A Guided Tour of CML, the Coded Modulation Library

Matthew Valenti and Terry Ferrett

Iterative Solutions

and West Virginia University

Morgantown, WV 26506-6109

mvalenti@wvu.edu

*Outline

1. CML overview
    - What is it? How to set it up and get started?

2. Uncoded Modulation
    - Simulate uncoded BPSK and QAM in AWGN and Rayleigh fading

3. Coded Modulation
    - Simulate a turbo code from UMTS 25.212
    - *Specify a custom eIRA LDPC parity check matrix

4. Ergodic (Shannon) capacity analysis
    - Determine the modulation constrained capacity of BPSK and QAM

5. EXIT analysis
    - Determine the EXIT characteristic for BPSK in AWGN for a particular LDPC degree distribution

Outline

6. Outage analysis
   - Determine the outage probability over block fading channels
   - Determine the outage probability of finite-length codes

7. Digital Network Coding
   - Describe CML implementation of digital network coding
   - Simulate error rate of digital network coding for non-channel-coded 4-FSK in Rayleigh fading

8. The internals of CML

9. Throughput calculation
   - Convert BLER to throughput for hybrid-ARQ

What is CML?

- CML is an open source toolbox for simulating capacity approaching codes in MATLAB.
- Available for free at the **Iterative Solutions** website:
  - www.iterativesolutions.com
- Runs in MATLAB, but uses c-mex for efficiency.
- First release was in Oct. 2005.
  - Used code that has been developed since 1996.

Features

- Simulation of BICM (bit interleaved coded modulation)
    - Turbo, LDPC, or convolutional codes.
    - User-specified eIRA LDPC parity check matrices.
    - PSK, QAM, FSK modulation.
    - BICM-ID: Iterative demodulation and decoding.
- Generation of ergodic capacity curves
    - BICM/CM constrained modulation.
- EXIT analysis
    - Generate LDPC EXIT characteristics for all CML detector types.
- Information outage probability
    - Block fading channels.
    - Blocklength-constrained channels (AWGN or fading)

Features

- Digital network coding
  - Error-rate of digital network coding in the two-way relay channel
  - Noncoherent FSK modulation
- Calculation of throughput of hybrid-ARQ.

Supported Standards

- Binary turbo codes:
  - UMTS/3GPP, including HSDPA and LTE.
  - cdma2000/3GPP2.
  - CCSDS.
- Duobinary turbo codes:
  - DVB-RCS.
  - WiMAX IEEE 802.16.
- LDPC codes:
  - DVB-S2.
  - Mobile WiMAX IEEE 802.16e.

Simulation Data is Valuable

- CML saves simulation state frequently
  - parameter called “save_rate” can be tuned to desired value.
- CML can be stopped at any time.
  - Intentionally: Hit CTRL-C within matlab.
  - Unintentionally: Power failure, reboot, etc.
- CML automatically resumes simulation
  - If a simulation is run again, it will pickup where it left off.
  - Can reset simulation by setting “reset=1”.
  - SNR points can be added or deleted prior to restarting.
- Simulations can be made more confident by requesting additional trials prior to restarting.
  - The new results will be added to the old ones.

Compiled Mode

- A flag called “compiled_mode” can be used to run CML independently of matlab.
- CML must first be compiled using the matlab compiler.
- Advantages:
  - Can run on machines without matlab.
  - Can run on a grid computer.

WebCML

- WebCML is a new initiative sponsored by NASA and NSF.
- Idea is to upload simulation parameters to a website and hit a “simulate” button.
  - Simulation begins on the webserver.
  - The webserver will divide the simulation into multiple jobs which are sent to a grid computer.
- Results can be retrieved while simulation is running and once it has completed.
- The grid is comprised of ordinary desktop computers.
  - The grid compute engine is a screen saver.
    - Kicks in only when computer is idle.
  - Users of WebCML are encouraged to donate their organizations computers to the grid.

Getting Started with CML

- Download
  - www.iterativesolutions.com/download.htm
  - Decompress downloaded archive
    - Root directory will be ./cml
- About simulation databases
  - A large database of previous simulation results is available.
  - Unzip each database and place each extracted directory into the ./cml/output directory
- About C-mex files.
  - C-mex files are compiled for PC computers.
  - For unix and mac computers, must compile.
    - Within matlab, cd to ./cml/source and type “make”.

Starting and Interacting with CML

- Launch MATLAB
- Cd to the <CMLROOT> directory
- Type “CmlStartup”
  - This sets up paths and determines the version of MATLAB.
- To run CML, only two functions are needed:
  - CmlSimulate
    - Runs one or more simulations.
    - Simulation parameters are stored in .m scripts.
    - Input arguments tell CML which simulation(s) to run.
  - CmlPlot
    - Plots the results of one or more simulations.

Scenario Files and the SimParam Structure

- The parameters associated with a set of simulations is stored in a scenario file.
  - Located in one of two directories
    - ./cml/scenarios for publicly available scenarios
    - ./cml/localscenarios for personal user scenarios
    - Other directories could be used if they are on the matlab path.
  - .m extension.
- Exercise
  - Edit the example scenario file: UncodedScenarios.m
- The main content of the scenario file is a structure called sim_param
  - Sim_param is an array.
  - Each element of the array is called a *record* and corresponds to a single distinct simulation.

Common Parameters

- List of all parameters can be found in:
  - ./cml/mat/DefineStructures.m
  - ./cml/documentation/readme.pdf
- Default values are in the DefineStructures.m file
- Some parameters can be changed between runs, others cannot.
  - sim_param_changeable
  - sim_param_unchangeable

Dissecting the SimParam Structure: The simulation type

- sim_param(record).sim_type =
  - 'uncoded'
    - BER and SER of uncoded modulation
  - 'coded'
    - BER and FER of coded modulation
  - 'capacity'
    - The Shannon capacity under modulation constraints.
  - 'exit'
    - EXIT characteristic of selected detector and parameterized LDPC code
  - 'outage'
    - The information outage probability of block fading channels
    - Assumes codewords are infinite in length
  - 'bloutage'
    - Information outage probability in AWGN or ergodic/block fading channels
    - Takes into account lenth of the code.

