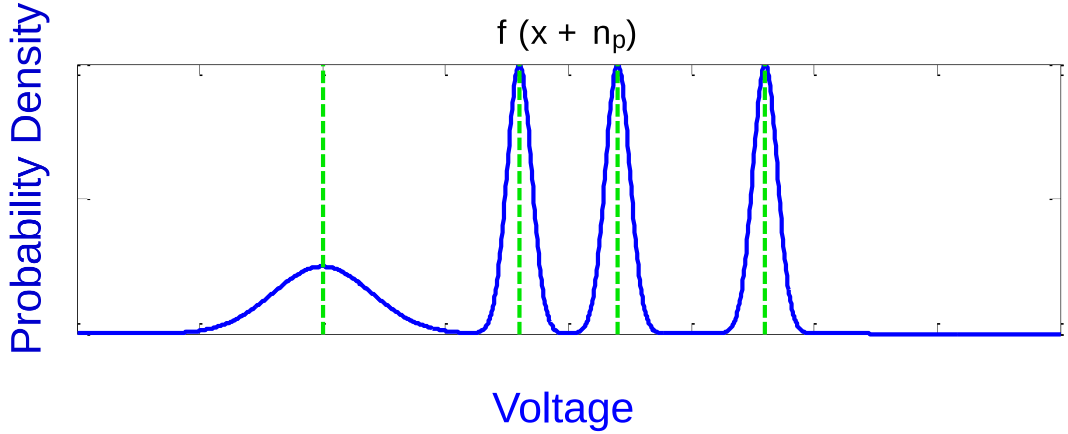
x : intended programmed state threshold voltage

n_p : programming noise

n_w : wear-out noise

n_r : retention noise

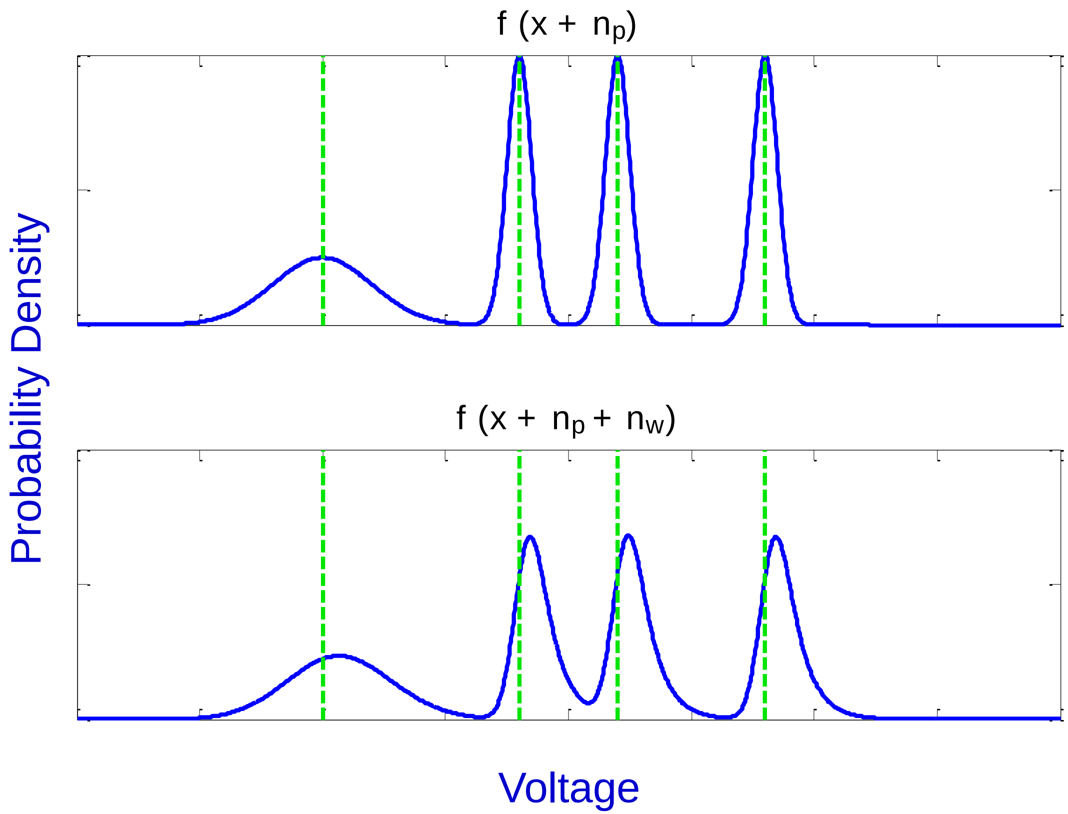
Programming Noise (n_p)



Programming Noise Only $f(n_p)$

$$f(n_p) = \begin{cases} N(0, \sigma_e^2) & \text{if } x = 0 \\ N(0, \sigma_p^2) & \text{if } x > 0 \end{cases} \quad \text{where } \sigma_e > \sigma_p$$

Wear-out Noise (n_w)



Programming Noise Only $f(n_p)$

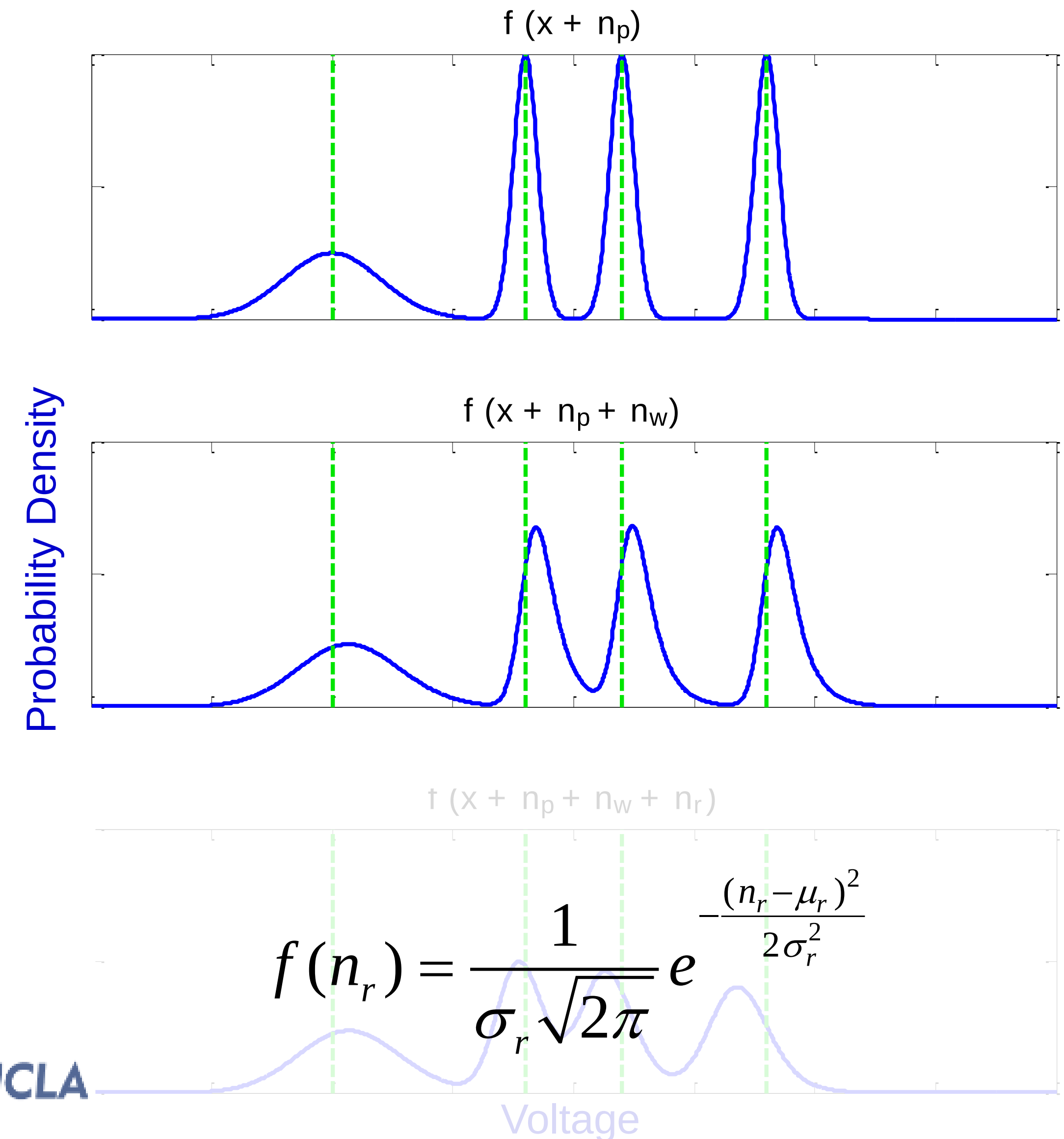
Programming and Wear-out Noise

$f(n_p) \otimes f(n_w)$

$$f(n_w) = \begin{cases} \frac{1}{\lambda} e^{-\frac{n_w}{\lambda}} & n_w \geq 0 \\ 0 & n_w < 0 \end{cases}$$

UCLA

Retention Noise (n_r)



Programming Noise Only
$f(n_p)$

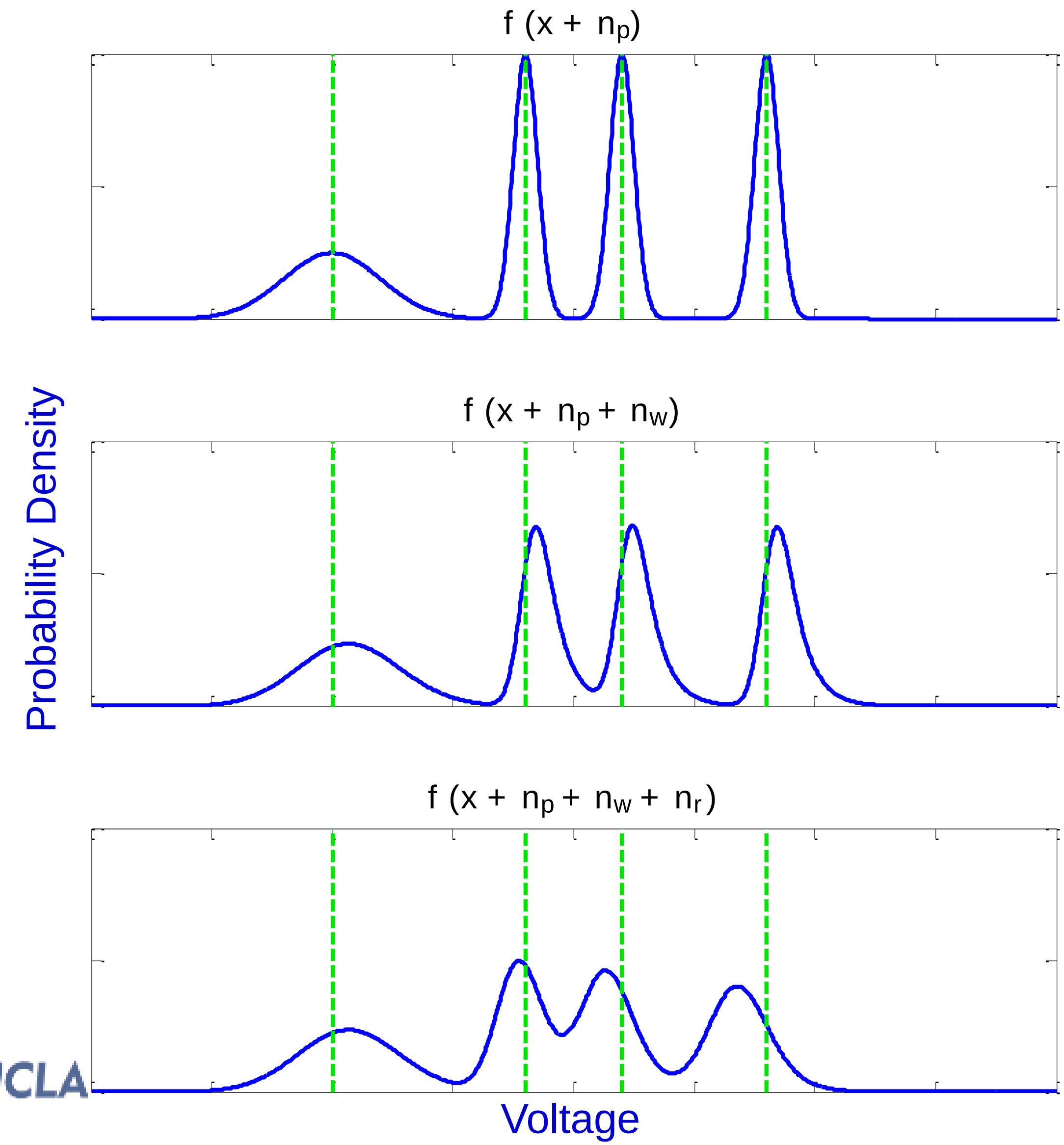
Programming and Wear-out Noise
$f(n_p) \otimes f(n_w)$

Programming, Wear-out and Retention Noise
$f(n_p) \otimes f(n_w) \otimes f(n_r)$

$$f(n_r) = \frac{1}{\sigma_r \sqrt{2\pi}} e^{-\frac{(n_r - \mu_r)^2}{2\sigma_r^2}}$$

UCLA

Sample PDF



Programming Noise Only

$f(n_p)$

Programming and Wear-out Noise

$f(n_p) \otimes f(n_w)$

Programming, Wear-out and Retention Noise

$f(n_p) \otimes f(n_w) \otimes f(n_r)$

UCLA

Replace number of P/E cycles with accumulated voltage

- Channel degradation is usually modeled as a function of the number of program/erase (P/E) cycles.

