LDPC Codes: Decoding

Andrew Thangaraj
andrew@iitm.ac.in

Decoder Modifications

- Offset and scaling in minsum
  - a (minsum) + b
  - Makes minsum as good as log-MAP
- Layered decoding
  - Faster convergence, fewer iterations
  - Codes in standards often support layering
- Most implementations
  - Offset, scaled minsum with layering

Layering in the LDPC matrix

- Rows of parity check matrix grouped into several layers
- Column weight of each layer = 1 (typically)

one block row in 5G LDPC codes

$$H = \begin{bmatrix} H_1 \\ \hline H_2 \end{bmatrix} = \left[\begin{array}{ccccccc} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ \hline 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{array}\right]$$

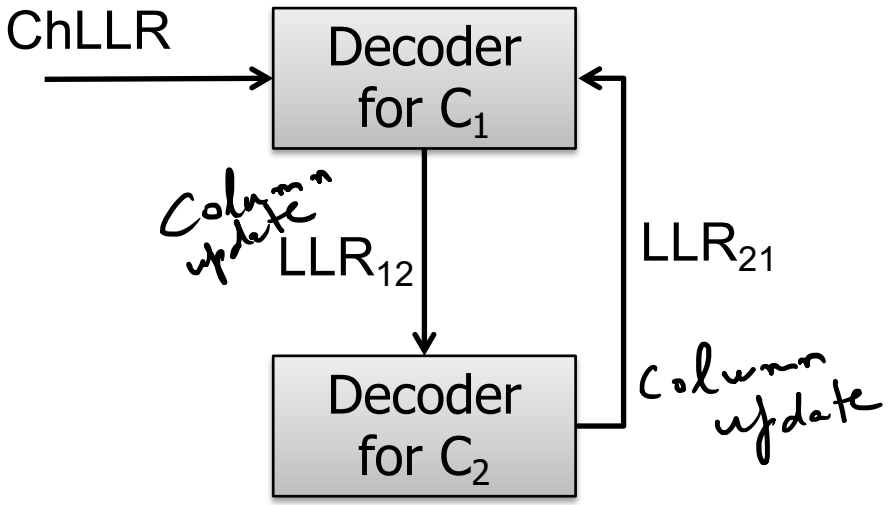
Layer 1

Layer 2

- C_1: Code with H_1 as parity check matrix
- C_2: Code with H_2 as parity check matrix
  - C_1, C_2: LDPC codes, Code = C_1 intersect C_2

Layered decoding

- Iterate between decoders for each layer



- Decoder for C_1 uses ChLLR only in first iteration
  - LLR_{21} is used afterwards
- Reduces required number of iterations by 1/2
  - Converges faster

Toy Example for Illustration

Initialization

$$H = \left[\begin{array}{ccccccc} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ \hline 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{array}\right]$$