Dissecting the SimParam Structure: The simulation type

- sim_param(record).sim_type =
  - 'throughput'
    - By using FER curves, determines throughput of hybrid ARQ
    - This is an example of an *analysis* function … no simulation involved.

Lesser Used Simulation Types

- sim_param(record).sim_type =
  - ‘bwcapacity’
    - Shannon capacity of CPFSK under bandwidth constraints.
  - ‘minSNRvsB’
    - Capacity limit of CPFSK as a function of bandwidth

Parameters Common to All Simulations

- sim_param(record).
  - comment = {string}
    - Text, can be anything.
  - legend = {string}
    - What to put in figure caption
  - linetype = {string}
    - Color, type, and marker of line. Uses syntax from matlab "plot".
  - filename = {string}
    - Where to save the results of the simulation
    - Once filename is changed, any parameter can be changed.
  - reset = {0,1} with default of 0
    - Indication to resume "0" or restart "1" simulation when run again.
    - If reset = 1, any parameter may be changed.

Specifying the Simulation

- sim_param(record).
  - SNR = {vector}
    - Vector containing SNR points in dB
    - Can add or remove SNR points between runs
  - SNR_type = {'Eb/No in dB' or 'Es/No in dB'}
    - For some simulation types, only one option is supported.
    - E.g. for *capacity* simulations, it must be Es/No
  - save_rate = {scalar integer}
    - An integer specifying how often the state of the simulation is saved
    - Number of trials between saves.
    - Simulation echoes a period '.' every time it saves.

Specifying the Simulation (cont'd)

- sim_param(record).
  - max_trials = {vector}
    - A vector of integers, one for each SNR point
    - Tells simulation maximum number of trials to run per point.
  - max_frame_errors = {vector}
    - Also a vector of integers, one for each SNR point.
    - Tells simulation maximum number of frame errors to log per point.
    - Simulation echoes a ‘x’ every time it logs a frame error.
  - minBER = {scalar}
    - Simulation halts once this BER is reached
  - minFER = {scalar}
    - Simulation halts once this FER is reached.

Outline

1. CML overview
   - What is it? How to set it up and get started?

2. Uncoded Modulation
   - Simulate uncoded BPSK and QAM in AWGN and Rayleigh fading

3. Coded Modulation
   - Simulate a turbo code from UMTS 25.212
   - *Specify a custom eIRA LDPC parity check matrix

4. Ergodic (Shannon) capacity analysis
   - Determine the modulation constrained capacity of BPSK and QAM

5. EXIT analysis
   - Determine the EXIT characteristic for BPSK in AWGN for a particular LDPC degree distribution

Specifying Modulation

- sim_param(record).
  - modulation = {string}
    - Specifies the modulation type
    - May be ‘BPSK’, ‘QPSK’, ‘QAM’, ‘PSK’, ‘APSK’, ‘HEX’, or ‘FSK’
    - ‘HSDPA’ used to indicate QPSK and QAM used in HSDPA.
    - All but FSK are 2 dimensional modulations
      - Uses a complex scalar value for each symbol.
    - Default is ‘BPSK’
    - New (version 1.9 and above): Can also be set to “custom”.
  - mod_order = {integer scalar}
    - Number of points in the constellation.
    - Power of 2.
    - Default is 2.
    - In some cases, M=0 is used to indicate an unconstrained Gaussian input.
  - S_matrix = {complex vector}
    - Only used for “custom” modulation type.
    - A vector of length “mod_order” containing the values of the symbols in the signal set S.

Specifying Modulation

- sim_param(record).
  - mapping = {integer vector}
    - A vector of length M specifying how data bits are mapped to symbols.
    - Vector contains the integers 0 through M-1 exactly once.
    - ith element of vector is the set of bits associated with the ith symbol.
    - Alternatively, can be a string describing the modulation, like ‘gray’ or ‘sp’
    - Default is ‘gray’
  - framesize = {integer scalar}
    - The number of symbols per Monte Carlo trial
    - For coded systems, this is number of bits per codeword
  - demod_type = {integer scalar}
    - A flag indicating how to implement the demodulator

      0 = log-MAP (approximated linearly)

      1 = max-log-MAP

      2 = constant-log-MAP

      3 and 4 other implementations of log-MAP
    - Max-log-MAP is fastest.
    - Does not effect the uncoded error rate.
      - However, effects coded performance

M-ary Complex Modulation

- $\mu = \log_2 M$ bits are mapped to the symbol $\mathbf{x}_k$, which is chosen from the set $S = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M\}$
  - The symbol is multidimensional.
  - 2-D Examples: QPSK, M-PSK, QAM, APSK, HEX
    - These 2-D signals take on complex values.
  - M-D Example: FSK
    - FSK signals are represented by the M-dimensional complex vector **X**.
- The signal $y = hx_k + n$ is received
  - h is a complex fading coefficient (scalar valued).
  - n is complex-valued AWGN noise sample
  - More generally (FSK), $\mathbf{Y} = h\,\mathbf{X} + \mathbf{N}$
    - Flat-fading: All FSK tones multiplied by the same fading coefficient h.
- Modulation implementation in CML
  - The complex signal set S is created with the **CreateConstellation** function.
  - Modulation is performed using the **Modulate** function.