Replace number of P/E cycles with accumulated voltage

- Channel degradation is usually modeled as a function of the number of program/erase (P/E) cycles.

- The volume of charge passing through dielectrics actually causes the degradation.

Replace number of P/E cycles with accumulated voltage

- Channel degradation is usually modeled as a function of the number of program/erase (P/E) cycles.
- The volume of charge passing through dielectrics actually causes the degradation.
- The use of the number of P/E cycles is approximately correct when the volume of charge passing through the dielectrics is the same for each P/E cycle.

Replace number of P/E cycles with accumulated voltage

- Channel degradation is usually modeled as a function of the number of program/erase (P/E) cycles.
- The volume of charge passing through dielectrics actually causes the degradation.
- The use of the number of P/E cycles is approximately correct when the volume of charge passing through the dielectrics is the same for each P/E cycle.
- We use a more precise metric named accumulated voltage Vacc to directly characterize the volume of charge that has passed since the first write.

Accumulated Voltage

$$V_{acc} = \sum_{j=1}^{N} \left(V_p^{(j)} - V_e \right)$$

V_{acc} : accumulated voltage over N P/E cycles,

$V_p^{(j)}$: programmed threshold voltage of the jth P/E cycle

V_e : threshold voltage of the erased state

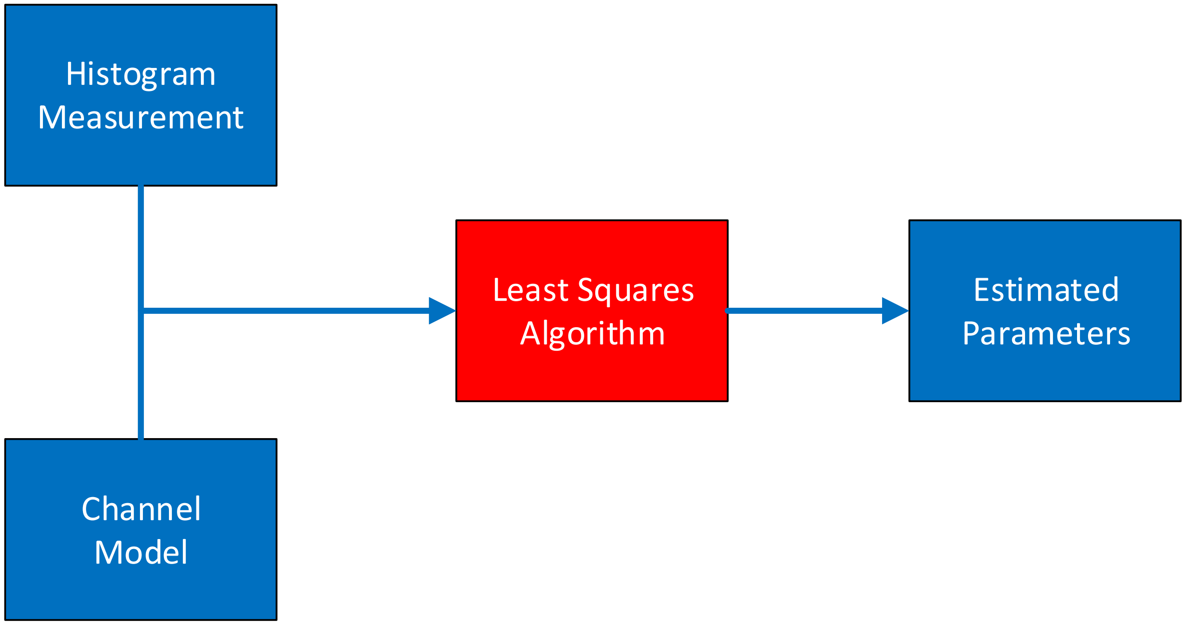
The normalized accumulated voltage is $V_{acc} / V_{\max}$, where $V_{\max}$ is the maximum of $V_p^{(j)} - V_e, \forall j.$

When using fixed voltage levels, $V_{acc} / V_{\max}$ ≈ # PE Cycles.

UCLA

Channel Parameter Estimation

- Channel parameter estimation workflow:



Parameter Vector

- $[\lambda, \sigma_{\text{programming}}, \sigma_{\text{erase}}, \sigma_{\text{retention}}, \mu_{\text{retention}}]$

- We actually estimate $[\lambda, \sigma_p, \sigma_e, m_r, n_r]$, where

$$\mu_{\text{retention}} = (x - x_0) \cdot n_r$$
$$\sigma^2_{\text{retention}} = (x - x_0) \cdot m_r^2 \quad .$$

UCLA

Estimation Objective Function

- Estimation Objective Function is the squared Euclidean distance between the predicted histogram and measured histogram

$$C_M = \sum_{i=0}^{M-1} \left(\frac{\hat{N}_{\mathrm{bin,i}} - N_{\mathrm{bin,i}}}{N} \right)^2$$

N : total number of cells in a page

$N_{\mathrm{bin,i}}$: total number of cells in ith bin of measure histogram

$\hat{N}_{\mathrm{bin,i}}$: total number of cells in ith bin by estimation

M : total number of bins

More about Least Squares Algorithms

- Objective
  - Minimize the cost function: $C_M = \sum_{i=0}^{M-1} \left(\frac{\hat{N}_{\text{bin,i}} - N_{\text{bin,i}}}{N} \right)^2$

More about Least Squares Algorithms

- Objective
  - Minimize the cost function: $C_M = \sum_{i=0}^{M-1} \left(\frac{\hat{N}_{\mathrm{bin,i}} - N_{\mathrm{bin,i}}}{N} \right)^2$
- Algorithm 1 – Gradient Descent
  - Follow the descending gradient with a fixed step size.

UCLA

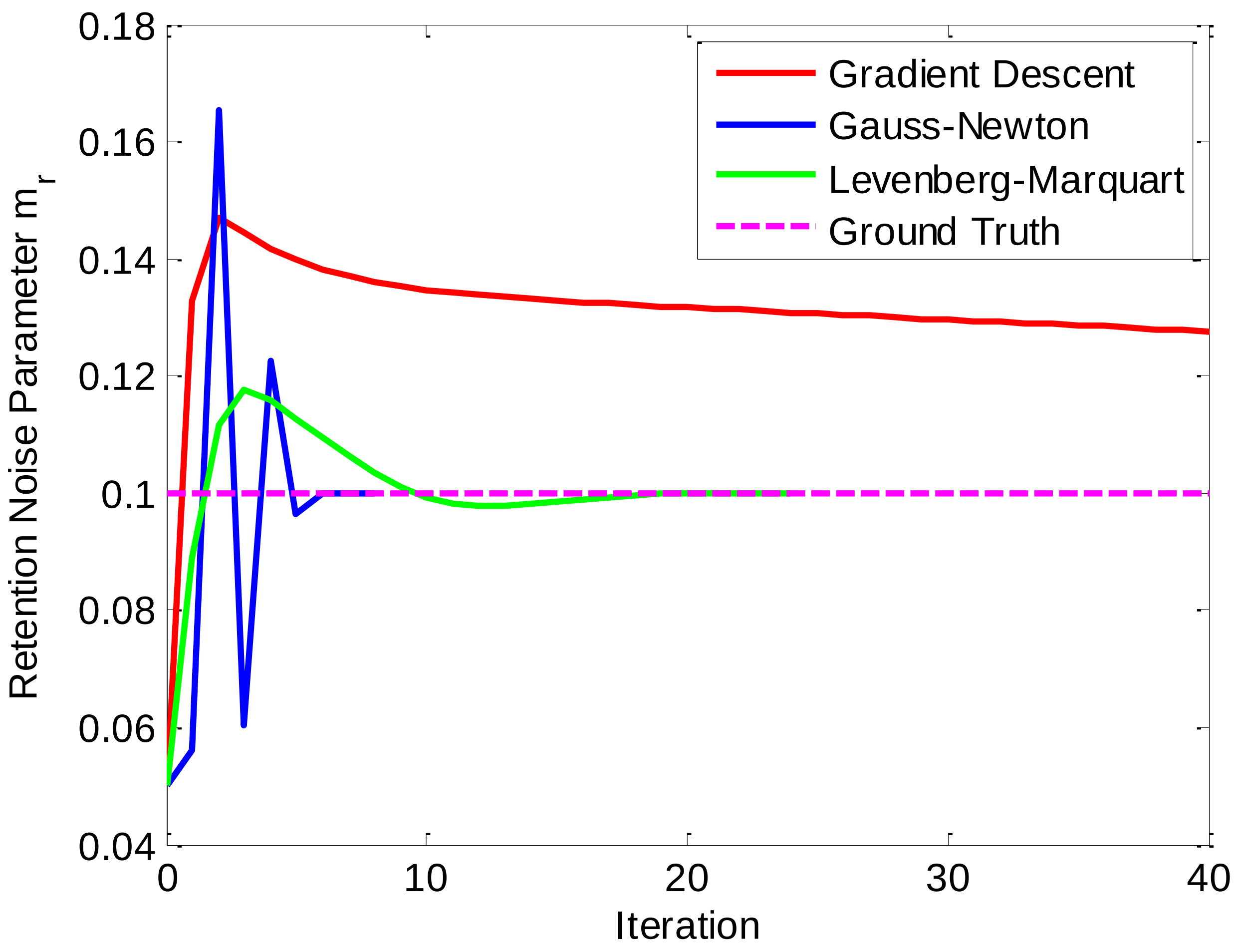
More about Least Squares Algorithms

- Objective
  - Minimize the cost function: $C_M = \sum_{i=0}^{M-1} \left(\frac{\hat{N}_{\text{bin,i}} - N_{\text{bin,i}}}{N} \right)^2$
- Algorithm 1 – Gradient Descent
  - Follow the descending gradient with a fixed step size.
- Algorithm 2 – Gauss–Newton Algorithm
  - Take each step based on quadratic approximation at current point.

More about Least Squares Algorithms

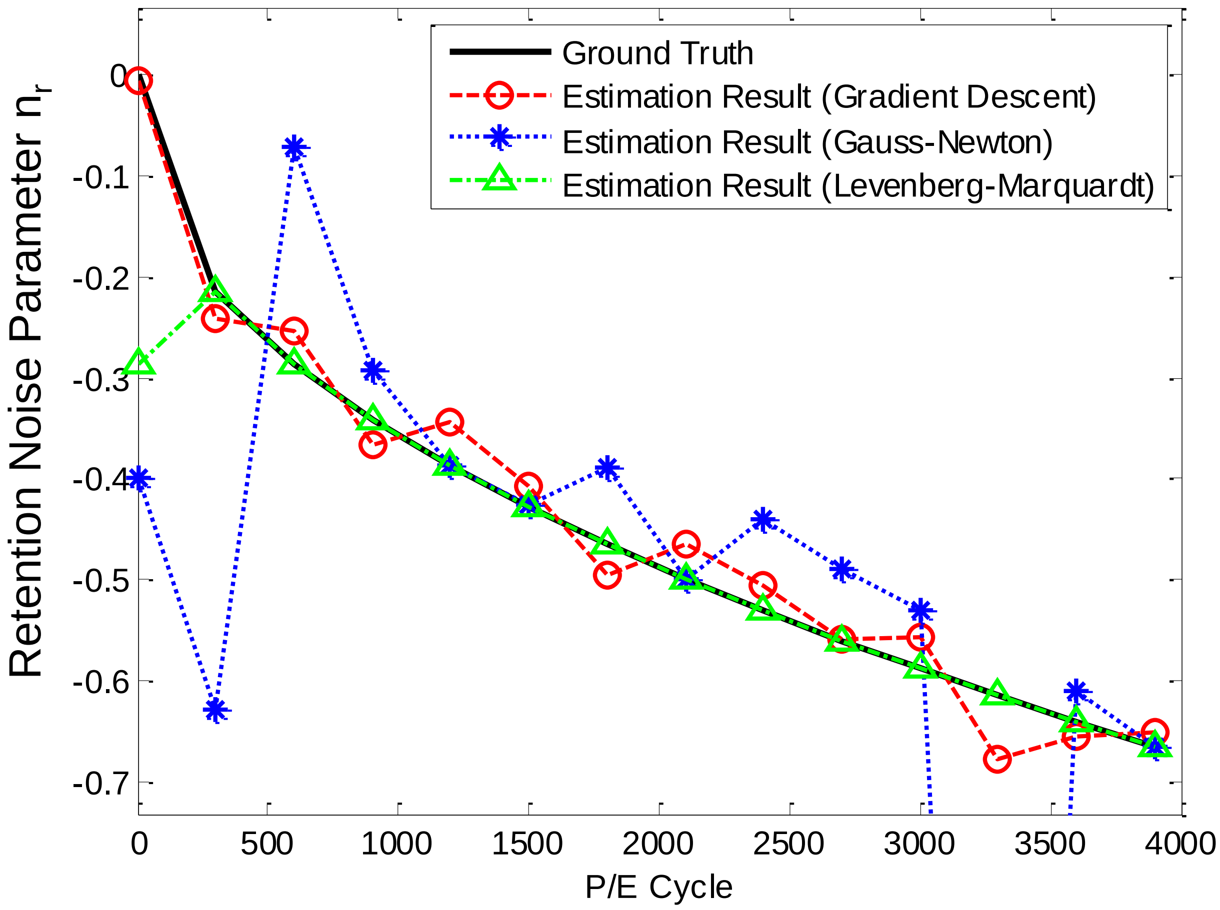
- Objective
  - Minimize the cost function: $$C_M = \sum_{i=0}^{M-1} \left(\frac{\hat{N}_{\text{bin,i}} - N_{\text{bin,i}}}{N} \right)^2$$
- Algorithm 1 – Gradient Descent
  - Follow the descending gradient with a fixed step size.
- Algorithm 2 – Gauss–Newton Algorithm
  - Take each step based on quadratic approximation at current point.
- Algorithm 3 – Levenberg–Marquardt Algorithm
  - Rotate Gauss-Newton increment vector toward the direction of descending gradient.