layer 1 → col wt = 1

layer 2

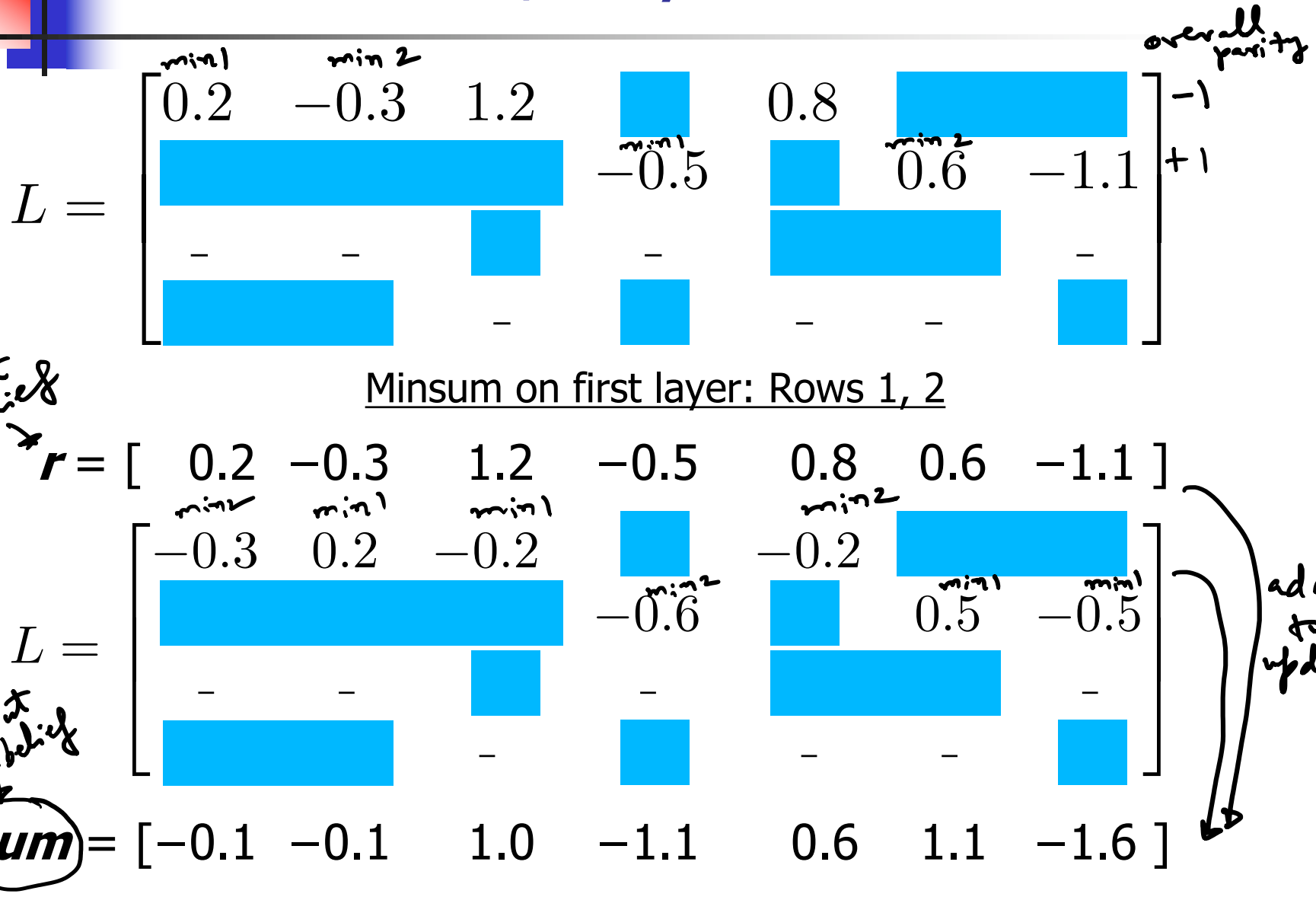
incoming received vector

$$\boldsymbol{r} = [\ 0.2 \quad -0.3 \quad 1.2 \quad -0.5 \quad 0.8 \quad 0.6 \quad -1.1\ ]$$

