Dynamic Threshold Schemes for Multi-Level Non-volatile Memories

Frederic Sala
http://fredsala.bol.ucla.edu/
LORIS Lab, UCLA

Joint work with:
Ryan Gabrys (UCLA),
Lara Dolecek (UCLA)

CoDESS Kickoff Meeting
Sept. 19th, 2013

About me

- 1st year Ph.D. student in UCLA LORIS Lab
- Research areas: Coding for storage, combinatorics
- Selected Papers:

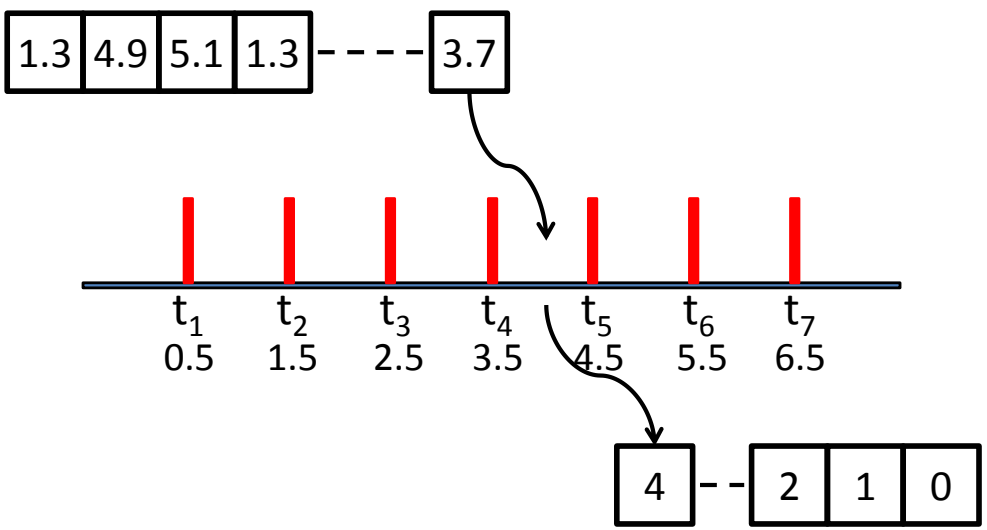
F. Sala, R. Gabrys and L. Dolecek, "Dynamic threshold schemes for multi-level non-volatile memories," *IEEE Trans. on Commun.*, 2013.

F. Sala and L. Dolecek, "Constrained Rank Modulation," in Proc. *IEEE Information Theory Workshop (ITW)*, 2013.

F. Sala and L. Dolecek, "Counting Sequences Obtained from the Synchronization Channel," in Proc. *IEEE International Symposium on Information Theory (ISIT)*, 2013.

Reading NVMs

- We wish to read a cell (in the MLC case) with q levels
- Traditionally, the voltage of a cell is measured, and this value is compared to a set of fixed thresholds
- For q levels, $q-1$ thresholds are required
- **Ex:** $q = 4$, $\mathbf{t} = (0.5, 1.5, 2.5)$: $1.54 \rightarrow 2$, $2.9 \rightarrow 3$, $0.34 \rightarrow 0$