Least Squares Algorithm Speed Comparison



UCLA

Least Squares Algorithm Accuracy Comparison



UCLA

Least Squares Algorithms Choice

- Algorithm 1 – Gradient Descent
  - Convergence speed is too slow.
- Algorithm 2 – Gauss–Newton Algorithm
  - Converge fast but lacks stability.
- Algorithm 3 – Levenberg–Marquardt Algorithm
  - Good for parameter estimation.

Binning Strategy

- Bin-placement Paradigm
- Number of Bins

UCLA

Bin-placement Paradigm

- Equal Interval (EI) Histogram
  - Not actually equal. Bins covering erased state distribution can be slightly wider.

- Maximum Mutual Information (MMI) Histogram
  - Bins optimized for decoding.

- Equal Probability (EP) Histogram
  - Each bin has the same number of cells.

One metric to consider…

- Squared Euclidean Distance between the Channel distribution $f(y)$ and the histogram induced by $f(y)$.

$$D_{E^2} = \sum_{i=0}^{M-1} \int_{q_i}^{q_{i+1}} \left(f(y) - \frac{H_i}{q_{i+1} - q_i} \right)^2 dy$$

$f(y)$: channel distribution

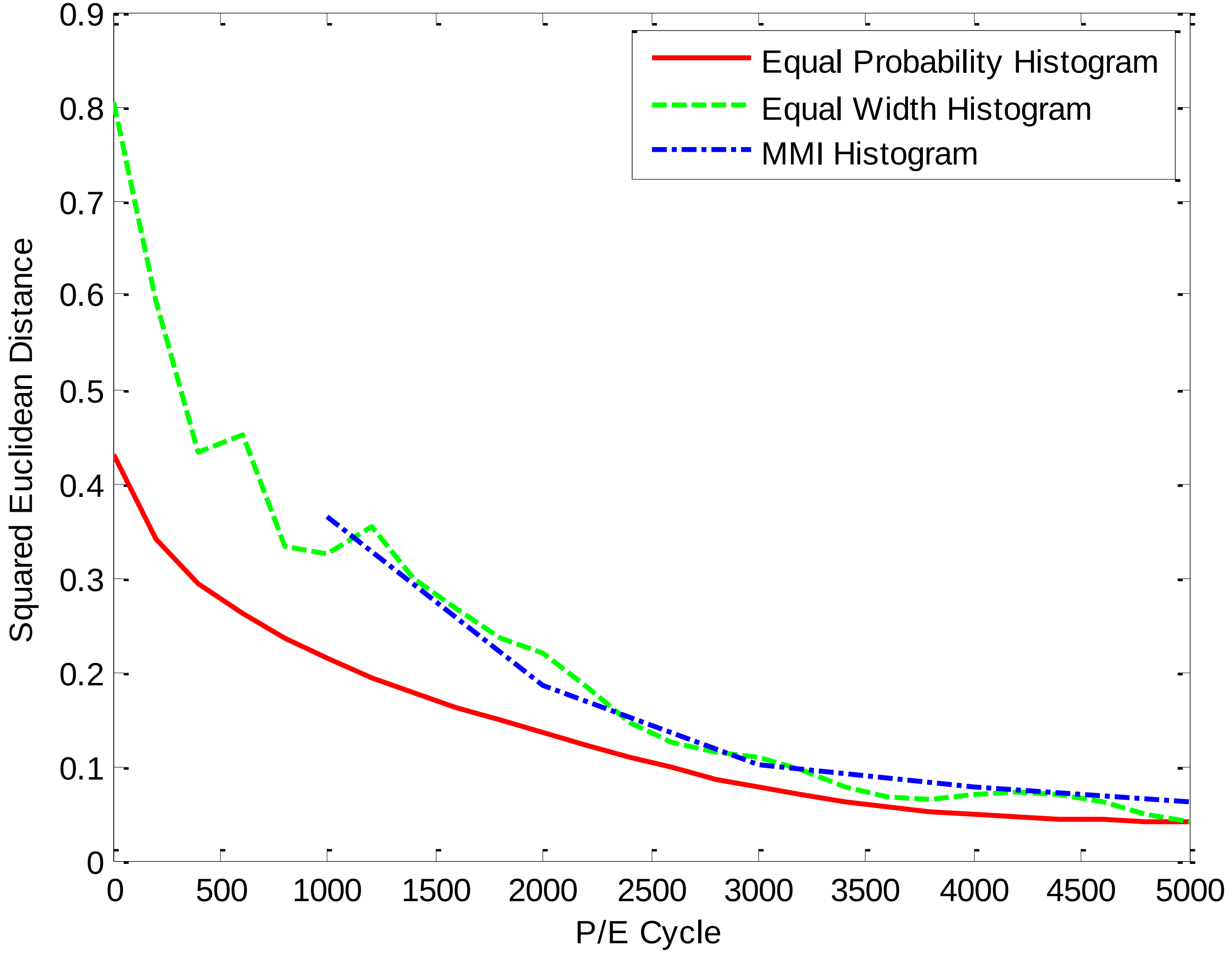
M : number of bins

q_i : left boundary of the ith interval

q_{i+1} : right boundary of the ith interval

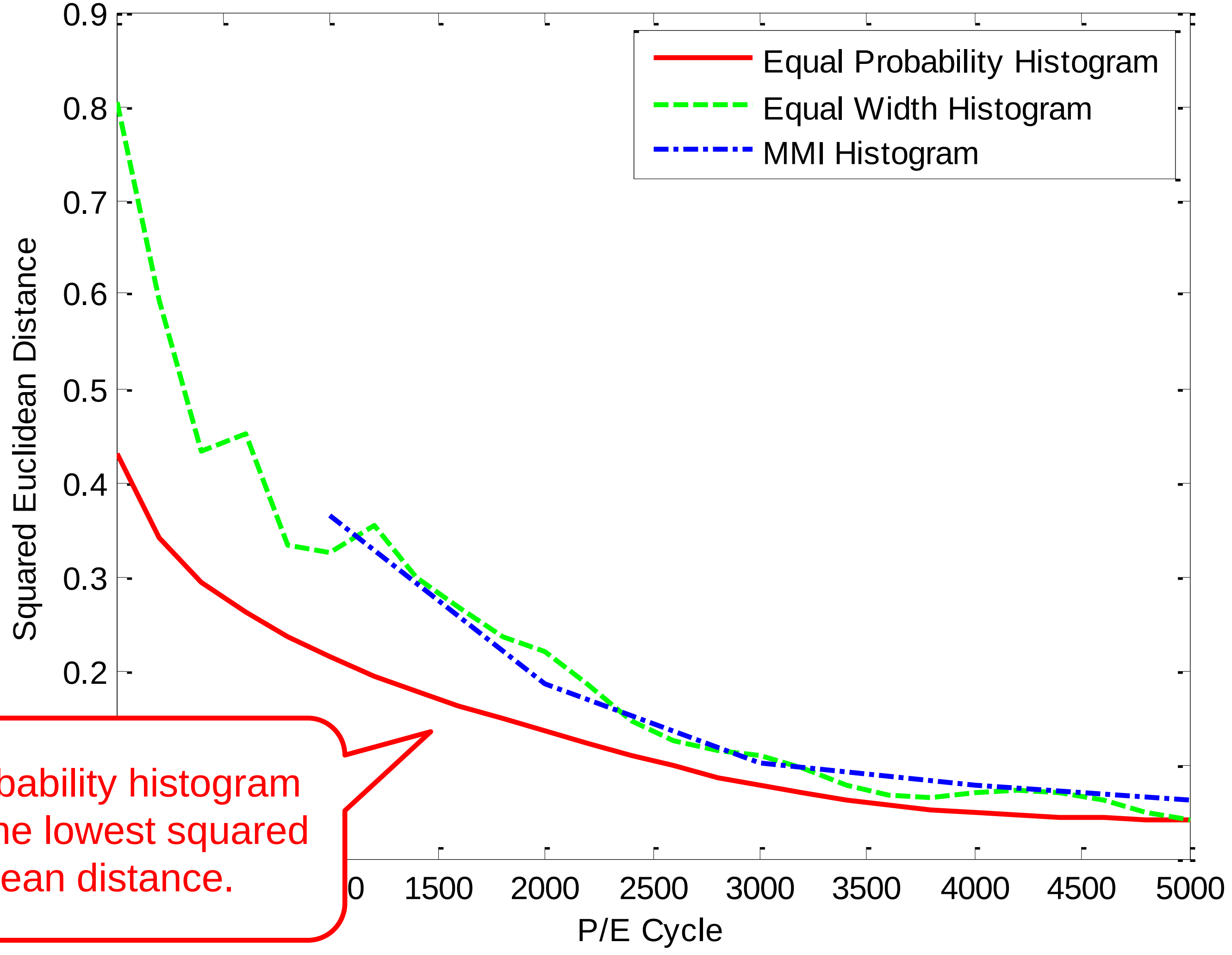
H_i : probability of the ith bin $H_i = \int_{q_i}^{q_{i+1}} f(y)dy$

Square Euclidean Distance Comparison



UCLA

Square Euclidean Distance Comparison



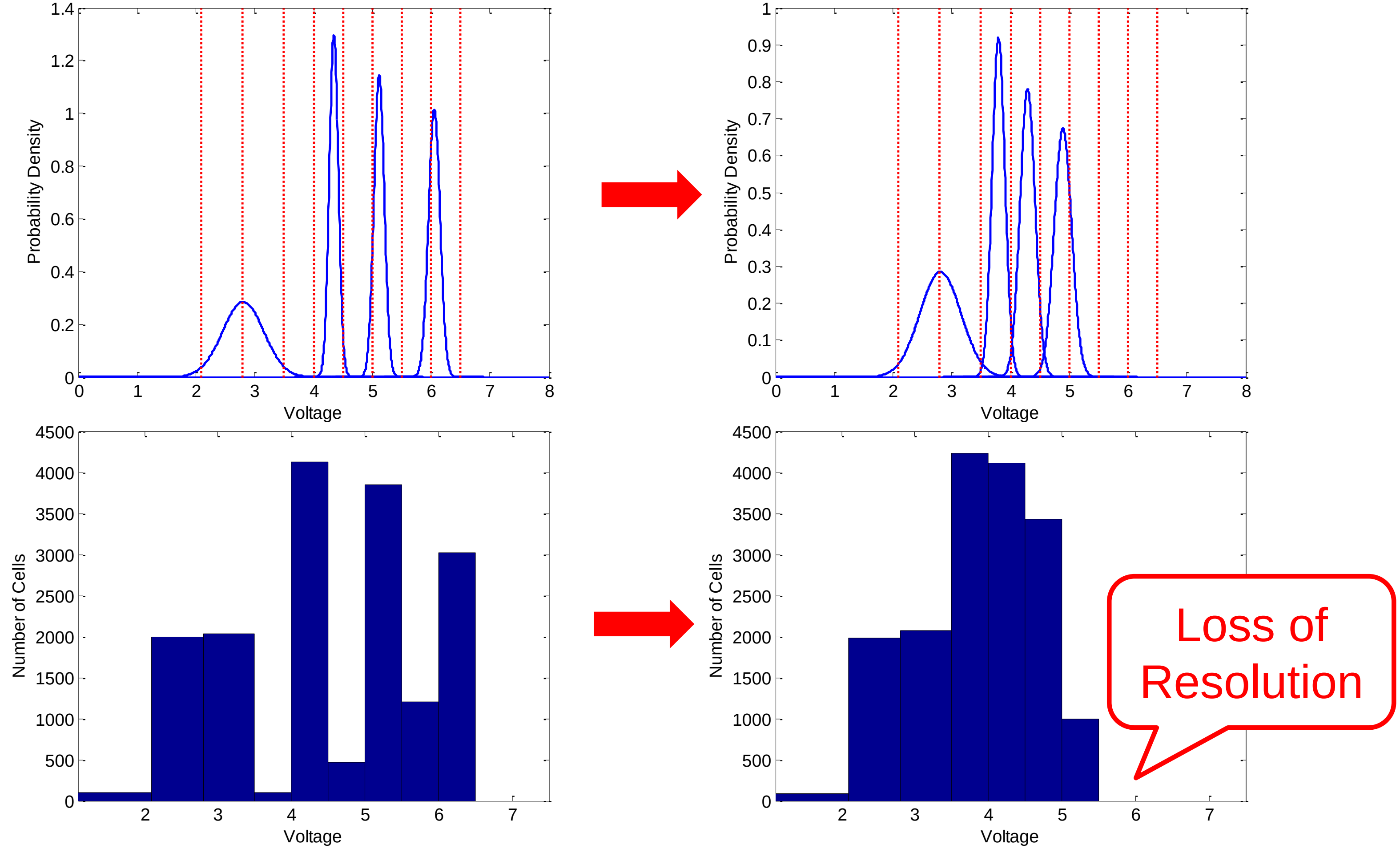
Equal probability histogram provides the lowest squared Euclidean distance.