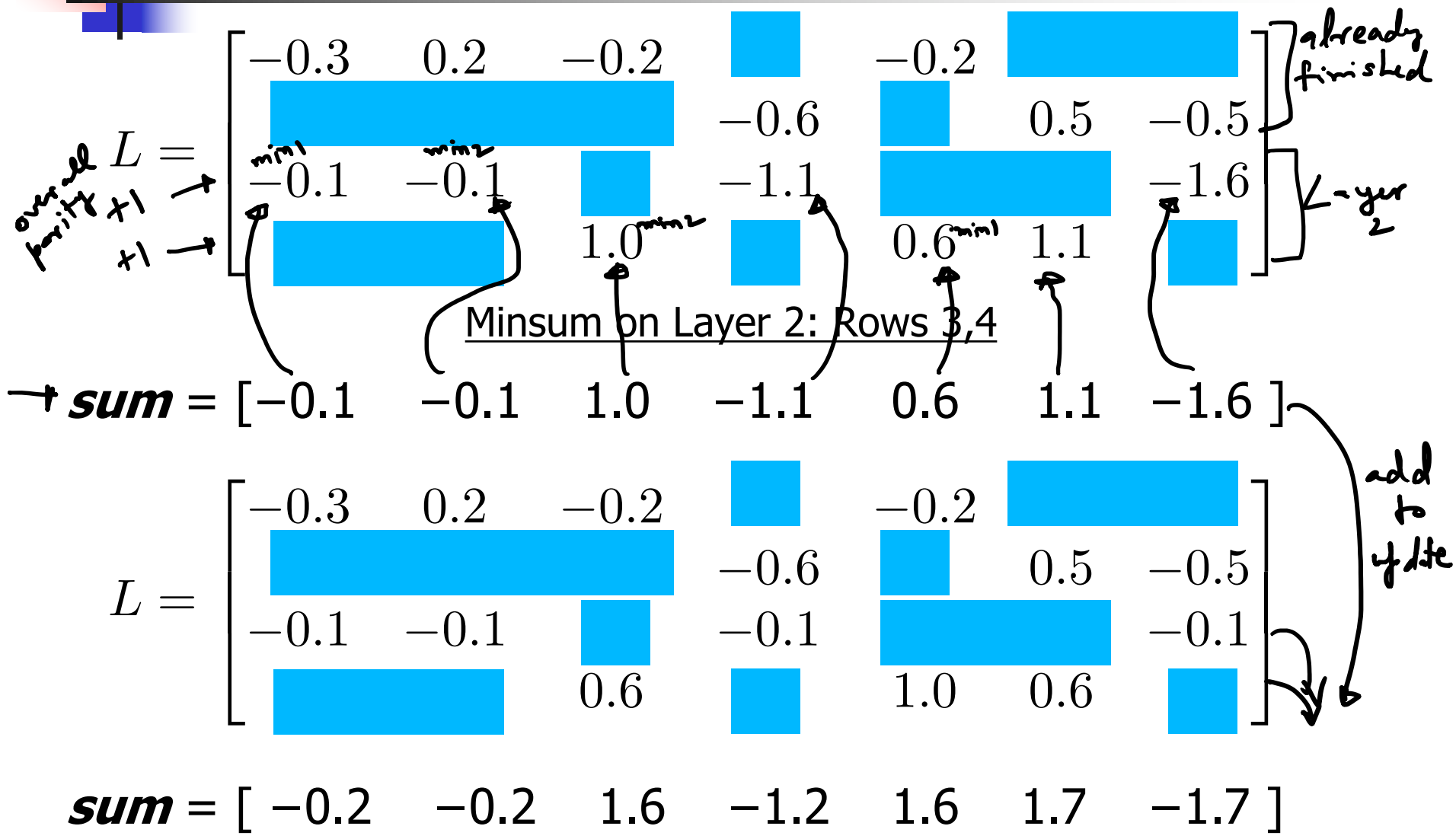
0's in PC matrix only by layer 1

$$L = \left[\begin{array}{ccccccc} 0.2 & -0.3 & 1.2 & & 0.8 & & \\ & & & -0.5 & & 0.6 & -1.1 \\ - & - & & - & & & - \\ & & - & & - & - & \end{array}\right]$$

Iteration 1, Layer 1



Iteration 1, Layer 2



Iteration 2, Layer 1: Subtract