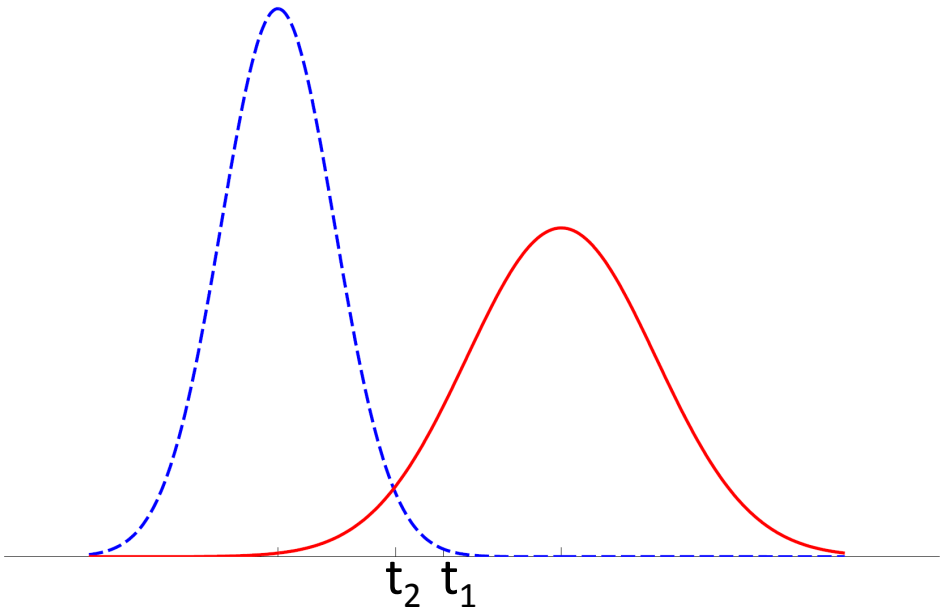


Downsides of fixed thresholds

- There are problems with this approach

Downsides of fixed thresholds

- There are problems with this approach
- Distributions of cell voltages change over time. Fixed thresholds cannot react to these changes
- For example, charge leakage causes distributions to skew in one direction
- Overwriting can cause skew in the opposite direction



Existing Solutions

- Necessary to change thresholds over time: "dynamic thresholds" instead of fixed thresholds
- Existing industry solutions
- Ex: patents 7,876,621; 8,125,833 - Adaptive dynamic reading of flash memories / SanDisk:
  - Write known values, evenly distributed among $\{0, 1, \ldots, q-1\}$ to a subset of the cells in a block
  - Read the voltages of these cells and build a histogram
  - From the histogram, determine the statistical parameters and set thresholds accordingly

Our approach

- We rely on a different dynamic thresholding approach.
- This solution was introduced (for the SLC case) by Zhou, Jiang, and Bruck in

H. Zhou, A. Jiang, and J. Bruck, "Error-correcting schemes with dynamic thresholds in non-volatile memories," *IEEE ISIT*, St. Petersburg, Jul.-Aug. 2011.

Our approach

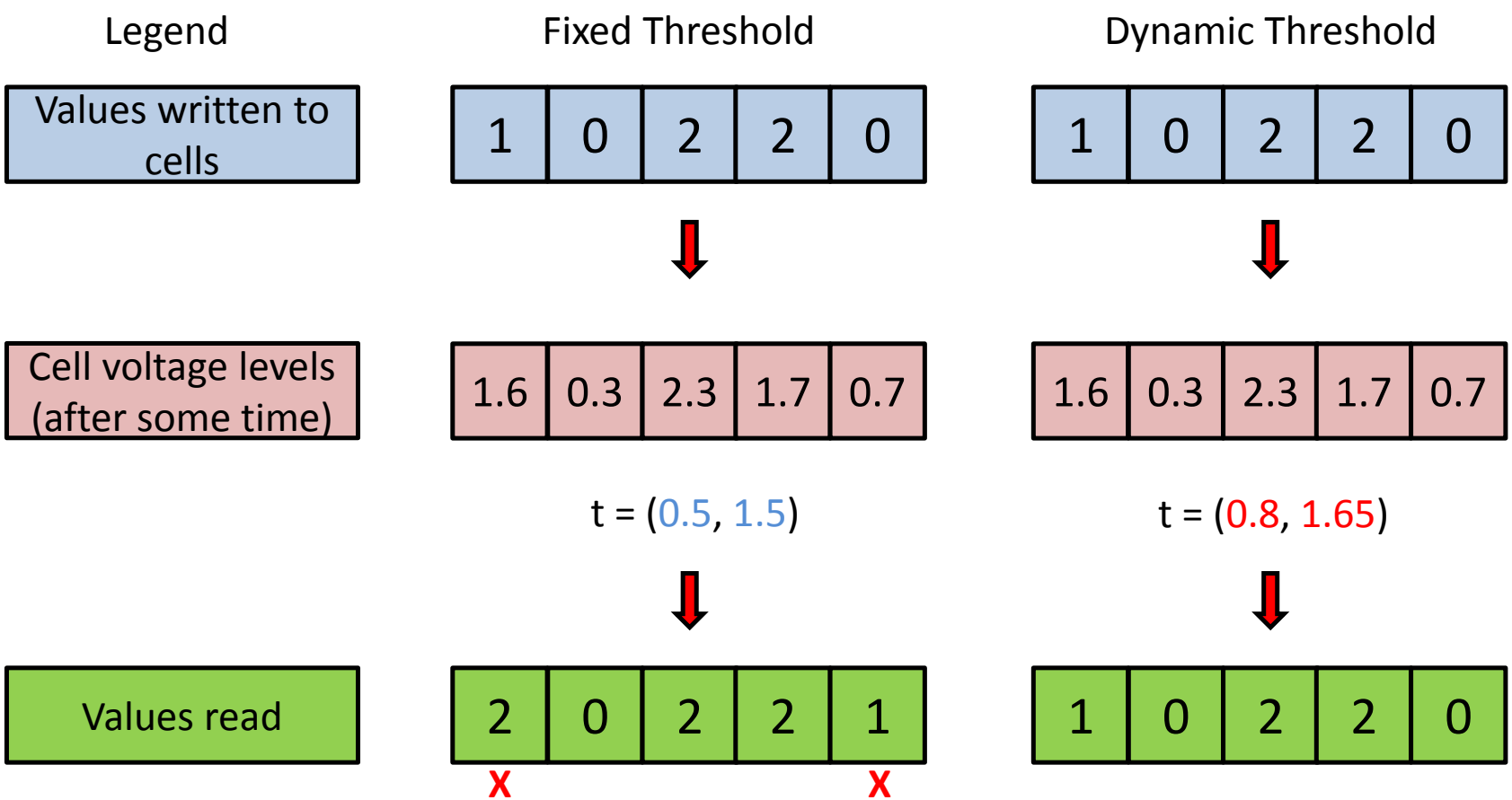
- We rely on a different dynamic thresholding approach.
- This solution was introduced (for the SLC case) by Zhou, Jiang, and Bruck in