UCLA

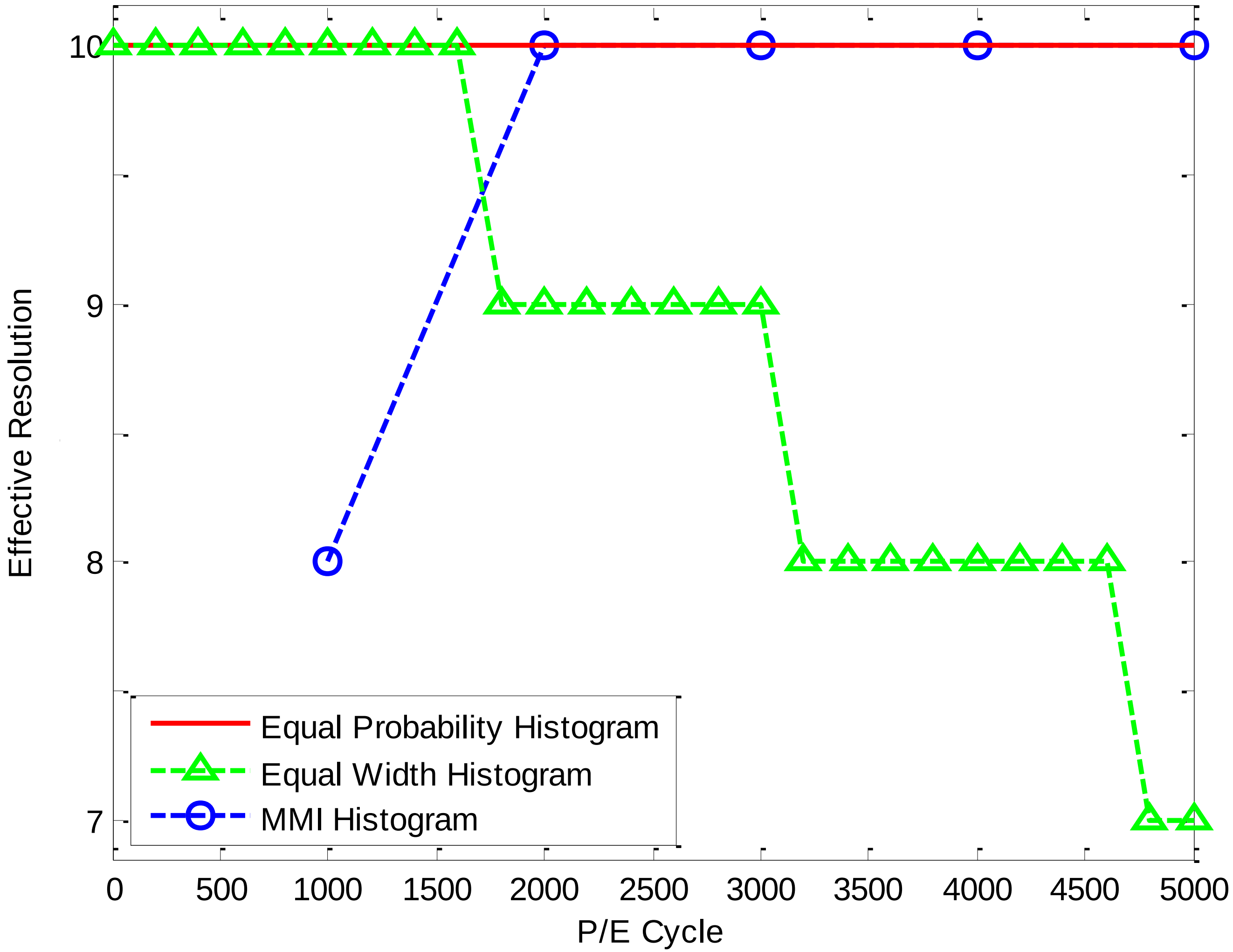
Another metric to consider…

- Effective Resolution
  - Two adjacent zero-height bins can be combined as one bin.
  - Effective resolution is the number of bins after this combination process.

Equal-interval histogram loses resolution with retention effect.

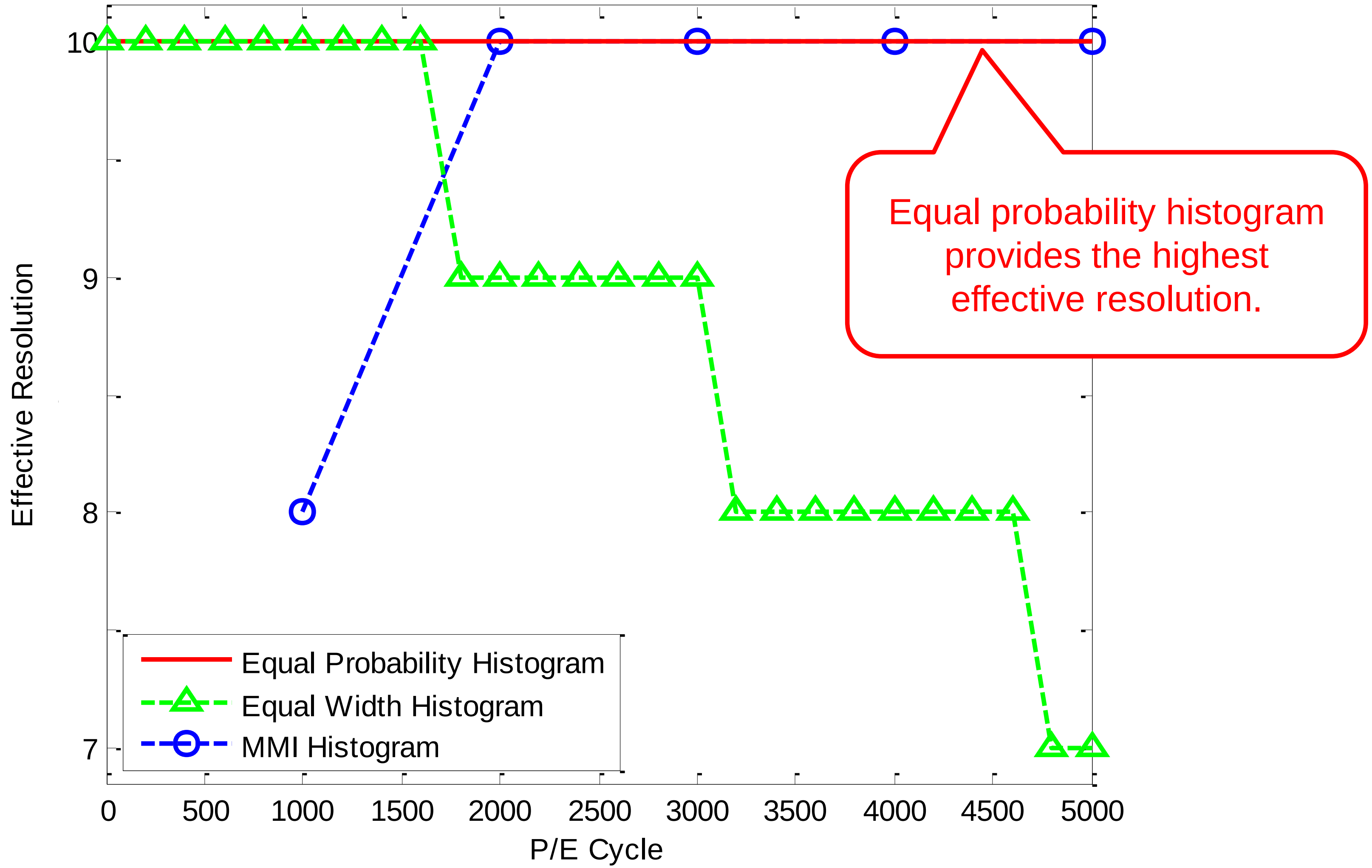


Effective Resolution Comparison



UCLA

Effective Resolution Comparison

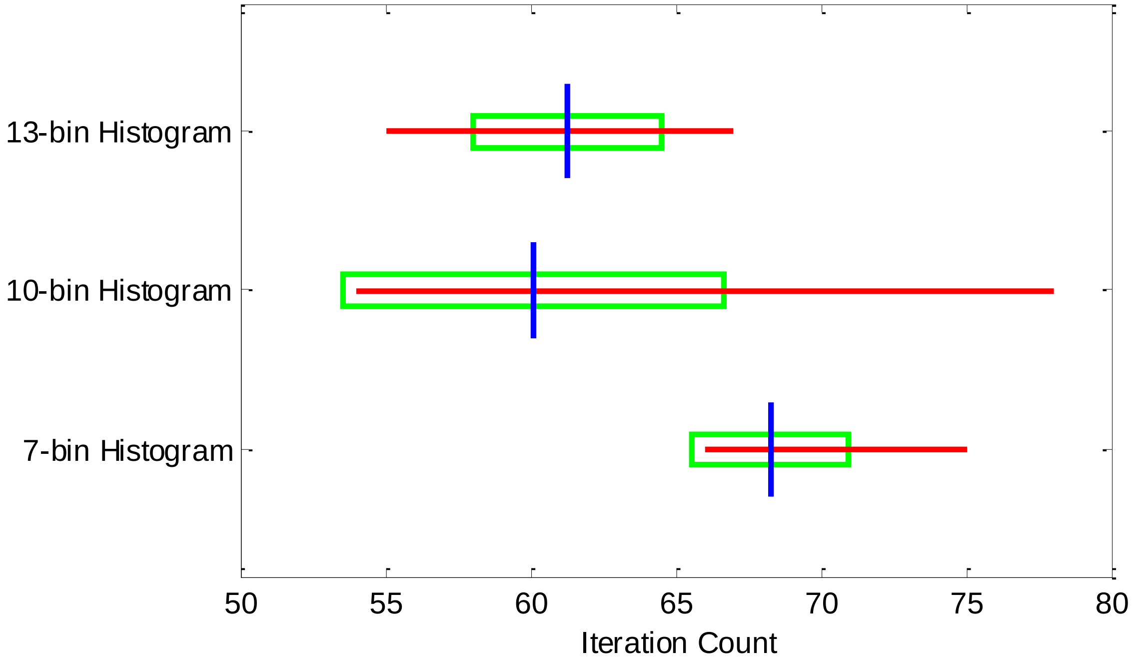


UCLA

Bin-placement Paradigm Choice

- Equal Interval Histogram
  - Equal interval histogram does not adapt well to retention loss.
- Maximum Mutual Information Histogram
  - This histogram optimizes decoder performance, but may not be the best for channel parameter estimation.
- Equal Probability Histogram
  - Every bin has an equal number of cells, good for parameter estimation.

Number of Bins Comparison



Horizontal Line Segment: Range of Iteration Count

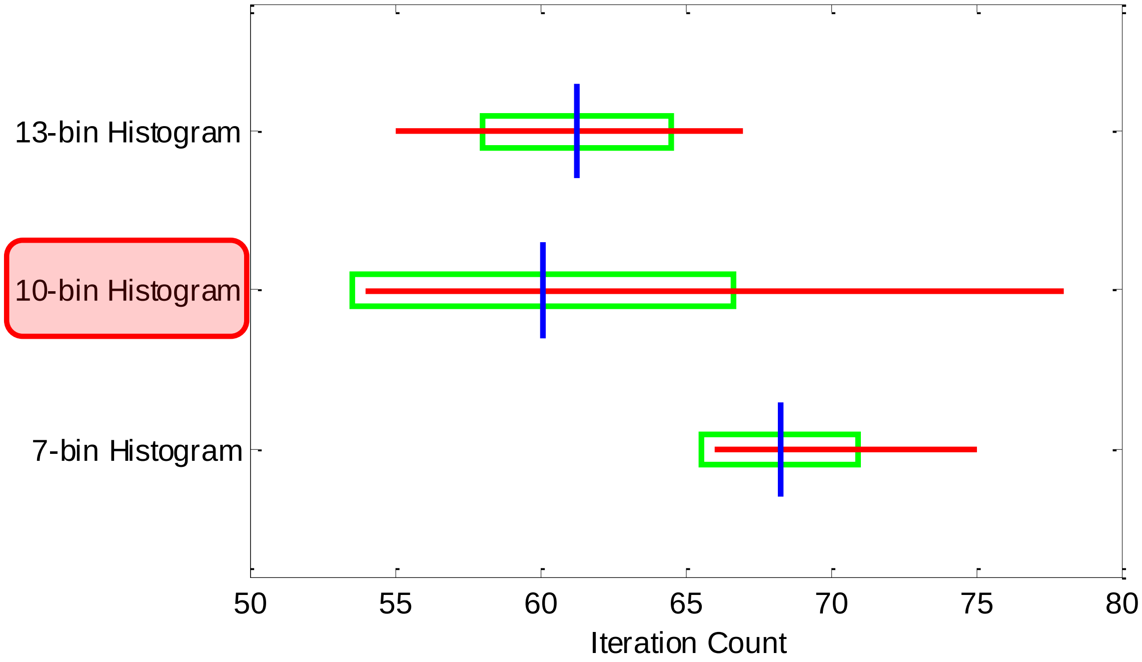
Vertical Line Segment: Mean of Iteration Count