Log-likelihood of Received Symbols

- Let $p(\mathbf{x}_k|\mathbf{y})$ denote the probability that signal $\mathbf{x}_k \in S$ was transmitted given that **y** was received.
- Let $f(\mathbf{x}_k|\mathbf{y}) = \mathrm{K}\, p(\mathbf{x}_k|\mathbf{y})$, where K is any multiplicative term that is constant for all $\mathbf{x}_k$.
- When all symbols are equally likely, $f(\mathbf{x}_k|\mathbf{y}) \propto f(\mathbf{y}|\mathbf{x}_k)$
- For each signal in S, the receiver computes $f(\mathbf{y}|\mathbf{x}_k)$
  - This function depends on the modulation, channel, and receiver.
  - Implemented by the **Demod2D** and **DemodFSK** functions, which actually computes $\log f(\mathbf{y}|\mathbf{x}_k)$.
- Assuming that all symbols are equally likely, the most likely symbol $\mathbf{x}_k$ is found by making a hard decision on $f(\mathbf{y}|\mathbf{x}_k)$ or $\log f(\mathbf{y}|\mathbf{x}_k)$.

Example: QAM over AWGN.

- Let y = x + n, where n is complex i.i.d. $N(0,N_0/2\ )$ and the average energy per symbol is $E[|x|^2] = E_s$

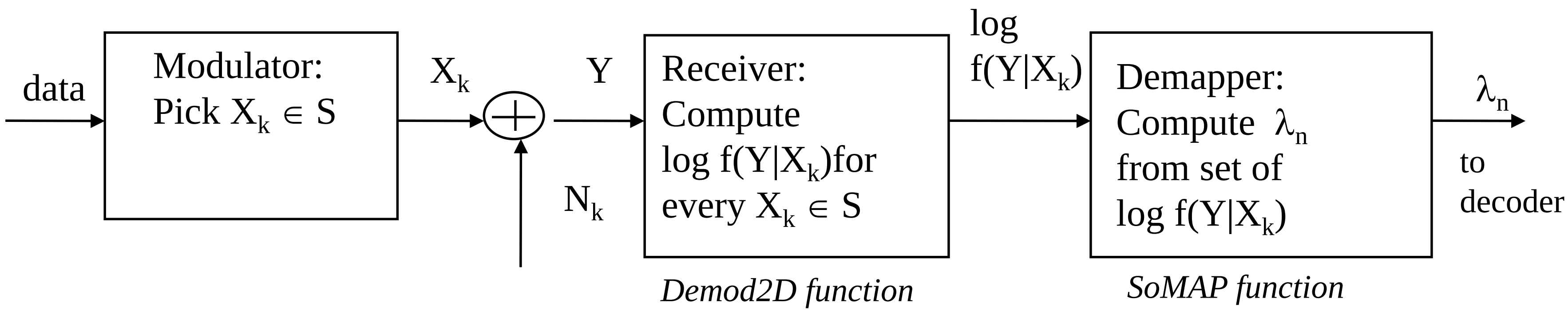
$$p(y \mid x_k) = \frac{1}{2\sigma^2} \exp\left\{ \frac{-\left| y - x_k \right|^2}{2\sigma^2} \right\}$$

$$f(y \mid x_k) = \exp\left\{ \frac{-\left| y - x_k \right|^2}{2\sigma^2} \right\}$$

$$\log f(y \mid x_k) = \frac{-\left| y - x_k \right|^2}{2\sigma^2}$$

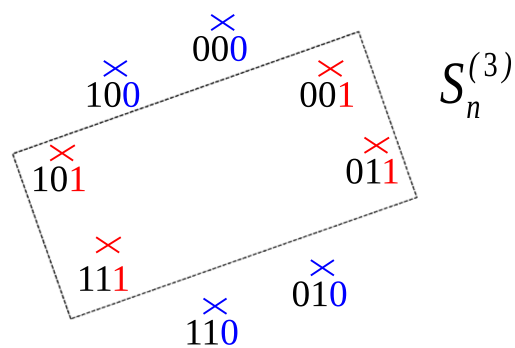
$$= \frac{-E_s \left| y - x_k \right|^2}{N_o}$$

Converting symbol liklihoods to bit LLR



- The symbol likelihoods must be transformed into bit log-likelihood ratios (LLRs):

$$\lambda_n = \log \frac{P[d_n = 1]}{P[d_n = 0]} = \log \frac{\sum_{X_k \in S_n^{(1)}} f(Y \mid X_k)}{\sum_{X_k \in S_n^{(0)}} f(Y \mid X_k)}$$



  - Where $S_n^{(1)}$ represents the set of symbols whose nth bit is a 1.
  - and $S_n^{(0)}$ is the set of symbols whose nth bit is a 0.

Log-domain Implementation

$$\lambda_n = \log \frac{\sum_{X_k \in S_n^{(1)}} f(Y \mid X_k)}{\sum_{X_k \in S_n^{(0)}} f(Y \mid X_k)}$$

$$= \log \sum_{X_k \in S_n^{(1)}} f(Y \mid X_k) - \log \sum_{X_k \in S_n^{(0)}} f(Y \mid X_k)$$