  H. Zhou, A. Jiang, and J. Bruck, "Error-correcting schemes with dynamic thresholds in non-volatile memories," *IEEE ISIT*, St. Petersburg, Jul.-Aug. 2011.
- Basic idea: pick the thresholds so that the number of cells at each level (in an entire block) stays the same

Example

Here, the blocksize $n = 5$ and the number of cell levels $q = 3$:



Comparison with existing approach

What are the advantages of this approach?

- Writing to cells is not (necessarily) needed, so write disturbs are minimized, yielding better device lifetime
- Circuits necessary for parameter estimation are not needed; complexity is reduced
- Interesting theoretical properties

Dynamic Thresholding Analysis

Computing Dynamic Thresholds

How can we generate the thresholds?

Computing Dynamic Thresholds

How can we generate the thresholds?

- Decoder (i.e., cell reader) needs the # of cells at each level $k_0, k_1, \ldots, k_{q-1}$. Two ways to accomplish this:
  1. Communicate these levels as metadata. $q-1$ numbers $\rightarrow$ cost $(q-1)\lceil \log_q(n+1) \rceil$

Computing Dynamic Thresholds

How can we generate the thresholds?

- Decoder (i.e., cell reader) needs the # of cells at each level $k_0, k_1, \ldots, k_{q-1}$. Two ways to accomplish this:
  1. Communicate these levels as metadata. $q-1$ numbers $\rightarrow$ cost $(q-1)\lceil \log_q(n+1) \rceil$
  2. Fix the number of cells at each level in a block in advance: a constant-weight code. # of levels as close as possible: non-binary balanced code

Computing Dynamic Thresholds

How can we generate the thresholds?

- Given:
  - Received sequence (voltage levels in a block)
  - # of cells at each level $k_0, k_1, \ldots, k_{q-1}$

Computing Dynamic Thresholds

How can we generate the thresholds?

- Given:
  - Received sequence (voltage levels in a block)
  - # of cells at each level $k_0, k_1, \ldots, k_{q-1}$
- Need: k_0th, (k_0+1)st smallest voltage, (k_0+k_1)st, (k_0+k_1+1)st smallest, ..., $(\sum_{i=0}^{j} k_i)$st, $(\sum_{i=0}^{j} k_i + 1)$st smallest values
- Solution: Sort sequence in ascending order. Place thresholds between the positions indicated above

Computing Dynamic Thresholds

How can we generate the thresholds?

- Given:
  - Received sequence (voltage levels in a block)
  - # of cells at each level $k_0, k_1, \ldots, k_{q-1}$
- Need: k_0th, (k_0+1)st smallest voltage, (k_0+k_1)st, (k_0+k_1+1)st smallest, ..., $(\sum_{i=0}^{j} k_i)$st, $(\sum_{i=0}^{j} k_i + 1)$st smallest values
- Solution: Sort sequence in ascending order. Place thresholds between the positions indicated above
- Example: $\mathbf{k} = (1, 4, 2, 3)$. Place thresholds in sequence:

$$(0.4, |0.8, 1.13, 1.2, 1.3, |1.48, 2.12, |2.14, 3.2, 3.8)$$

Practical Concerns

What if we don't have access to discrete cell level values?

Practical Concerns

What if we don't have access to discrete cell level values?

- We can use an algorithm similar to binary search:

Practical Concerns

What if we don't have access to discrete cell level values?