Rectangle: Standard Deviation of Iteration Count

Levenberg-Marquardt Algorithm Iteration Count

UCLA

Number of Bins Comparison



Horizontal Line Segment: Range of Iteration Count

Vertical Line Segment: Mean of Iteration Count

Rectangle: Standard Deviation of Iteration Count

Levenberg-Marquardt Algorithm Iteration Count

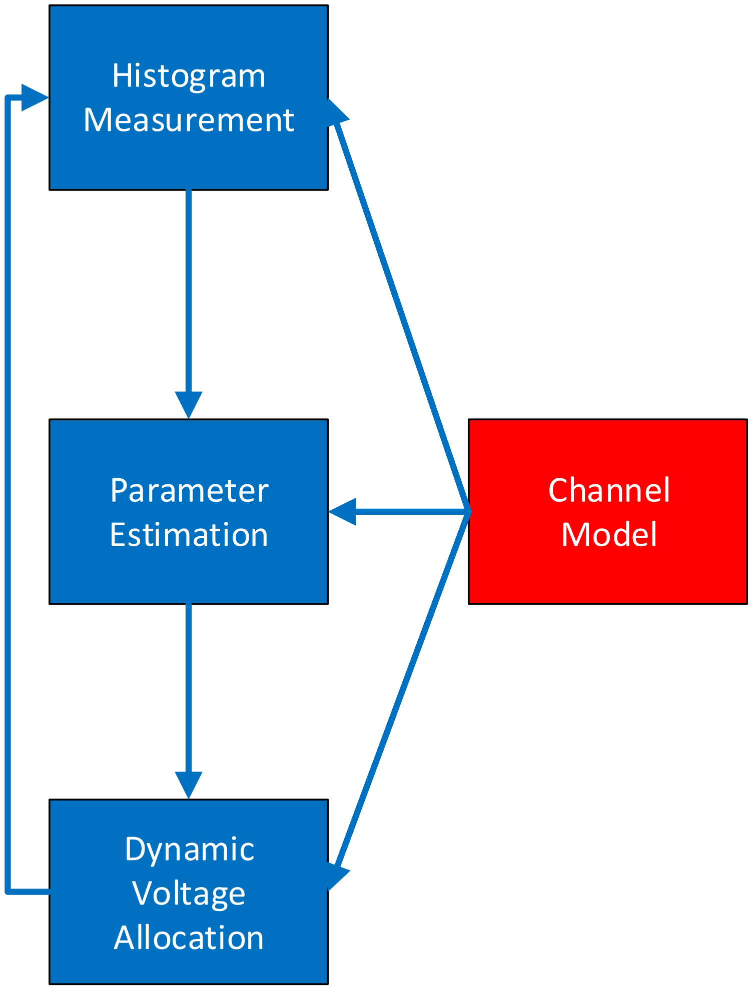
- 10-bin histogram strikes the right balance, which provides sufficient information to narrow the set of possible channels but not so large as to overstrain the optimization algorithm.

UCLA

Dynamic (Write) Voltage Allocation

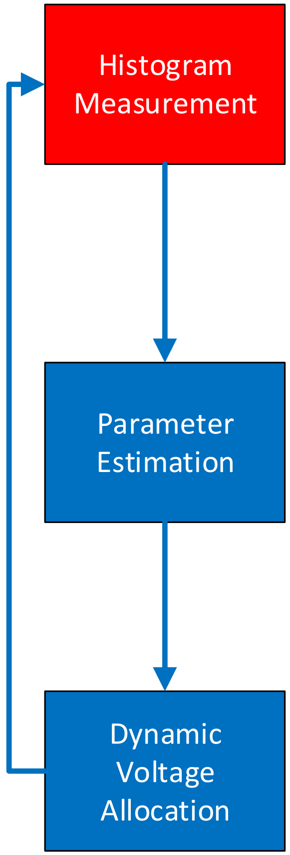
- Fixed threshold voltage allocation provides unnecessary margin at the beginning of Flash memory's lifetime, causing accelerated wear-out.
- Dynamic Voltage Allocation can reduce unnecessary wear-out, and thus increase lifetime by using lower threshold voltages for early writes.
- The threshold voltages can be gradually increased as needed using a single scaling factor to combat channel degradation.
- The target of the anti-degradation process is to maintain a minimum mutual information as long as possible.

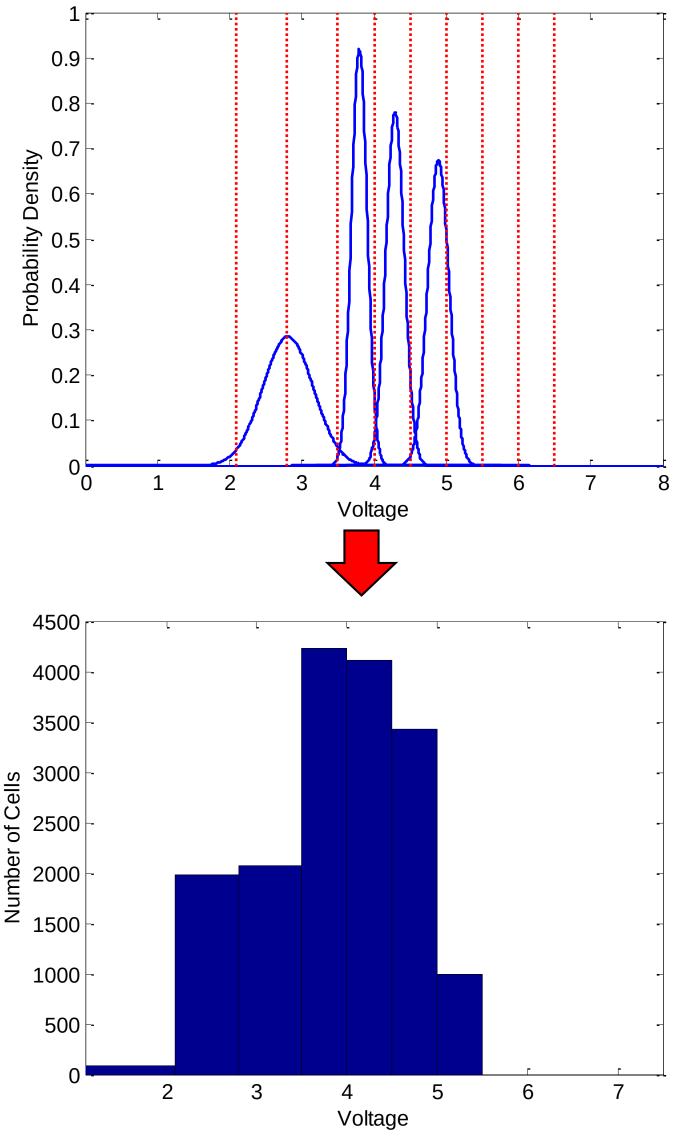
DVA using Histogram-based Channel Estimation



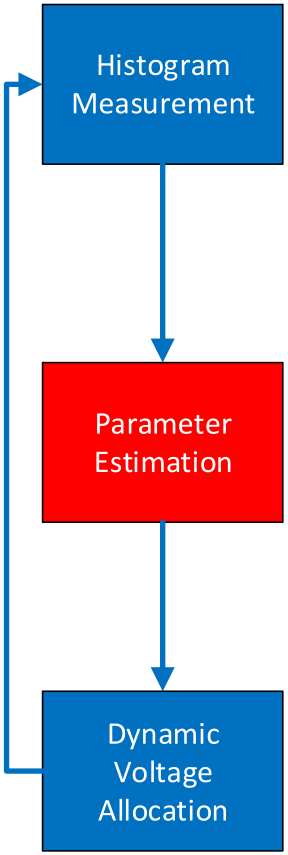
UCLA

Histogram Measurement

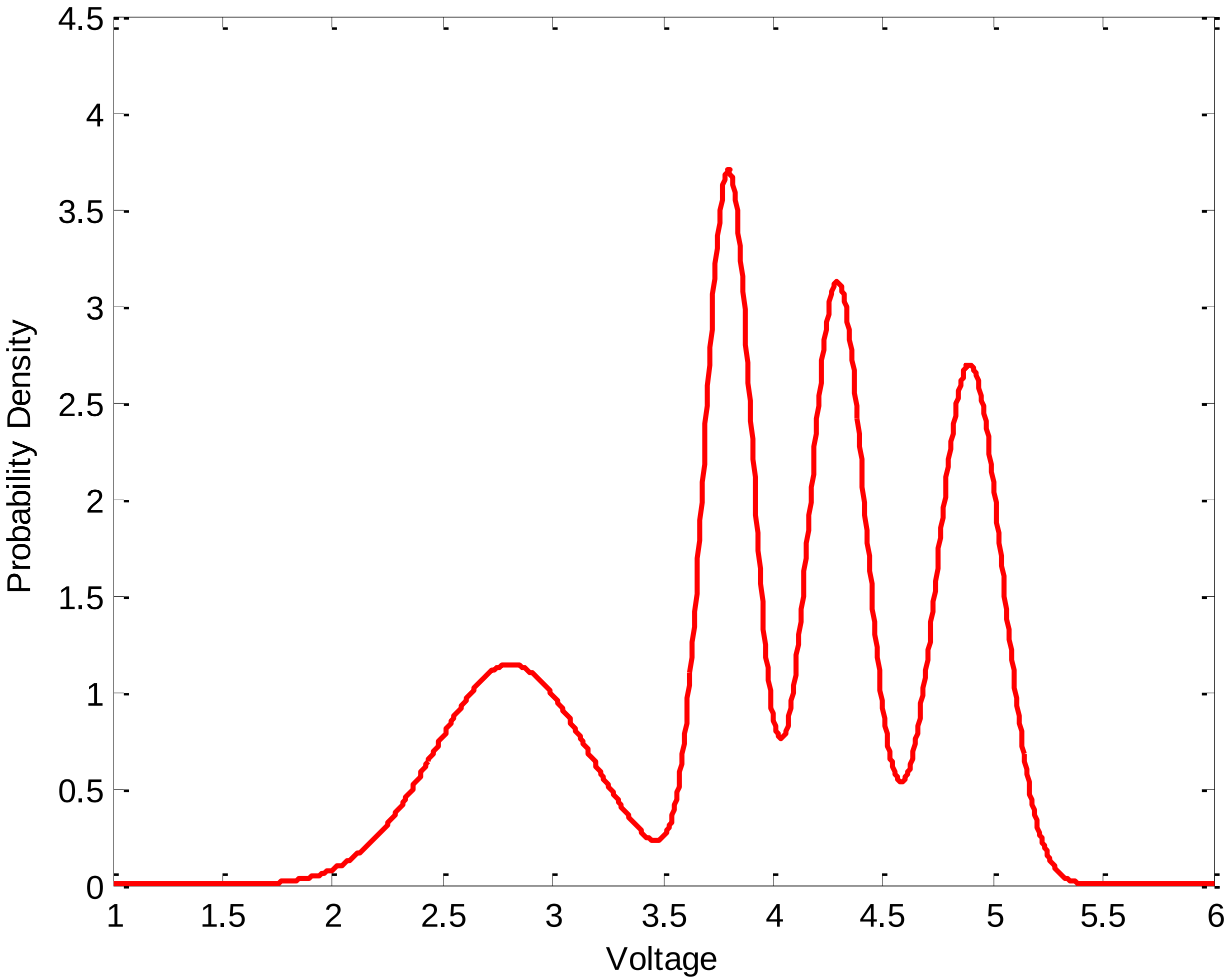




Parameter Estimation

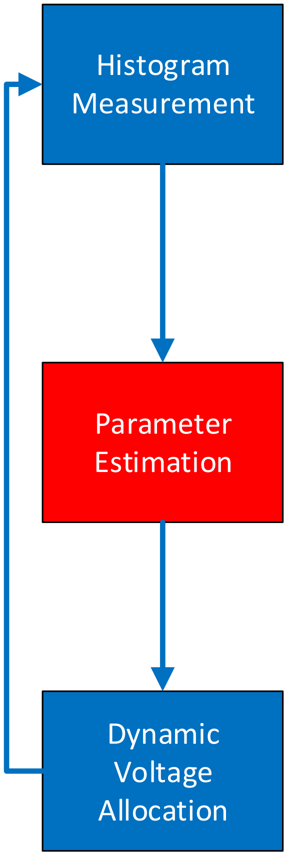


Ground Truth: [0.0099,0.3500,0.0500,0.0617,-0.5882]