sum = [−0.2 −0.2 1.6 −1.2 1.6 1.7 −1.7] incoming belief

$$L = \begin{bmatrix} -0.3 & 0.2 & -0.2 & & -0.2 & & \\ & & & -0.6 & & 0.5 & -0.5 \\ -0.1 & -0.1 & & -0.1 & & & -0.1 \\ & & 0.6 & & 1.0 & 0.6 & \end{bmatrix}$$

Layer 1

incoming belief − layer 1

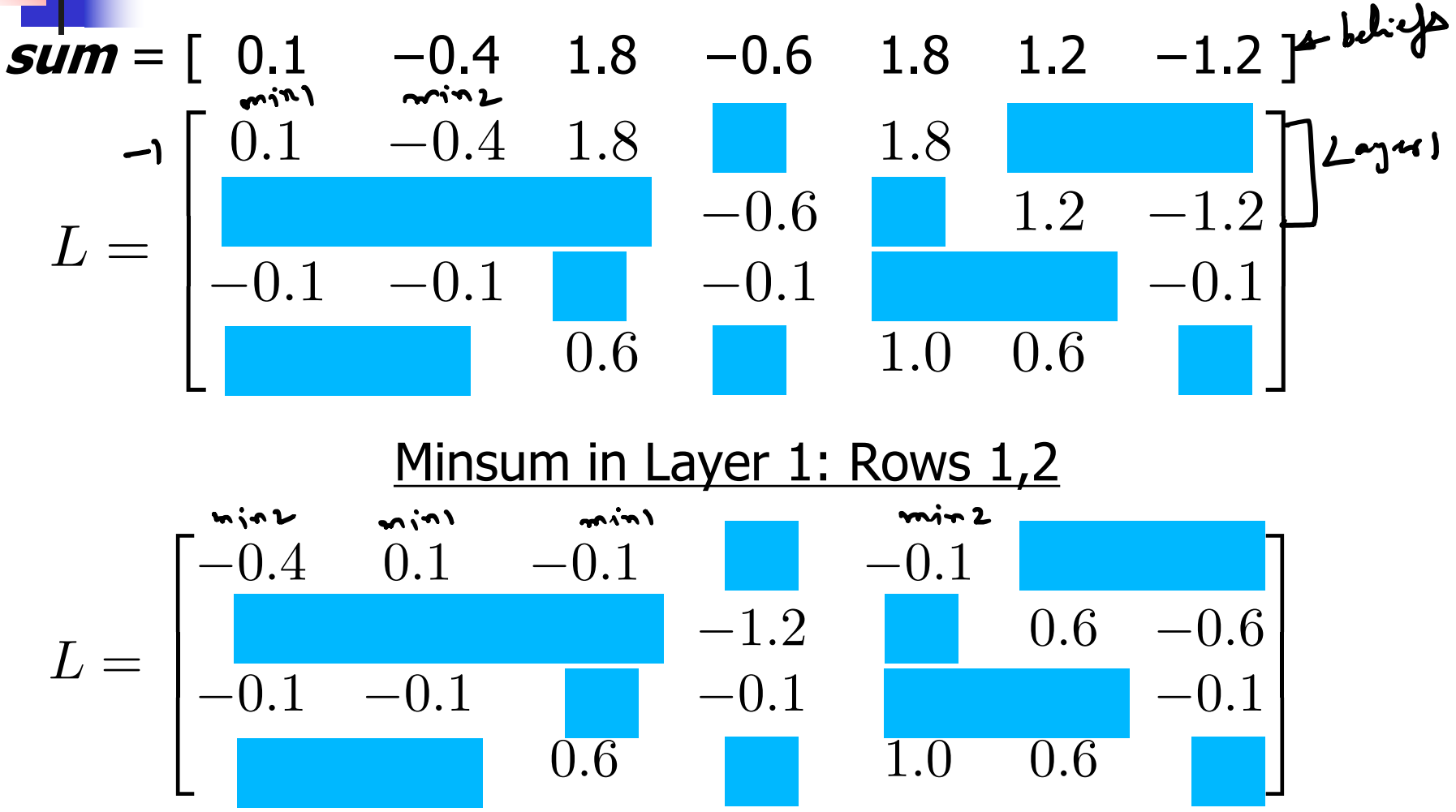
0.2 − (−0.3)