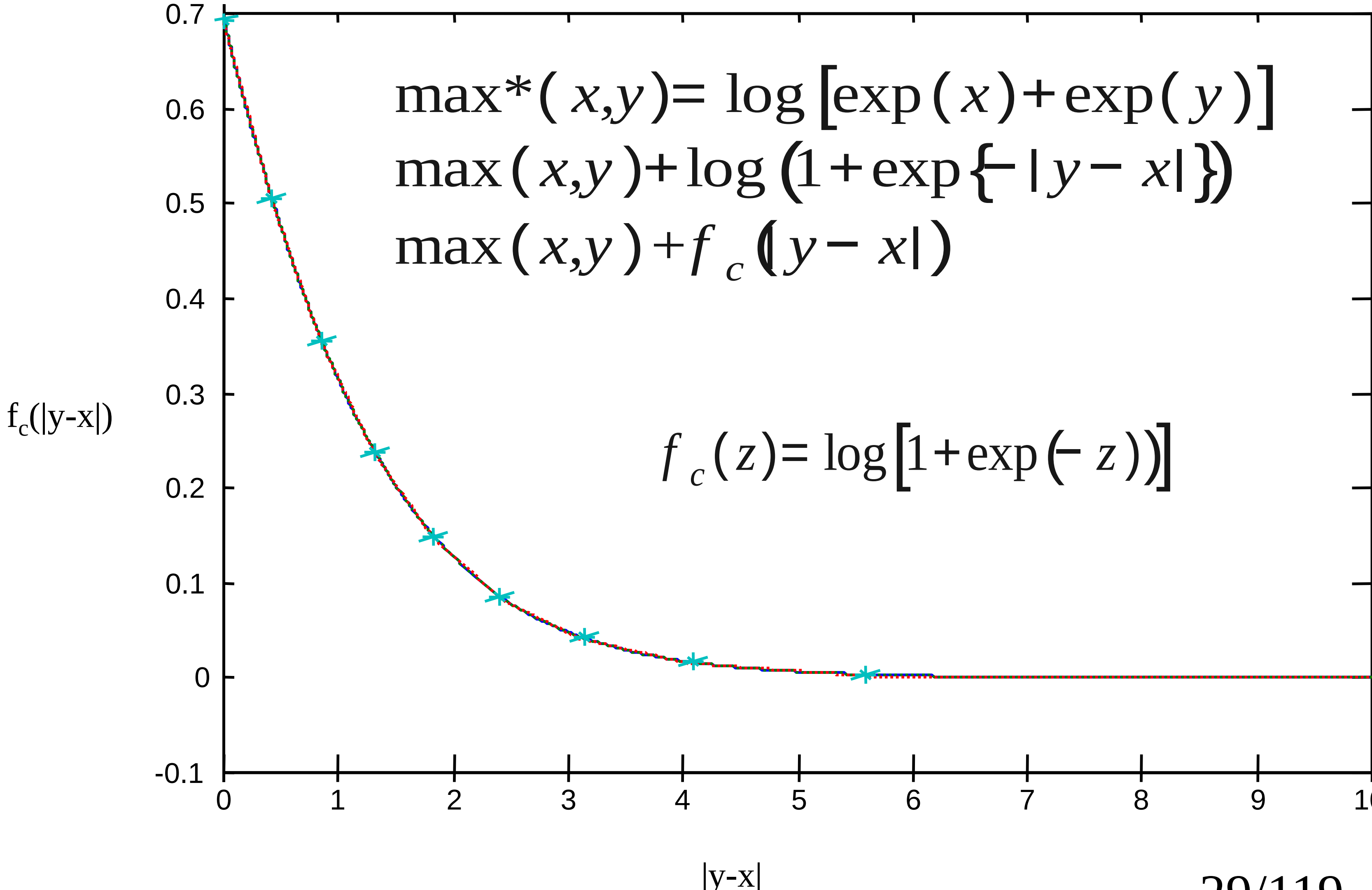
$$= \max_{X_k \in S_n^{(1)}}{}^{*} \{\log f(Y \mid X_k)\} - \max_{X_k \in S_n^{(0)}}{}^{*} \{\log f(Y \mid X_k)\}$$

log-MAP
demod_type = 0

$$\approx \max_{X_k \in S_n^{(1)}} \{\log f(Y \mid X_k)\} - \max_{X_k \in S_n^{(0)}} \{\log f(Y \mid X_k)\}$$

max-log-MAP
demod_type = 1

The max* function



$$\max{}^*(x,y) = \log\left[\exp(x)+\exp(y)\right]$$
$$\max(x,y)+\log\left(1+\exp\{-|y-x|\}\right)$$
$$\max(x,y)+f_c(|y-x|)$$

$$f_c(z) = \log\left[1+\exp(-z)\right]$$

FSK-Specific Parameters

- sim_param(record).
  - h = {scalar}
    - The modulation index
    - h=1 is orthogonal
  - csi_flag = {integer scalar}

    0 = coherent (only available when h=1)

    1 = noncoherent w/ perfect amplitudes

    2 = noncoherent without amplitude estimates

Specifying the Channel

- sim_param(record).
  - channel = {'AWGN', 'Rayleigh', 'block'}
    - 'Rayleigh' is "fully-interleaved" Rayleigh fading
    - 'block' is for coded simulation type only
  - blocks_per_frame = {scalar integer}
    - For block channel only.
    - Number of independent blocks per frame.
    - Block length is framesize/blocks_per_frame
  - bicm = {integer scalar}
    - 0 do not interleave bits prior to modulation
    - 1 interleave bits prior to modulation (default)
    - 2 interleave and perform iterative demodulation/decoding
    - This option is irrelevant unless a channel code is used

Exercises

- Create and run the following simulations:
  - BPSK in AWGN
  - 64QAM with gray labeling in AWGN
  - 64QAM with gray labeling in Rayleigh fading
- Choices that need to be made?
  - Framesize?
  - Save_rate?
  - Min_BER?
  - Min_frame_errors?
  - Demod_type?
- Plot all the results on the same figure.

Outline

1. CML overview
   - What is it? How to set it up and get started?

2. Uncoded Modulation
   - Simulate uncoded BPSK and QAM in AWGN and Rayleigh fading

3. Coded Modulation
   - Simulate a turbo code from UMTS 25.212
   - *Specify a custom eIRA LDPC parity check matrix

4. Ergodic (Shannon) capacity analysis
   - Determine the modulation constrained capacity of BPSK and QAM

5. EXIT analysis
   - Determine the EXIT characteristic for BPSK in AWGN for a particular LDPC degree distribution

Coded Systems: Code Configuration

- Only for sim_param(record).sim_type = ‘coded’
- sim_param(record).code_configuration = {scalar int}
  - 0 = Convolutional
  - 1 = binary turbo code (PCCC)
  - 2 = LDPC
  - 3 = HSDPA turbo code
  - 4 = UMTS turbo code with rate matching
  - 5 = WiMAX duobinary tailbiting turbo code (CTC)
  - 6 = DVB-RCS duobinary tailbiting turbo code