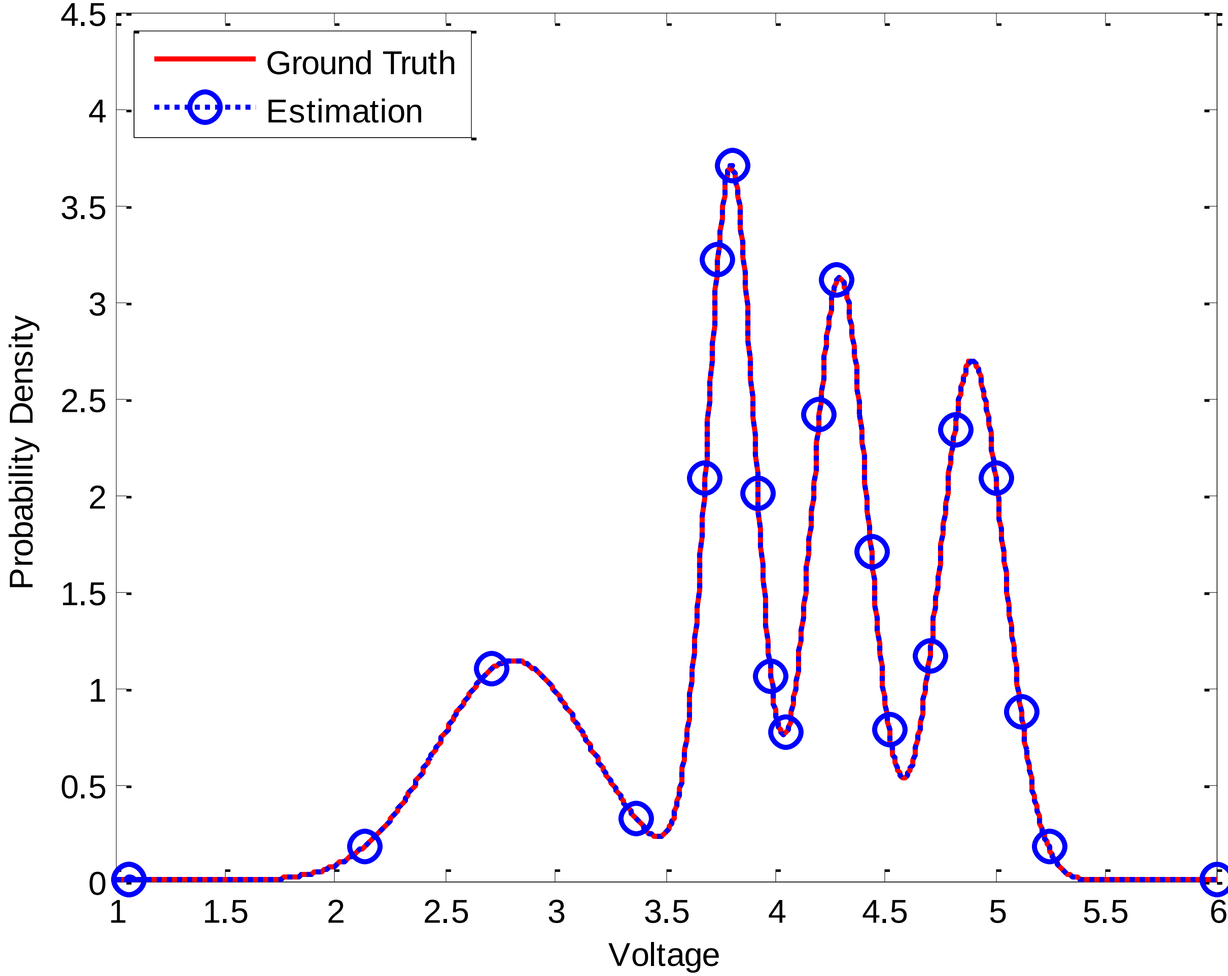
UCLA

Parameter Estimation



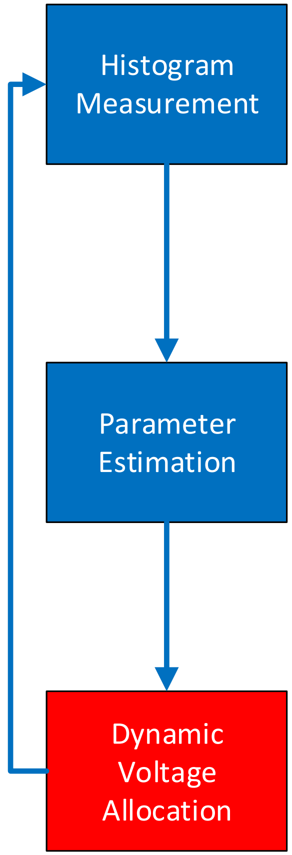
Ground Truth: [0.0099,0.3500,0.0500,0.0617,-0.5882]

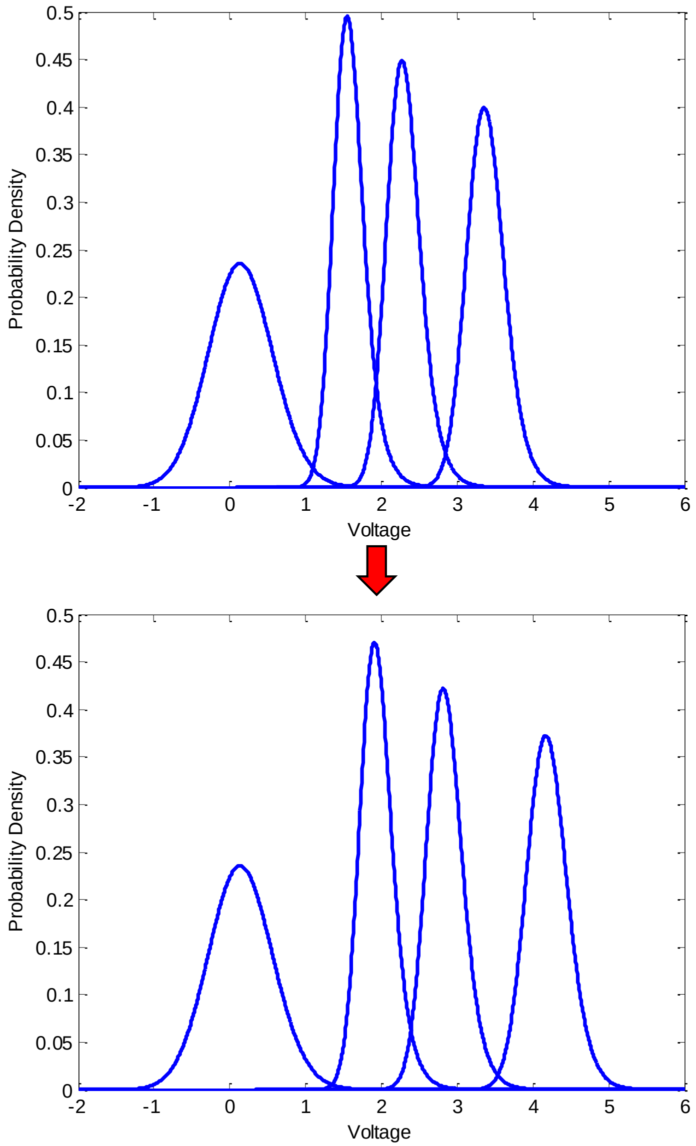
Estimation Error: $10^{-4} \times$ [0.0101 0.0214 -0.1774 0.0405 -0.0044]



UCLA

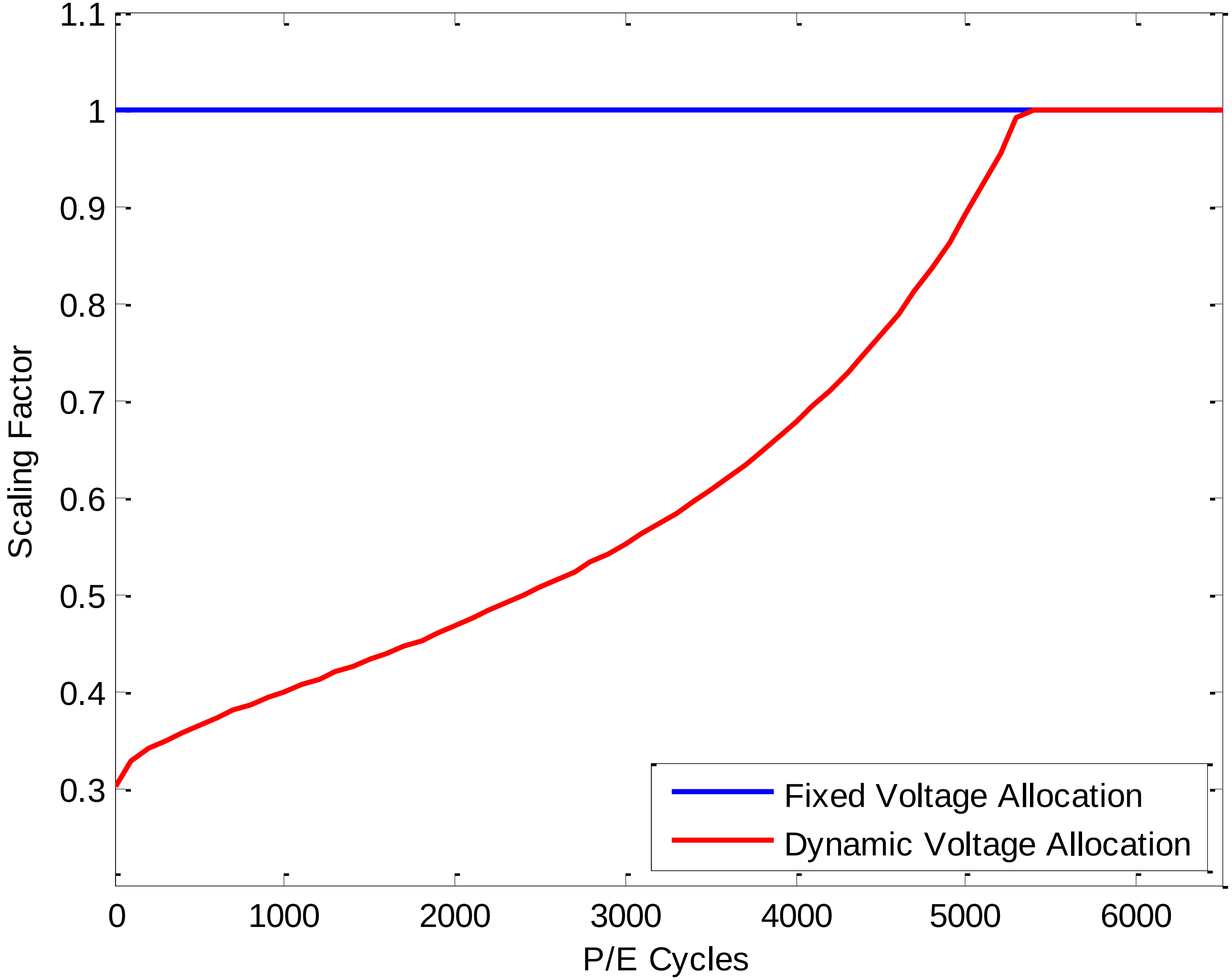
Voltage Levels Adapted to Degraded Channel