- We can use an algorithm similar to binary search:
  - Guess some initial threshold $\mathbf{t}^{(0)} = (t_1^{(0)}, t_2^{(0)}, \ldots, t_{q-1}^{(0)})$
  - Read block with $\mathbf{t}^{(0)}$. Results in cell level distribution $\mathbf{k}(\mathbf{t}^{(0)})$

Practical Concerns

What if we don't have access to discrete cell level values?

- We can use an algorithm similar to binary search:
  - Guess some initial threshold $\mathbf{t}^{(0)} = (t_1^{(0)}, t_2^{(0)}, \ldots, t_{q-1}^{(0)})$
  - Read block with $\mathbf{t}^{(0)}$. Results in cell level distribution $\mathbf{k}(\mathbf{t}^{(0)})$
  - Compare $\mathbf{k}(\mathbf{t}^{(0)})$ with the desired $\mathbf{k}$.

Practical Concerns

What if we don't have access to discrete cell level values?

- We can use an algorithm similar to binary search:
  - Guess some initial threshold $\mathbf{t}^{(0)} = (t_1^{(0)}, t_2^{(0)}, \ldots, t_{q-1}^{(0)})$
  - Read block with $\mathbf{t}^{(0)}$. Results in cell level distribution $\mathbf{k}(\mathbf{t}^{(0)})$
  - Compare $\mathbf{k}(\mathbf{t}^{(0)})$ with the desired $\mathbf{k}$.
  - Appropriately adjust $\mathbf{t}^{(0)}$ to yield $\mathbf{t}^{(1)}$ and repeat

Practical Concerns

Example of algorithm:

Practical Concerns

Example of algorithm:

- Say that $q = 2$ (binary case) and $n = 100$
- We are given half of the 100 cells should be 0 and half 1

Practical Concerns

Example of algorithm:

- Say that $q = 2$ (binary case) and $n = 100$
- We are given half of the 100 cells should be 0 and half 1
- Read block with $t^{(0)}$. We read 75 1s and 25 0s.

Practical Concerns

Example of algorithm:

- Say that $q = 2$ (binary case) and $n = 100$
- We are given half of the 100 cells should be 0 and half 1
- Read block with $t^{(0)}$. We read 75 1s and 25 0s.
- Increase $t^{(0)}$ to form $t^{(1)}$ and read with this new threshold

Practical Concerns

Example of algorithm:

- Say that $q = 2$ (binary case) and $n = 100$
- We are given half of the 100 cells should be 0 and half 1
- Read block with $t^{(0)}$. We read 75 1s and 25 0s.
- Increase $t^{(0)}$ to form $t^{(1)}$ and read with this new threshold
- If we have overshot, and there are now too many 0s, we may take

$$t^{(2)} = \frac{t^{(0)} + t^{(1)}}{2}$$

Practical Concerns

Another concern - when should we generate a set of dynamic thresholds?

Practical Concerns

Another concern - when should we generate a set of dynamic thresholds?

- Every time we read a single cell is too frequent
- Instead, only generate dynamic thresholds at specific times, and use them as fixed thresholds until next “refresh”

Practical Concerns

Another concern - when should we generate a set of dynamic thresholds?

- Every time we read a single cell is too frequent
- Instead, only generate dynamic thresholds at specific times, and use them as fixed thresholds until next "refresh"
- We can trigger a refresh:

Practical Concerns

Another concern - when should we generate a set of dynamic thresholds?

- Every time we read a single cell is too frequent
- Instead, only generate dynamic thresholds at specific times, and use them as fixed thresholds until next "refresh"
- We can trigger a refresh:
  - After a fixed timeout
  - When an entire block of cells is being read
  - After sufficiently many cells have been written to

Error Analysis

When do errors occur when using D.T.?

Error Analysis

When do errors occur when using D.T.?