Convolutional Codes

- Only rate 1/n mother codes supported.
  - Can puncture to higher rate.
- Code is always terminated by a tail.
  - Can puncture out the tail.
- sim_param(record).
  - g1 = {binary matrix}
    - Example: (133,171) code from Proakis
      - g1 = [1 0 1 1 0 1 1
        1 1 1 1 0 0 1];
    - Constraint length = number of columns
    - Rate 1/n where n is number of rows.
  - nsc_flag1 = {scalar integer}
    - 0 for RSC
    - 1 for NSC
- Can handle cyclic block codes as a rate 1 terminated RSC code

Convolutional Codes: Decoding Algorithms

- sim_param(record).decoder_type = {integer scalar}

  negative value for Viterbi algorithm

  0 = log-MAP (approximated linearly)

  1 = max-log-MAP

  2 = constant-log-MAP

  3 and 4 other implementations of log-MAP

- Decodes over entire trellis (no sliding window traceback)

Punctured Convolutional Codes

- sim_param(record).
  - pun_pattern1 = {binary matrix}
    - Puncturing pattern
    - n rows
    - arbitrary number of columns (depends on puncture period)
    - 1 means keep bit, 0 puncture it.
    - number greater than 1 is number of times to repeat bit.
  - tail_pattern1 = {binary matrix}
    - tail can have its own puncturing pattern.

Turbo Codes

- sim_param(record).
  - Parameters for first constituent code
    - g1
    - nsc_flag1
    - pun_pattern1
    - tail_pattern1
  - Parameters for second constituent code
    - g2
    - nsc_flag2
    - pun_pattern2
    - tail_pattern2

Turbo Codes (cont'd)

- sim_param(record).
  - code_interleaver = {string}
    - A string containing the command used to generate the interleaver.
    - Examples include:
      - "CreateUmtsInterleaver(5114)" % UMTS interleaver.
      - "CreateLTEInterleaver(6144)" % LTS interleaver.
      - "CreateCCSDSInterleaver(8920)" % CCSDS interleaver.
      - "randperm(40)-1" % a random interleaver of length 40.
      - Can replace above lengths with other valid lengths.
  - decoder_type = {integer scalar}
    - Same options as for convolutional codes (except no Viterbi allowed).
  - max_iterations = {integer scalar}
    - Number of decoder iterations.
    - Decoder will automatically halt once codeword is correct.
  - plot_iterations = {integer scalar}
    - Which iterations to plot, in addition to max_iterations

UMTS Rate Matching

- sim_param(record)
  - framesize = {integer scalar}
    - number of data bits
  - code_bits_per_frame = {integer scalar}
    - number of code bits
- When code_configuration = 4, automatically determines rate matching parameters according to UMTS (25.212)

HSDPA Specific Parameters

- sim_param(record).
  - N_IR = {integer scalar}
    - Size of the virtual IR buffer
  - X_set = {integer vector}
    - Sequence of redundancy versions (one value per ARQ transmission)
  - P = {integer scalar}
    - Number of physical channels per turbo codeword
- Examples from HSET-6 TS 25.101
  - N_IR = 9600
  - QPSK
    - framesize = 6438
    - X_set = [0 2 5 6]
    - P = 5 (i.e. 10 physical channels used for 2 turbo codewords)
  - 16-QAM
    - framesze = 9377
    - X_set = [6 2 1 5]
    - P = 4 (i.e. 8 physical channels used for 2 turbo codewords)

LDPC

- sim_param(record).
  - parity_check_matrix = {string}
    - A string used to generate the parity check matrix
  - decoder_type
    - 0 Sum-product (default)
    - 1 Min-sum
    - 2 Approximate-min-star
  - max_iterations
    - Number of decoder iterations.
    - Decoder will automatically halt once codeword is correct.
  - plot_iterations
    - Which iterations to plot, in addition to max_iterations

LDPC

- Specifying a Parity Check Matrix
  - Several methods available
    - DVB-S2 type via CML function 'InitializeDVBS2()'
    - Flat-text alist file [1]
    - CML-native H_rows, H_cols format stored as .mat file
  - All parity check matrix data files must be stored in <CMLROOT>/data/ldpc

    Example:
    <CMLROOT>/data/ldpc/test_ldpc_hmat.alist

[1] http://www.inference.phy.cam.ac.uk/mackay/codes/alist.html

LDPC

- Specifying a Parity Check Matrix
  Function InitializeDVBS2()
  - Generates parity check matrix conforming to DVB-S2 standard
  - Example

  CML Scenario: LdpcHmat

  Record: 1

LDPC

- Specifying a Parity Check Matrix
  File-based specification
  - Consider an eIRA LDPC code having rate K/N
  - The parity check matrix has form

$$
\begin{array}{cc}
 & \overbrace{\hspace{12em}}^{N} \\
 & \overbrace{\hspace{6em}}^{K}\;\overbrace{\hspace{6em}}^{N-K} \\
H = & \left[\begin{array}{c:ccccc}
 & 1 & 1 & & & \\
 & & 1 & 1 & & \\
(1) & & & \ddots & & \\
 & & & & 1 & 1 \\
 & & & & & 1
\end{array}\right] \left.\vphantom{\begin{array}{c}1\\1\\1\\1\\1\end{array}}\right\} N-K
\end{array}
$$

  - User-specified parity check matrices define the N-K x K sub-matrix denoted by (1)

LDPC

- Specifying a Parity Check Matrix
  H_rows, H_cols format
  - Parity check matrix stored as .mat file
  - File contains two variables, H_rows and H_cols
    - H_rows: N-K x R double matrix
    - H_cols: K x C double matrix

    where R denotes the maximum weight of all rows and
    C denotes the maximum weight of all columns
  - First dimension of H_rows and H_cols denotes a particular parity check matrix row or column, respectively.
  - Each entry of H_rows and H_cols denotes the location of a “1” for row or column specified by the first dimension with entry “0” denoting the absence of a “1”.