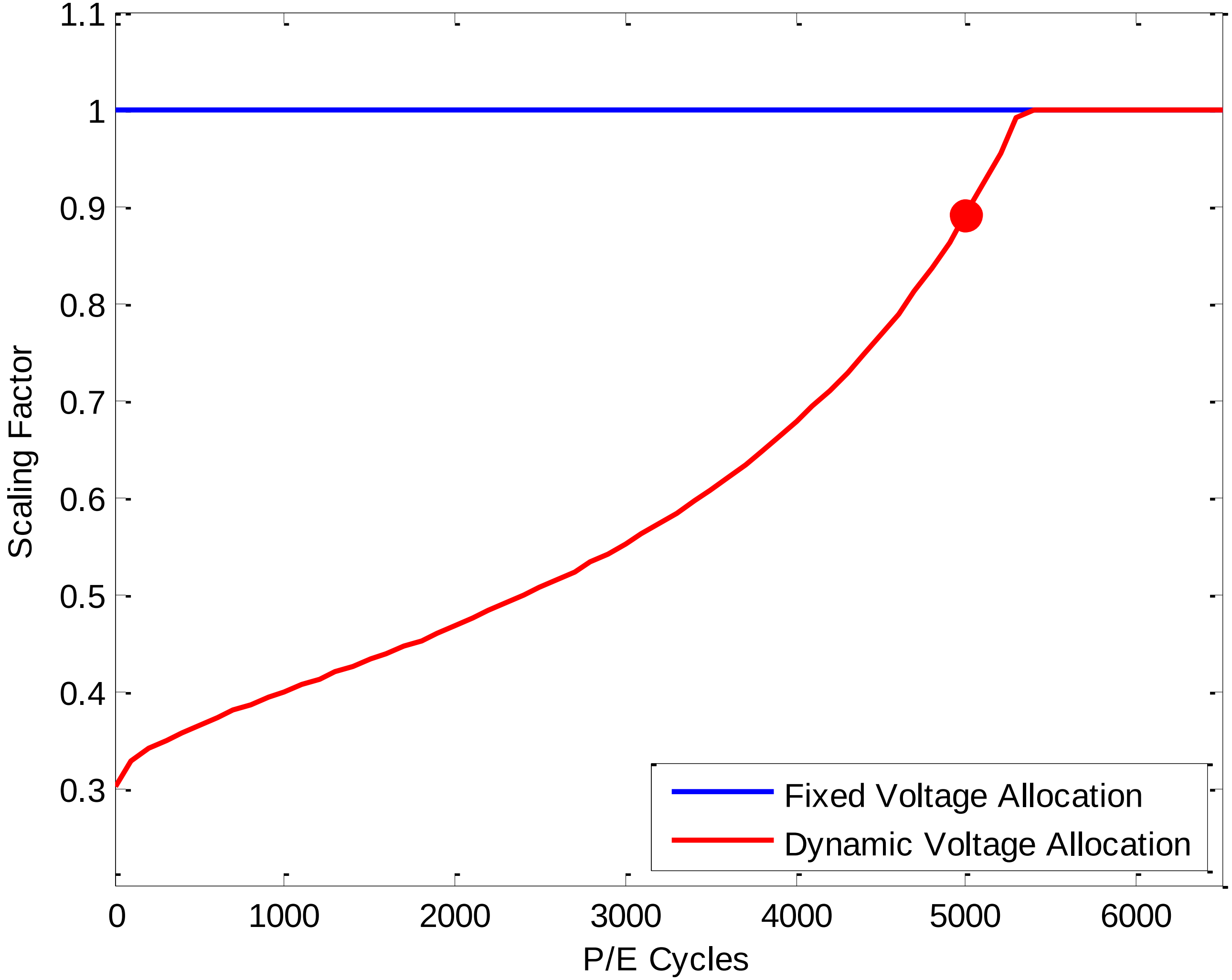
UCLA

Dynamic Voltage Allocation Scaling Factor Example



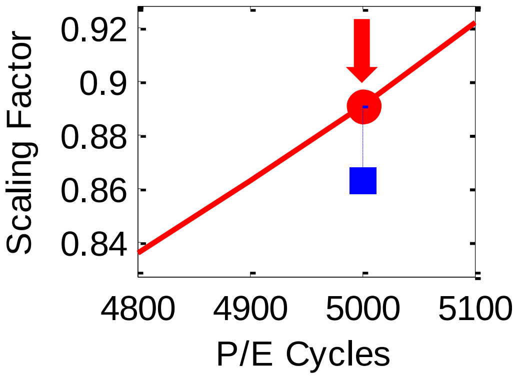
UCLA

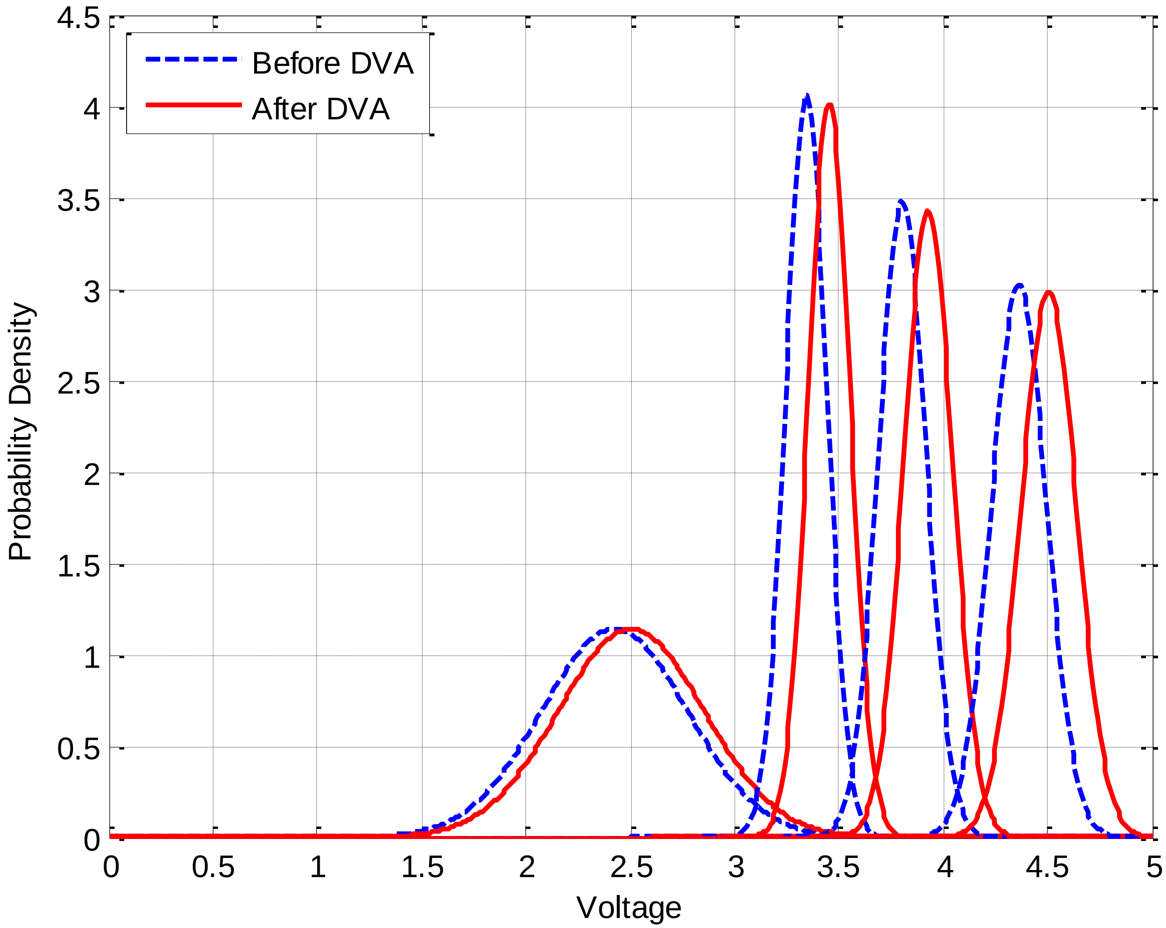
Dynamic Voltage Allocation Scaling Factor Example



UCLA

Dynamic Voltage Allocation Scaling Example (P/E = 5000)