Subtraction in Layer 1: Rows 1,2

sum = [0.1 −0.4 1.8 −0.6 1.8 1.2 −1.2] ← corrected input for layer 1

$$L = \begin{bmatrix} 0.1 & -0.4 & 1.8 & & 1.8 & & \\ & & & -0.6 & & 1.2 & -1.2 \\ -0.1 & -0.1 & & -0.1 & & & -0.1 \\ & & 0.6 & & 1.0 & 0.6 & \end{bmatrix}$$

Iteration 2, Layer 1: Minsum



Iteration 2, Layer 1: Update

sum = [0.1 −0.4 1.8 −0.6 1.8 1.2 −1.2]

corrected version

Update in Layer 1: Rows 1,2

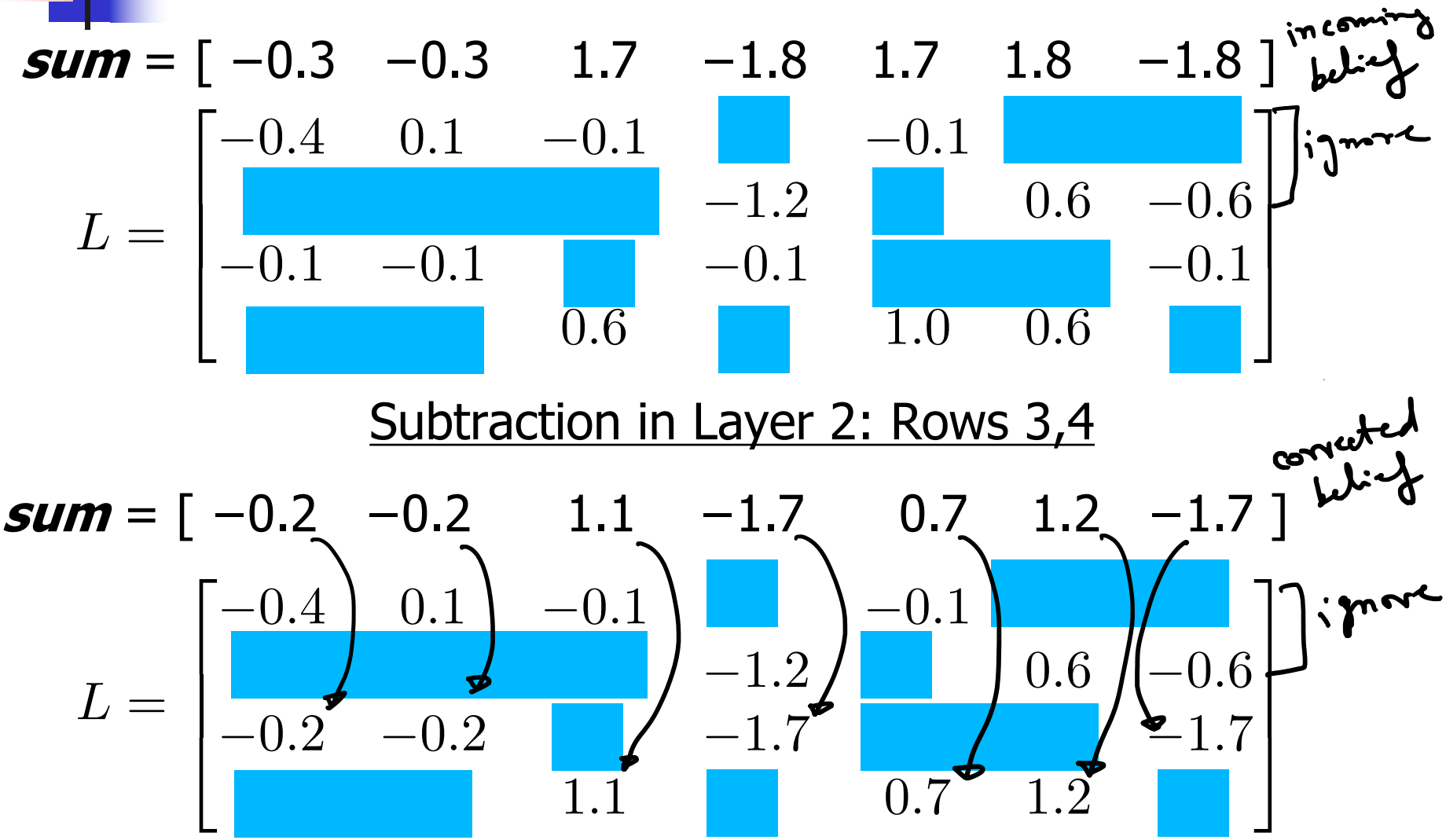
$$L = \begin{bmatrix} -0.4 & 0.1 & -0.1 & & -0.1 & \\ & & & -1.2 & & 0.6 & -0.6 \\ -0.1 & -0.1 & & -0.1 & & & -0.1 \\ & & 0.6 & & 1.0 & 0.6 & \end{bmatrix}$$

add to update

sum = [−0.3 −0.3 1.7 −1.8 1.7 1.8 −1.8]

Subtract → minsum → update

Iteration 2, Layer 2: Subtract



Iteration 2, Layer 2: Minsum

sum = [−0.2 −0.2 1.1 −1.7 0.7 1.2 −1.7]

$$L = \begin{bmatrix} -0.4 & 0.1 & -0.1 & & -0.1 & \\ & & & -1.2 & & 0.6 & -0.6 \\ -0.2 & -0.2 & & -1.7 & & & -1.7 \\ & & 1.1 & & 0.7 & 1.2 & \end{bmatrix}$$

Minsum in Layer 2: Rows 3,4

$$L = \begin{bmatrix} -0.4 & 0.1 & -0.1 & & -0.1 & \\ & & & -1.2 & & 0.6 & -0.6 \\ -0.2 & -0.2 & & -0.2 & & & -0.2 \\ & & 0.7 & & 1.1 & 0.7 & \end{bmatrix}$$

Iteration 2, Layer 2: Update

sum = [−0.2 −0.2 1.1 −1.7 0.7 1.2 −1.7]

Update in Layer 2: Rows 3,4

$$L = \begin{bmatrix} -0.4 & 0.1 & -0.1 & & -0.1 & \\ & & & -1.2 & & 0.6 & -0.6 \\ -0.2 & -0.2 & & -0.2 & & & -0.2 \\ & & 0.7 & & 1.1 & 0.7 & \end{bmatrix}$$

sum = [−0.4 −0.4 1.8 −1.9 1.8 1.9 −1.9]

Decision

- If $Sum_j > 0$, Decision on Bit j = 0
- If $Sum_j < 0$, Decision on Bit j = 1
  - Assuming BPSK 0 -> +1 and 1 -> -1

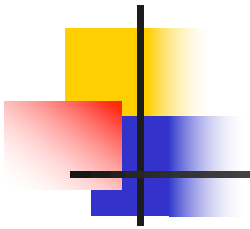
belief

After two iterations,

Sum = [−0.4 −0.4 1.8 −1.9 1.8 1.9 −1.9]

Dec = [1 1 0 1 0 0 1]

- For more iterations, continue…



Summary

- LDPC codes provide close-to-capacity performance
- Adopted in several standards
- VLSI architectures and implementations are available today
- Interesting issues
  - Optimized implementations
  - Speed and efficiency
  - Optimal layering?