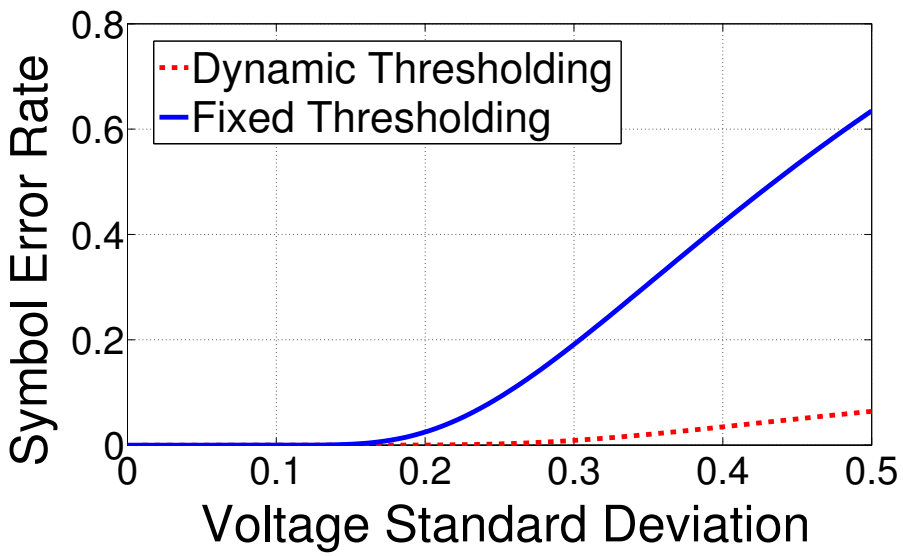
- By definition, *values* in a block cannot change
- But, the *order* of values may change
- Example: $(2, 1, 3)$ is written. Voltages read are $(2.4, 1.9, 1.8)$. Then, $(3, 2, 1)$ is the decoded word: a permutation of $(2, 1, 3)$

Error Analysis

- Note that a single error is not possible
- However, a large deviation in one voltage can cause 2 errors

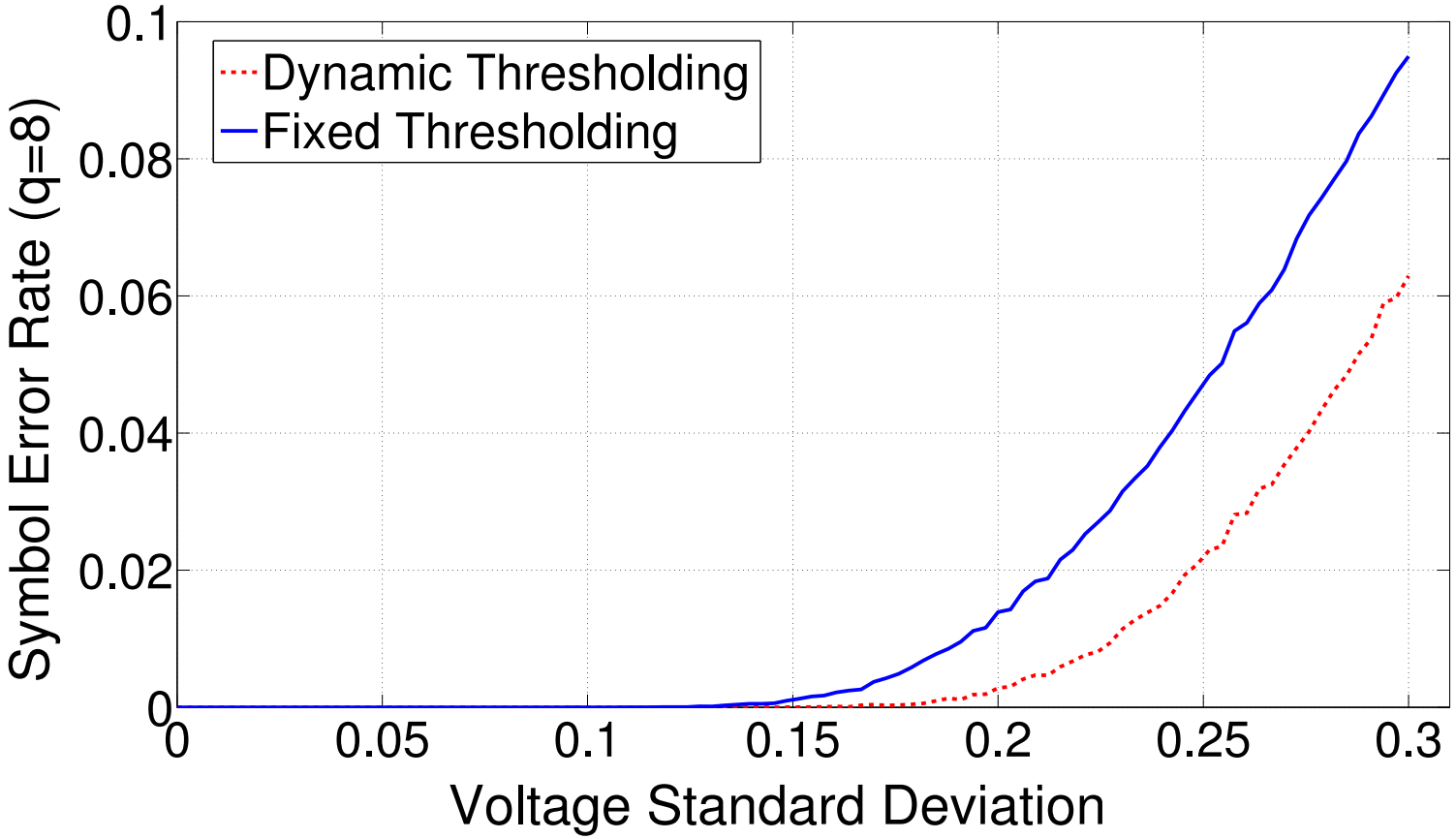
Error Analysis

- Note that a single error is not possible
- However, a large deviation in one voltage can cause 2 errors
- In general, errors take place when cells a, b with values $x < y$ have voltages such that $v_a > v_b$
- Fortunately, much more rare than fixed threshold errors:



Simulation Results

TLC cells ($q = 8$), $n = 10^5$, voltages modeled as Gaussians with increasing std. dev.



Theoretical Analysis

We have a specific scheme for selecting thresholds

- How does it compare to the "optimal" threshold?

Theoretical Analysis

We have a specific scheme for selecting thresholds

- How does it compare to the "optimal" threshold?
  - Optimal threshold minimizes the number of errors N_e when reading a sequence
  - Cannot be determined without knowledge of the original sequence, so it is not "practical"
  - $\mathbf{t}^* \rightarrow$ optimal threshold, $\mathbf{t}^d \rightarrow$ dynamic threshold

Theoretical Analysis

- For this analysis, we set ℓ as the maximum deviation
  - Example: $\ell = 2$. $2 \rightarrow 4$ is a possible error. $2 \rightarrow 5$ is not
  - This limitation is reasonable: $2 \rightarrow 5$ error requires cell with value 2's voltage to exceed that of all cells with values 3,4
- "Performance bound:" How much worse do we do using $\mathbf{t}^d$ instead of $\mathbf{t}^*$?

Theoretical Analysis

- For this analysis, we set ℓ as the maximum deviation
  - Example: $\ell = 2$. $2 \rightarrow 4$ is a possible error. $2 \rightarrow 5$ is not
  - This limitation is reasonable: $2 \rightarrow 5$ error requires cell with value 2's voltage to exceed that of all cells with values 3,4
- "Performance bound:" How much worse do we do using $\mathbf{t}^d$ instead of $\mathbf{t}^*$?
- Result for $\ell \geq 1$:

$$N_e(\mathbf{t}^d) \leq (\ell + 1) N_e(\mathbf{t}^*)$$

- So: our scheme is very close to the theoretical (and unobtainable) optimal scheme

Error-correction codes

If we wish to guarantee the correction of specified types of common errors?

Error-correction codes

If we wish to guarantee the correction of specified types of common errors?

- Need an ECC. Preferably one suited to the effects of D.T.
- Several strategies. Select between
  1. Using the “metadata” implementation
  2. Using the “balanced code” implementation

Error-correction codes

If we wish to guarantee the correction of specified types of common errors?

- Need an ECC. Preferably one suited to the effects of D.T.
- Several strategies. Select between
  1. Using the "metadata" implementation
  2. Using the "balanced code" implementation

  and
  1. Correcting a fixed number of errors (up to t errors limited to magnitude ℓ)
  2. Correcting any number of errors (of magnitude limited to ℓ)

Error-correction codes

- We refer to a code that corrects t errors limited to magnitude ℓ under D.T. as a (t, ℓ)-DTC code

Error-correction codes

- We refer to a code that corrects t errors limited to magnitude ℓ under D.T. as a (t, ℓ)-DTC code
- Start with the "metadata" approach, and assume that the decoder knows the number of cells at each level
- Received sequence $\mathbf{y}$ is a permutation of the input $\mathbf{x}$ with at least $n - t$ fixed points
- If $a \to b$ is an error, $|b - a| \leq \ell$

(t, ℓ)-DTC codes

Error vector $\mathbf{e} = \mathbf{y} - \mathbf{x}$ has certain interesting properties

- It is $(t-1)$-decomposable (using Roth's concept of "decomposability" for location-correction codes)

(t, ℓ)-DTC codes

Error vector $\mathbf{e} = \mathbf{y} - \mathbf{x}$ has certain interesting properties

- It is $(t-1)$-decomposable (using Roth's concept of "decomposability" for location-correction codes)
- Difference of error vectors $\mathbf{e_2} - \mathbf{e_1}$ is $2t-2$-decomposable
- If a code has minimum decomposability distance $2t-1$, it is a (t, ℓ)-DTC code

(t, ℓ)-DTC codes

Error vector $\mathbf{e} = \mathbf{y} - \mathbf{x}$ has certain interesting properties

- It is $(t-1)$-decomposable (using Roth's concept of "decomposability" for location-correction codes)
- Difference of error vectors $\mathbf{e_2} - \mathbf{e_1}$ is $2t-2$-decomposable
- If a code has minimum decomposability distance $2t-1$, it is a (t, ℓ)-DTC code
- Thus, location-correction codes are suitable for use with D.T.