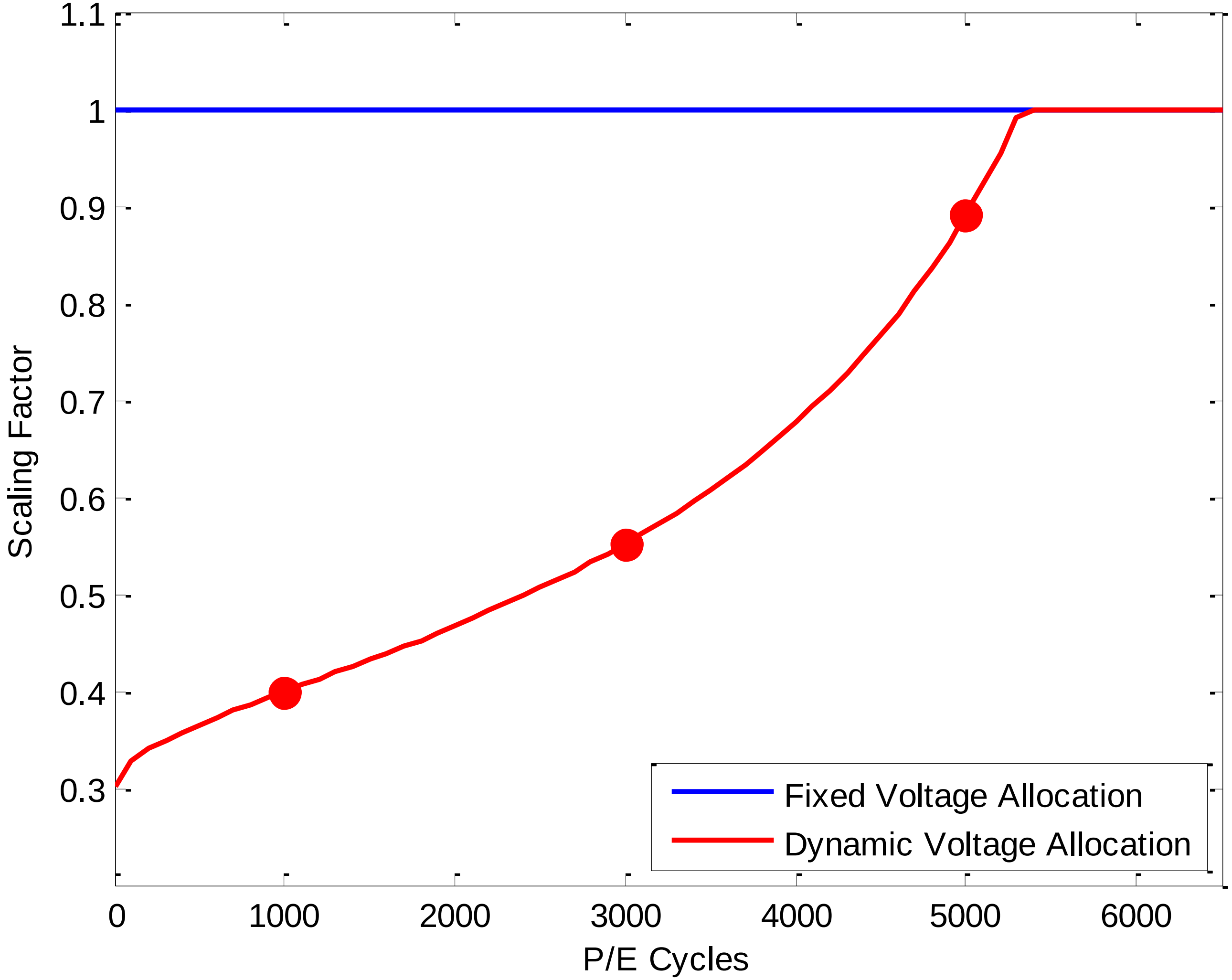




1.905 bits --> 1.935 bits

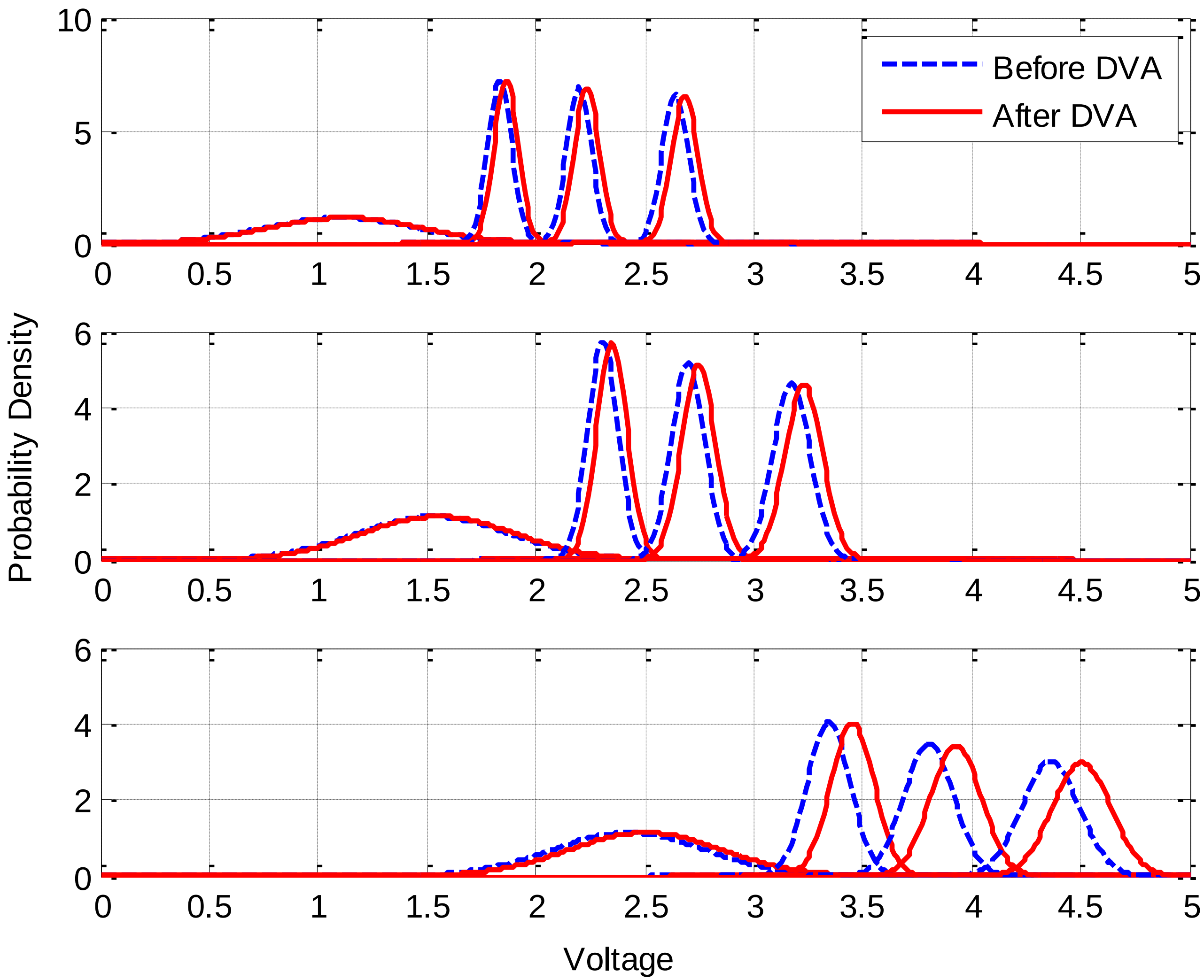
UCLA

Dynamic Voltage Allocation Scaling Factor Example



UCLA

Dynamic Voltage Allocation Scaling Example



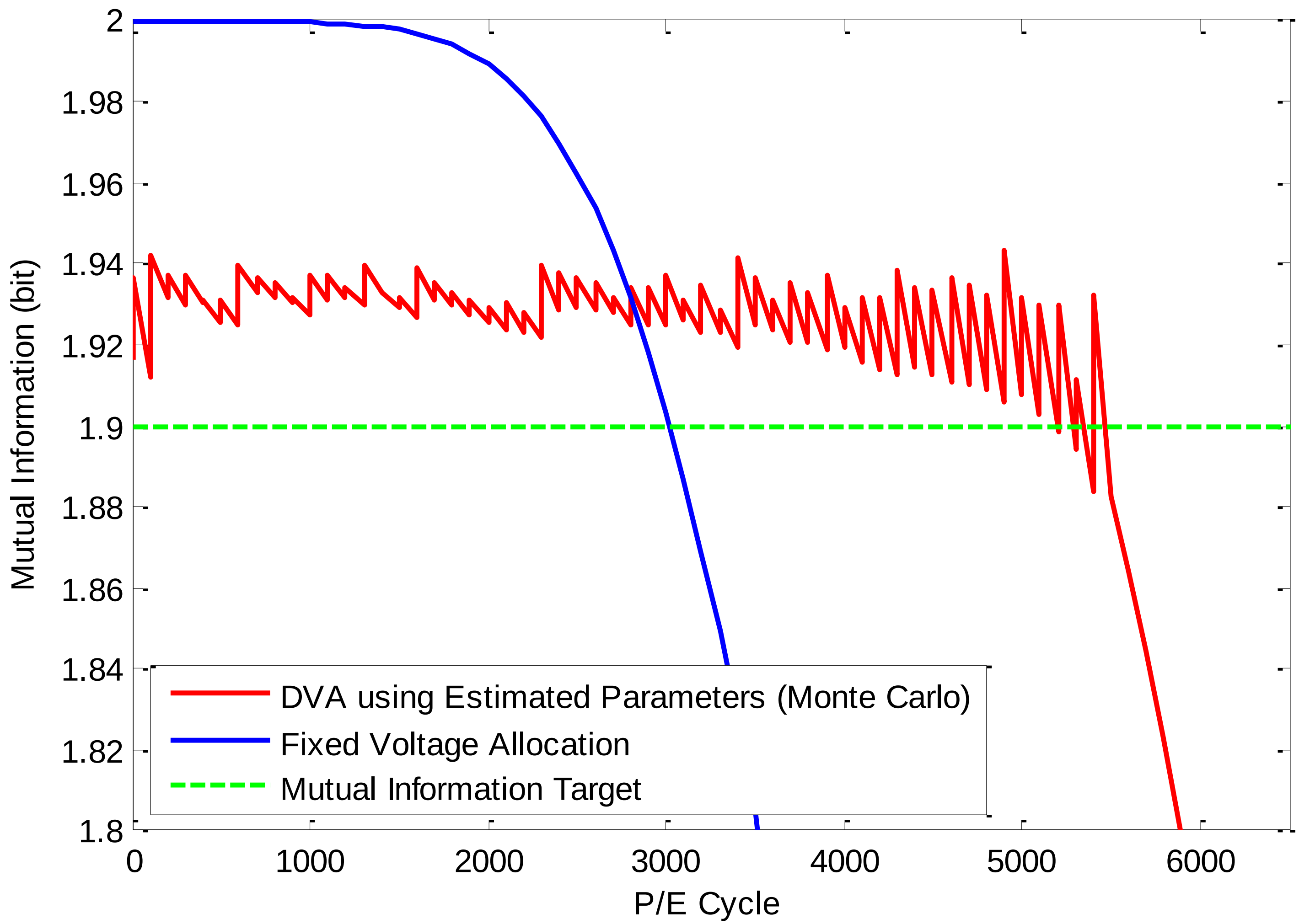
P/E = 1000
1.930 bits --> 1.935 bits

P/E = 3000
1.926 bits --> 1.935 bits

P/E = 5000
1.905 bits --> 1.935 bits

UCLA

Monte Carlo Simulation Result for MLC Flash



UCLA

Conclusion

- Levenberg-Marquardt algorithm can provide accurate channel parameter estimations using limited resolution histograms.

- 10-bin equal-probability binning strategy is a good choice for Flash channel estimation using least squares algorithms.

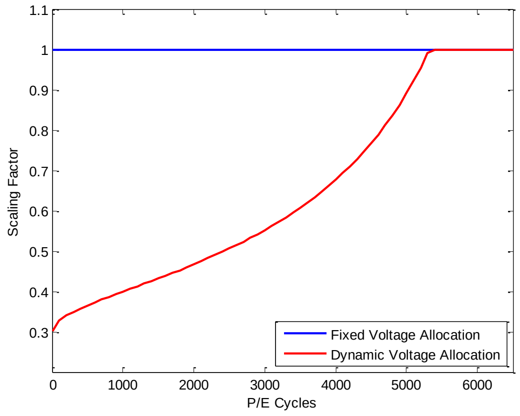
- Dynamic voltage allocation with histogram-based Flash channel estimation can extend lifetime significantly.

Challenges

- Flash can only place write voltage at certain positions.
  - Have results showing the impact is limited.

Challenges

- Flash can only place write voltage at certain positions.
  – Have results showing the impact is limited.
- Impractical to estimate the channel on the fly. & Parameter difference between chips.
  – Estimate offline. We only need a scaling curve guaranteeing the worst case error rate.
  – Machine learning.



Thank you!

Characterize the Peeling Process

To calculate $p_l(x)$, we first need to calculate $p_l(x|i,\omega)$, where i is the initial left degree of a randomly selected edge from B, and ω is the initial erasure state of its incident left node. If $\omega > i-1$, $p_l(x|i,\omega) = 1$ because even if all the neighboring edges of the selected edge provide increments, the VL decoder corresponding to the incident left cannot decode. If $\omega = i-1$, $p_l(x|i,\omega) = 1-(1-x)^{\omega}$, where $1-x$ is the probability that an edge is in the set $\{Q_1,\cdots,Q_t\}$ and the corresponding VL decoder requires all ω increments to successfully decode. If $\omega < i-1$,

$$p_l(x|i,\omega) = \sum_{j=0}^{\omega-1} \binom{i-1}{j} (1-x)^j x^{i-1-j} \ . \qquad (1)$$

Combine the three scenarios,

$$p_l(x|i,\omega) = \sum_{j=0}^{\min(\omega,i)-1} \binom{i-1}{j} (1-x)^j x^{i-1-j} \ , \qquad (2)$$

and when $\omega = 0$, $p_l(x|i,\omega) = 0$.

Summing over all possible combinations of initial left degree i and initial erasure state ω regarding an edge in B,

$$p_l(x) = \sum_{\omega=0}^{m} \delta_\omega \sum_{i=1}^{d_L} \lambda_i p_l(x|i,\omega) \qquad (3)$$

UCLA

Characterize the Peeling Process

For a specified edge, define the right neighboring edges of an edge as the *other* edges connected to its incident right node. An edge can be right-degree-one only when all of its right neighboring edges in the original graph B have been removed because they are incident to a left node corresponding to a VL decoder that has already successfully decoded. For each such right neighboring edge, the probability that the left node corresponds to a VL decoder that has already successfully decoded is $1-p_l(x)$. Thus the probability that all $i-1$ right neighboring edges have left nodes corresponding to a VL decoder that has already successfully decoded is $p_r(x|i) = (1-p_l(x))^{i-1}$. Summing over all possible initial right degrees, we have

$$p_r(x) = \sum_{i=1}^{d_R} \rho_i (1-p_l(x))^{i-1} = \rho((1-p_l(x)) \ . \qquad (7)$$

Which adjacent degrees to choose?

- For the inter-frame LDGM code,

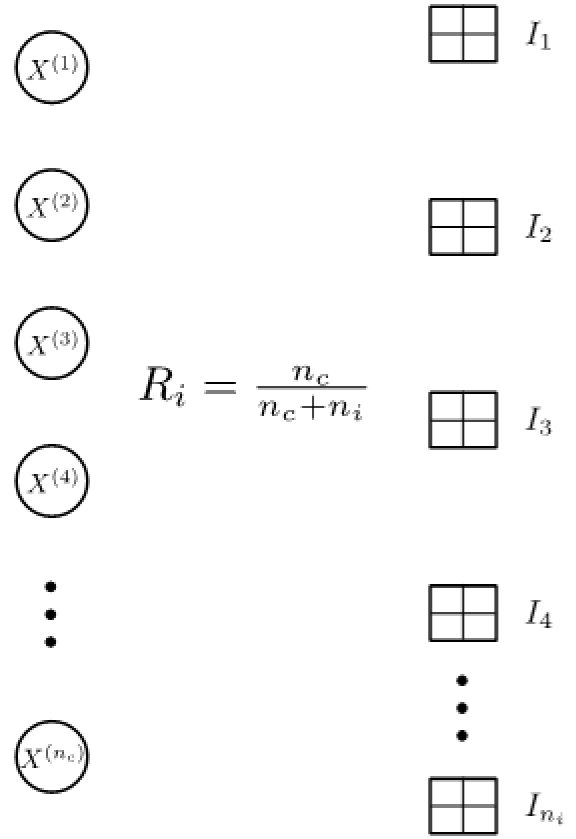
$$\beta_{FF} = \frac{n_i}{n_c} = \frac{1}{R_i} - 1$$

- For any $\lambda(x)$,$\rho(x)$,

$$\beta_{FF} = \frac{\int_0^1 \rho(x)\,dx}{\int_0^1 \lambda(x)\,dx} \geq \beta_{FB}$$

- When $\lambda(x) = x^3$,

$$4\int_0^1 \rho(x)\,dx \geq \beta_{FB}$$



Programming Noise (n_p)

- The uncertainty of the programmed threshold voltage immediately after program operation can be modeled by a Gaussian random variable.

- The variance of the programmed threshold voltage is larger when left in the erased state than when actively programmed.

$$f(n_p) = \begin{cases} N(0, \sigma_e^2) & \text{if } x = 0 \\ N(0, \sigma_p^2) & \text{if } x > 0 \end{cases} \quad \text{where } \sigma_e > \sigma_p$$

UCLA

Wear-out Noise (n_w)

- Wear-out induces threshold voltage shift as a result of traps generation and electron trapping/de-trapping during P/E cycling. The number of traps grows as the number of program/erase cycles increases.
- Trap behavior is modeled as random telegraph noise (RTN). This causes the distribution of measured thresholds features exponential tails.
- In some devices, the positive-shift tail is more significant than the negative-shift one, so we use an exponential distribution to model wear-out noise.

$$f(n_w) = \begin{cases} \frac{1}{\lambda} e^{-\frac{n_w}{\lambda}} & n_w \geq 0 \\ 0 & n_w < 0 \end{cases}$$

UCLA

Retention Noise (n_r)

- Retention loss is the reduction of programmed threshold voltage over time caused primarily by electron de-trapping.

- Retention noise is modeled as a Gaussian random variable where the mean and variance depend on the retention time and number of traps.

$$f(n_r) = \frac{1}{\sigma_r\sqrt{2\pi}} e^{-\frac{(n_r-\mu_r)^2}{2\sigma_r^2}}$$

Parameter Degradation Model

- Degradation Model
  - Wear-out noise:

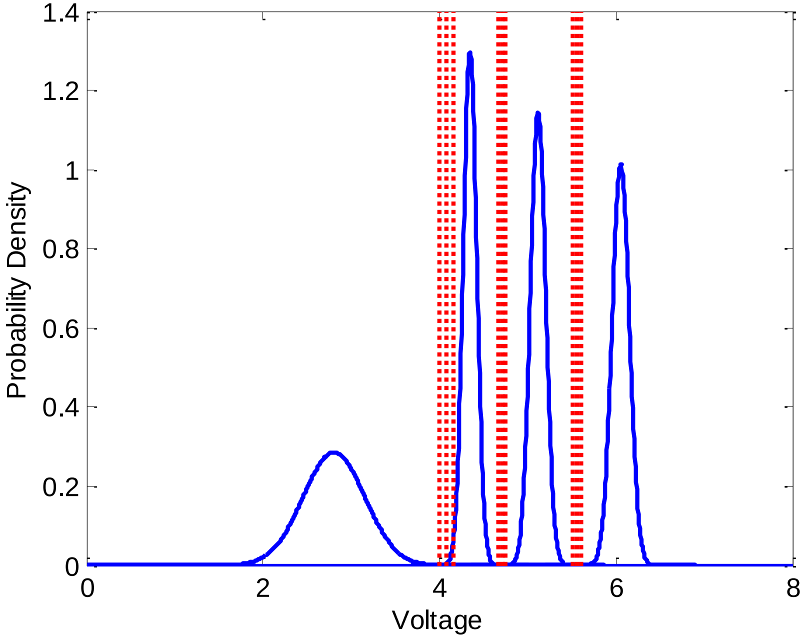
$$\lambda = C_w + A_w \cdot \left(\frac{V_{acc}}{V_{\max}} \right)^{0.62}$$

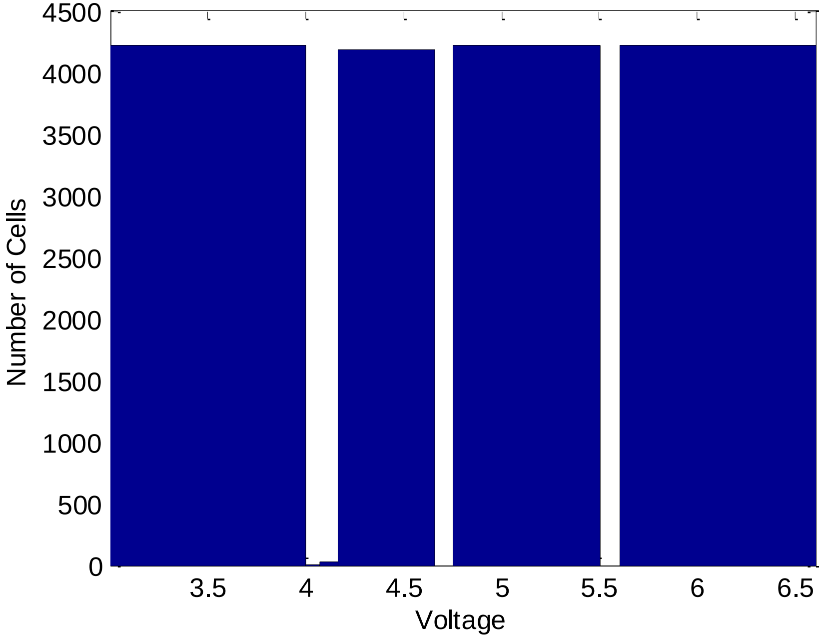
  - Retention noise:

$$\mu_r = -x \cdot \ln\left(1 + \frac{t}{t_0}\right) \cdot \left[A_r \cdot \left(\frac{V_{acc}}{V_{\max}} \right)^{0.62} + B_r \cdot \left(\frac{V_{acc}}{V_{\max}} \right)^{0.3} \right]$$

$$\sigma_r^2 = 0.1x \cdot \ln\left(1 + \frac{t}{t_0}\right) \cdot \left[A_r \cdot \left(\frac{V_{acc}}{V_{\max}} \right)^{0.62} + B_r \cdot \left(\frac{V_{acc}}{V_{\max}} \right)^{0.3} \right]^2$$

MMI Histogram only provides resolution at decision boundaries.





UCLA