LDPC

- Specifying a Parity Check Matrix
  H_rows, H_cols format
  - Trivial Example

$$H = \left[\begin{array}{cccc|cccc} 1 & 1 & & 1 & 1 & 1 & & \\ 1 & & & 1 & & 1 & 1 & \\ & 1 & 1 & & & & 1 & 1 \\ & & 1 & & & & & 1 \end{array}\right]$$

$$H\_rows = \begin{bmatrix} 1 & 2 & 4 \\ 1 & 4 & 0 \\ 2 & 3 & 0 \\ 3 & 0 & 0 \end{bmatrix} \qquad H\_cols = \begin{bmatrix} 1 & 2 \\ 1 & 3 \\ 3 & 4 \\ 1 & 2 \end{bmatrix}$$

LDPC

- Specifying a Parity Check Matrix
  alist format
  - Parity check matrix stored as .alist file in
    - <CMLROOT>/data/ldpc
  - Parity check matrix must obey eIRA constraint
  - For a full description of the alist format, see
  http://www.inference.phy.cam.ac.uk/mackay/codes/alist.html

LDPC

- Specifying a Parity Check Matrix
  Examples

CML Scenario: LdpcHmat

- Record 1: InitializeDVBS2()
- Record 2: alist
  - Data file: <CMLROOT>/data/ldpc/test_ldpc_hmat.alist
- Record 3: H_rows, H_cols
  - Data file: <CMLROOT>/data/ldpc/test_ldpc_hmat.mat

Block Fading

- For coded simulations, block fading is supported.
- Sim_param(record).channel = ‘block’
- Sim_param(record).blocks_per_frame
  - The number of independent blocks per frame
- Example, HSDPA with independent retransmissions
  - blocks_per_frame = length(X_set);

Exercises

- Simulate
  - A convolutional code with g=(7,5) over AWGN with BPSK
  - The same convolutional code punctured to rate 3/4.
  - The UMTS turbo code with 16-QAM
    - Unpunctured w/ 640 input bits
    - Punctured to force the rate to be 1/2.
    - Compare log-MAP and max-log-MAP
  - HSDPA
    - HSET-6
    - Quasi-static block fading

Outline

1. CML overview
   - What is it? How to set it up and get started?

2. Uncoded Modulation
   - Simulate uncoded BPSK and QAM in AWGN and Rayleigh fading

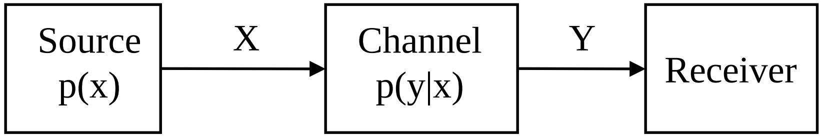
3. Coded Modulation
   - Simulate a turbo code from UMTS 25.212
   - *Specify a custom eIRA LDPC parity check matrix

4. Ergodic (Shannon) capacity analysis
   - Determine the modulation constrained capacity of BPSK and QAM

5. EXIT analysis
   - Determine the EXIT characteristic for BPSK in AWGN for a particular LDPC degree distribution

Noisy Channel Coding Theorem (Shannon 1948)

- Consider a memoryless channel with input X and output Y



  – The channel is completely characterized by p(x,y)
- The ***capacity*** C of the channel is

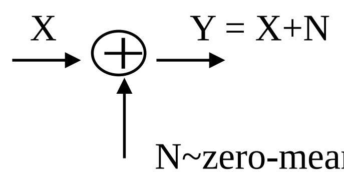
$$C = \max_{p(x)} I(X;Y) = \max_{p(x)} \left\{ \iint p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \, dxdy \right\}$$

  – where I(X,Y) is the (average) ***mutual information*** between X and Y.
- The channel capacity is an upper bound on ***information rate*** r.
  – There exists a code of rate r < C that achieves reliable communications.
  – "Reliable" means an arbitrarily small error probability.

Capacity of the AWGN Channel with Unconstrained Input

- Consider the one-dimensional AWGN channel

The input X is drawn from ***any*** distribution with average energy $E[X^2] = E_s$



N~zero-mean white Gaussian with energy $E[N^2]= N_0/2$

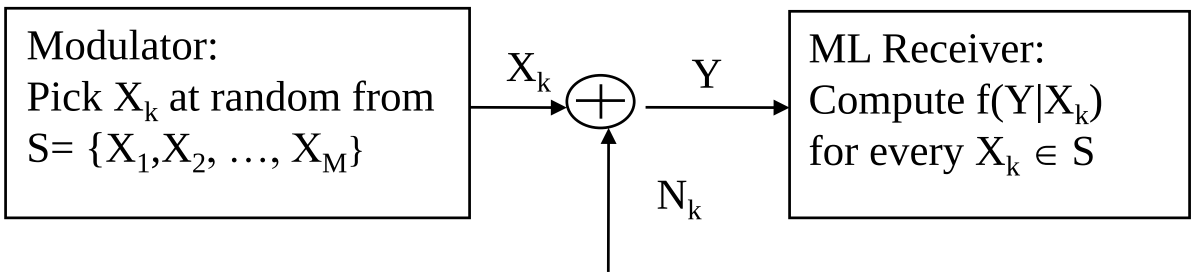
- The capacity is

$$I(X;Y) = \frac{1}{2}\log_2\left(\frac{2E_s}{N_o} + 1\right)$$ bits per channel use

- The X that attains capacity is Gaussian distributed.
  - Strictly speaking, Gaussian X is not practical.

Capacity of the AWGN Channel with a Modulation-Constrained Input

- Suppose X is drawn with equal probability from the finite set $S = \{X_1, X_2, \ldots, X_M\}$



  – where $f(Y|X_k) = \kappa\, p(Y|X_k)$ for any κ common to all X_k

- Since p(x) is now fixed

$$\max_{p(x)}\{I(X;Y)\} = I(X;Y)$$

  – i.e. calculating capacity boils down to calculating mutual info.