(t, ℓ)-DTC codes

- *Construction 1*: Let C be a $[n, n-2]$ linear block code with parity-check matrix

$$H = \begin{bmatrix} a_1 & a_2 & \dots & a_n \\ a_1^2 & a_2^2 & \dots & a_n^2 \end{bmatrix},$$

  where $S = \{a_1, a_2, \dots, a_n\}$ is a set in $\{0, 1, \dots, q-1\}$
- Then, C is a $(2, q-1)$-DTC code if S is a Sidon set (all pairwise sums of elements are different: $a, b, c, d \in S \implies a + b \neq c + d$)

(t, ℓ)-DTC codes

- In Construction 1, we had $\ell = q - 1$, so essentially no magnitude limit
- Possible to derive constructions for any particular t, ℓ

(t, ℓ)-DTC codes

- In Construction 1, we had $\ell = q - 1$, so essentially no magnitude limit
- Possible to derive constructions for any particular t, ℓ
- Particularly simple construction if $\ell = 1$
- Downside - code length limited by size of Sidon sets

Balanced (t, ℓ)-DTC codes

What about the "balanced code" D.T. implementation?

Balanced (t, ℓ)-DTC codes

What about the "balanced code" D.T. implementation?

- Say $n = eq$. Then, the space of possible codewords B is the set of permutations of the multiset

$$\{\underbrace{0, 0, \ldots, 0}_{e \text{ 0s}}, \underbrace{1, 1, \ldots, 1}_{e \text{ 1s}}, \ldots, \underbrace{q-1, q-1, \ldots, q-1}_{e \ (q-1)\text{s}}\}$$

- Code on permutations ("rank modulation codes") can be generalized to codes on multi-set permutations to form balanced (t, ℓ)-DTC codes

ℓ-DTC codes

- We refer to a code that corrects any number of errors limited to magnitude ℓ under D.T. as a ℓ-DTC code
- *Construction 2*:

$$C = \{\mathbf{x} \in \mathrm{GF}_q^n \mid 0 < x_i - x_j \leq \ell \implies i > j\}$$

ℓ-DTC codes

- We refer to a code that corrects any number of errors limited to magnitude ℓ under D.T. as a ℓ-DTC code
- *Construction 2*:

$$C = \{\mathbf{x} \in \mathrm{GF}_q^n \mid 0 < x_i - x_j \leq \ell \implies i > j\}$$

  - Idea: Pick only one out of a candidate set of permutations
  - Select the permutation that is in "increasing" order
- Similar to ℓ-AEC (ℓ-asymmetric error correcting) codes studied by Bose and Elarief

ℓ-DTC codes

- Note: dual of the balanced (t, ℓ)-DTC code. In the balanced code approach, *all* codewords are permutations of one another. Here, *no* codewords are permutations

ℓ-DTC codes

- Note: dual of the balanced (t, ℓ)-DTC code. In the balanced code approach, *all* codewords are permutations of one another. Here, *no* codewords are permutations
- Duality due to the choice of channel: limited number of errors versus unlimited number of errors
- Decoding: Examine the received sequence $\mathbf{y}$ for y_i and y_j with $i < j$ and $0 < y_i - y_j \leq \ell$. Then, flip y_i and y_j
- Continue until no further pairs are found

Summary

- We showed how to apply dynamic threshold schemes to multi-level cell memories
- We analyzed the performance of our scheme and showed it performs well practically while being close to the theoretical "optimal" limit
- We explored the question of combining error-correction codes and dynamic thresholding