Entropy and Conditional Entropy

- Mutual information can be expressed as:

$$I(X;Y) = H(X) - H(X \mid Y)$$

- Where the ***entropy*** of X is

$$H(X) = E[h(X)] = \int p(x)h(x)dx$$

where $$\underbrace{h(x) = \log\frac{1}{p(x)} = -\log p(x)}_{\text{self-information}}$$

- And the ***conditional entropy*** of X given Y is

$$H(X \mid Y) = E[h(X \mid Y)] = \iint p(x,y)h(x \mid y)dxdy$$

where $h(x \mid y) = -\log p(x \mid y)$

Calculating Modulation-Constrained Capacity

- To calculate:

$$I(X;Y) = H(X) - H(X \mid Y)$$

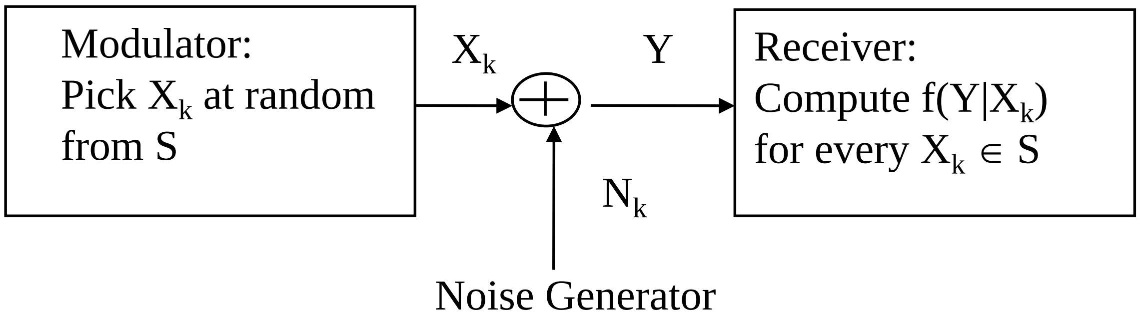
- We first need to compute H(X)

$$\begin{aligned} H(X) &= E[h(X)] \\ &= E\left[\log \frac{1}{p(X)}\right] \\ &= E[\log M] \\ &= \log M \end{aligned}$$

$$p(X) = \frac{1}{M}$$

- Next, we need to compute H(X|Y)=E[h(X|Y)]
  - This is the “hard” part.
  - In some cases, it can be done through numerical integration.
  - Instead, let’s use Monte Carlo simulation to compute it.

Step 1: Obtain p(x|y) from f(y|x)



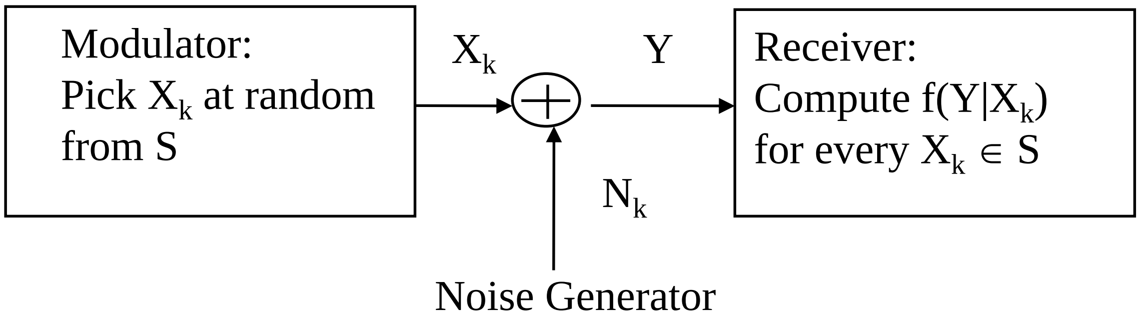
- Since

$$\sum_{x' \in S} p(x'|y) = 1$$

- We can get p(x|y) from

$$p(x|y) = \frac{p(x|y)}{\sum_{x' \in S} p(x'|y)} = \frac{\dfrac{p(y|x)\,p(x)}{p(y)}}{\sum_{x' \in S} \dfrac{p(y|x')\,p(x')}{p(y)}} = \frac{f(y|x)}{\sum_{x' \in S} f(y|x')}$$

Step 2: Calculate h(x|y)



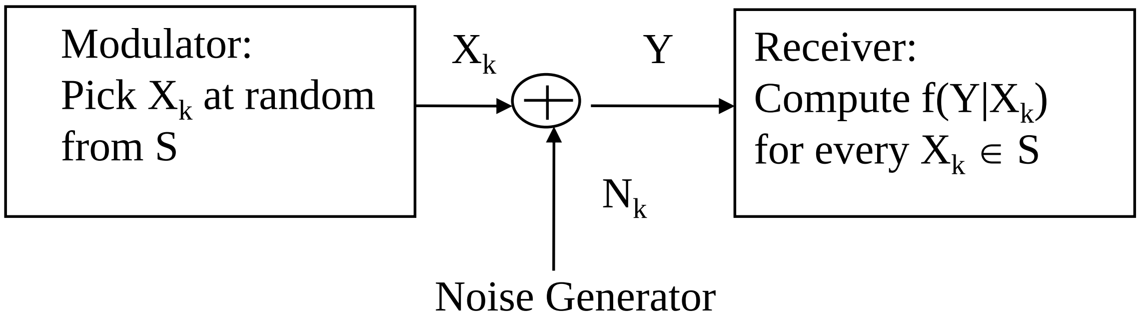
- Given a value of x and y (from the simulation) compute

$$p(x|y) = \frac{f(y|x)}{\sum_{x' \in S} f(y|x')}$$

- Then compute

$$h(x \mid y) = -\log p(x \mid y) = -\log f(y \mid x) + \log \sum_{x' \in S} f(y \mid x')$$