Future work

- Develop practical error-correcting code constructions for multi-permutations

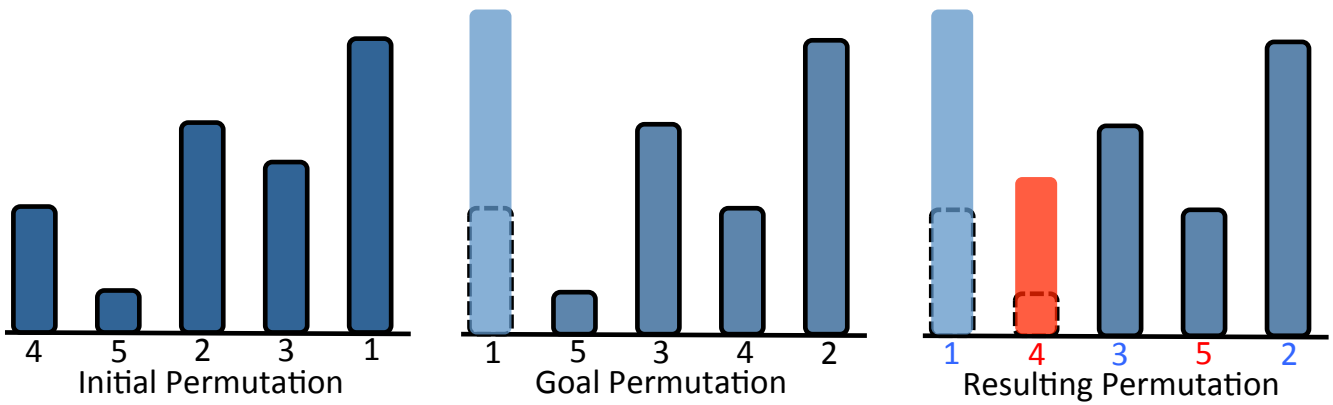
Thank you!

Constrained Permutations for Rank Modulation

- **Rank Modulation**: Represent information by cell rankings instead of absolute values. Ex:

$$\mathbf{x} = (2.3, 0.4, 1.2, 0.8, 2.0) \rightarrow (1\ 5\ 3\ 4\ 2)$$

- Resolves many NVM issues. Affected by **inter-cell coupling**: undesired rise in charge level due to large neighboring levels
- Solution: constrain differences between adjacent permutation elements. Constructions available. Asymptotically, capacity 1



Insertion and Deletion Sphere Bounds

- Insertions and deletions are particularly difficult types of errors:

$$\text{BABCBDCC} \rightarrow \text{AABBDC}$$

Insertion and Deletion Sphere Bounds

- Insertions and deletions are particularly difficult types of errors:

$$\text{BABCBDCC} \rightarrow \text{AABBDC}$$

- Useful to look at spheres formed by insertions and deletions

Insertion and Deletion Sphere Bounds

- Insertions and deletions are particularly difficult types of errors:

$$\text{BABCBDCC} \rightarrow \text{AABBDC}$$

- Useful to look at spheres formed by insertions and deletions
- Exact expression for # of sequences formed by 1 ins., 1 del ($\tau(X)$ is the # of runs of identical consecutive symbols in X)

$$|E_{1,1}(X)| \approx \tau(X)(n(q-1)-1)+2$$

- Combinatorially-derived bounds available for larger number of insertions and deletions

Insertion and Deletion Sphere Bounds

- Such bounds are useful when determining the rates of insertion and deletion error-correcting codes:

Insertion and Deletion Sphere Bounds

- Such bounds are useful when determining the rates of insertion and deletion error-correcting codes:
- Here, we use 2 insertions and 2 deletions

Insertion and Deletion Sphere Bounds

- Such bounds are useful when determining the rates of insertion and deletion error-correcting codes:
- Here, we use 2 insertions and 2 deletions

$$|E_{2,2}(X)| \geq \underbrace{\left(\sum_{i=0}^{2} \binom{\tau(X)-2}{i}\right)}_{\geq |D_2(X)|}$$

Insertion and Deletion Sphere Bounds

- Such bounds are useful when determining the rates of insertion and deletion error-correcting codes:
- Here, we use 2 insertions and 2 deletions

$$|E_{2,2}(X)| \geq \underbrace{\left(\sum_{i=0}^{2} \binom{\tau(X)-2}{i}\right)}_{\geq |D_2(X)|} \underbrace{\left(n(q-1) + \binom{n}{2}(q-1)^2\right)}_{|I_2(X')|}$$

Insertion and Deletion Sphere Bounds

- Such bounds are useful when determining the rates of insertion and deletion error-correcting codes:
- Here, we use 2 insertions and 2 deletions

$$|E_{2,2}(X)| \geq \underbrace{\left(\sum_{i=0}^{2}\binom{\tau(X)-2}{i}\right)}_{\geq |D_2(X)|}\underbrace{\left(n(q-1)+\binom{n}{2}(q-1)^2\right)}_{|I_2(X')|}$$
$$-\left(\binom{\tau(X)-1}{3}+3\binom{\tau(X)-2}{2}\right)$$
$$\underbrace{-\tau(X)\left(\binom{\tau(X)}{2}+\tau(X)-1\right)n(q-1)+1}_{\geq -\sum_{X_1,X_2\in D_2(X)}|I_2(X_1)\cap I_2(X_2)|}$$