Step 3: Calculating H(X|Y)

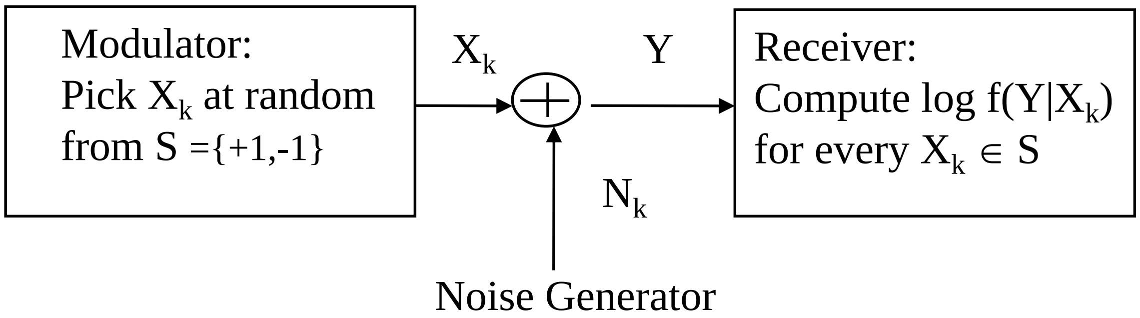


- Since: $$H(X\,|\,Y) = E[h(X\,|\,Y)] = \iint p(x,y)h(x\,|\,y)\,dx\,dy$$
- Because the simulation is ergodic, H(X|Y) can be found by taking the sample mean:

$$\frac{1}{N}\sum_{n=1}^{N} h(X^{(n)}\,|\,Y^{(n)})$$

- where $(X^{(n)},Y^{(n)})$ is the n^{th} realization of the random pair (X,Y).
  – i.e. the result of the n^{th} simulation trial.

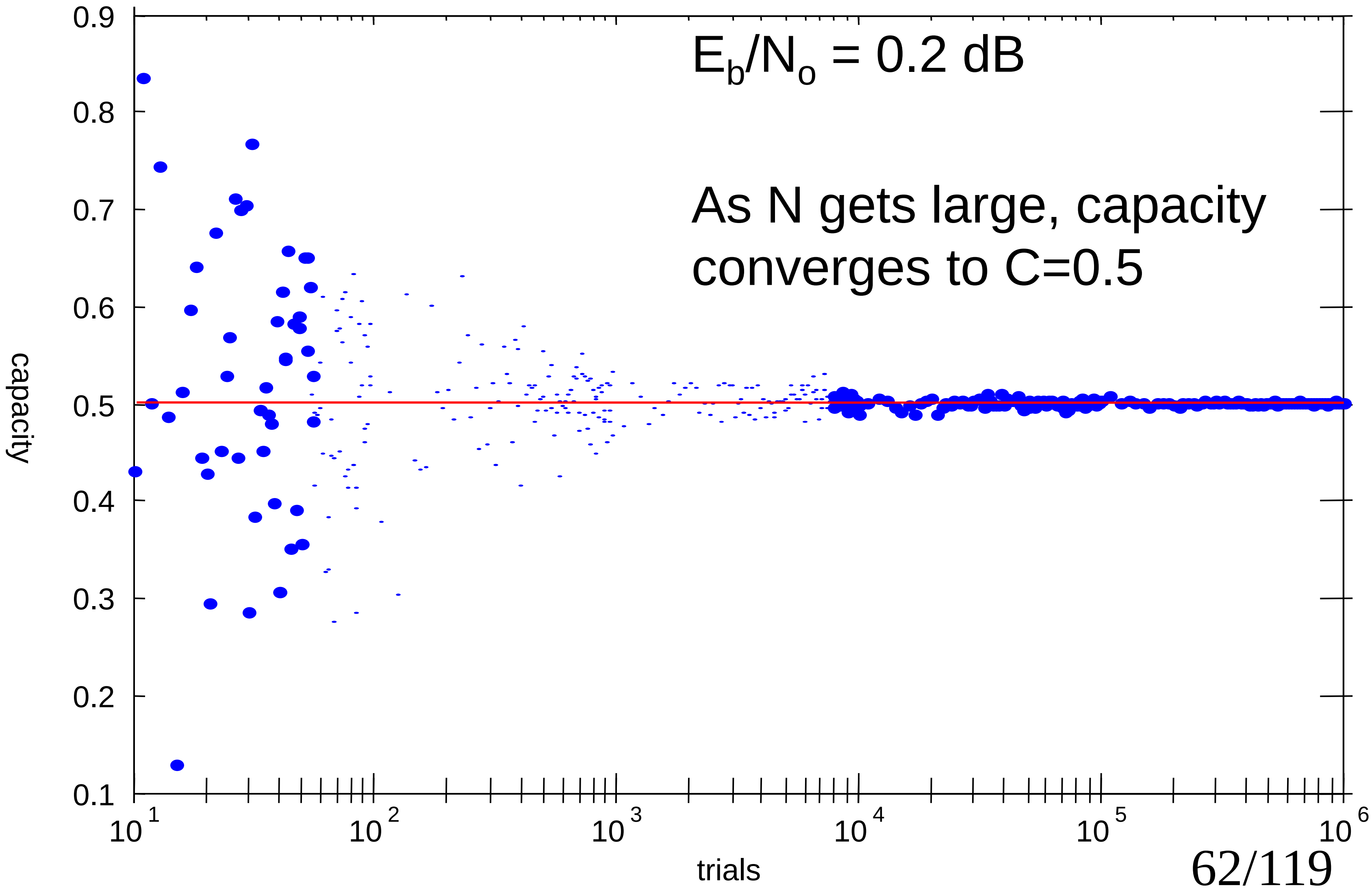
Example: BPSK



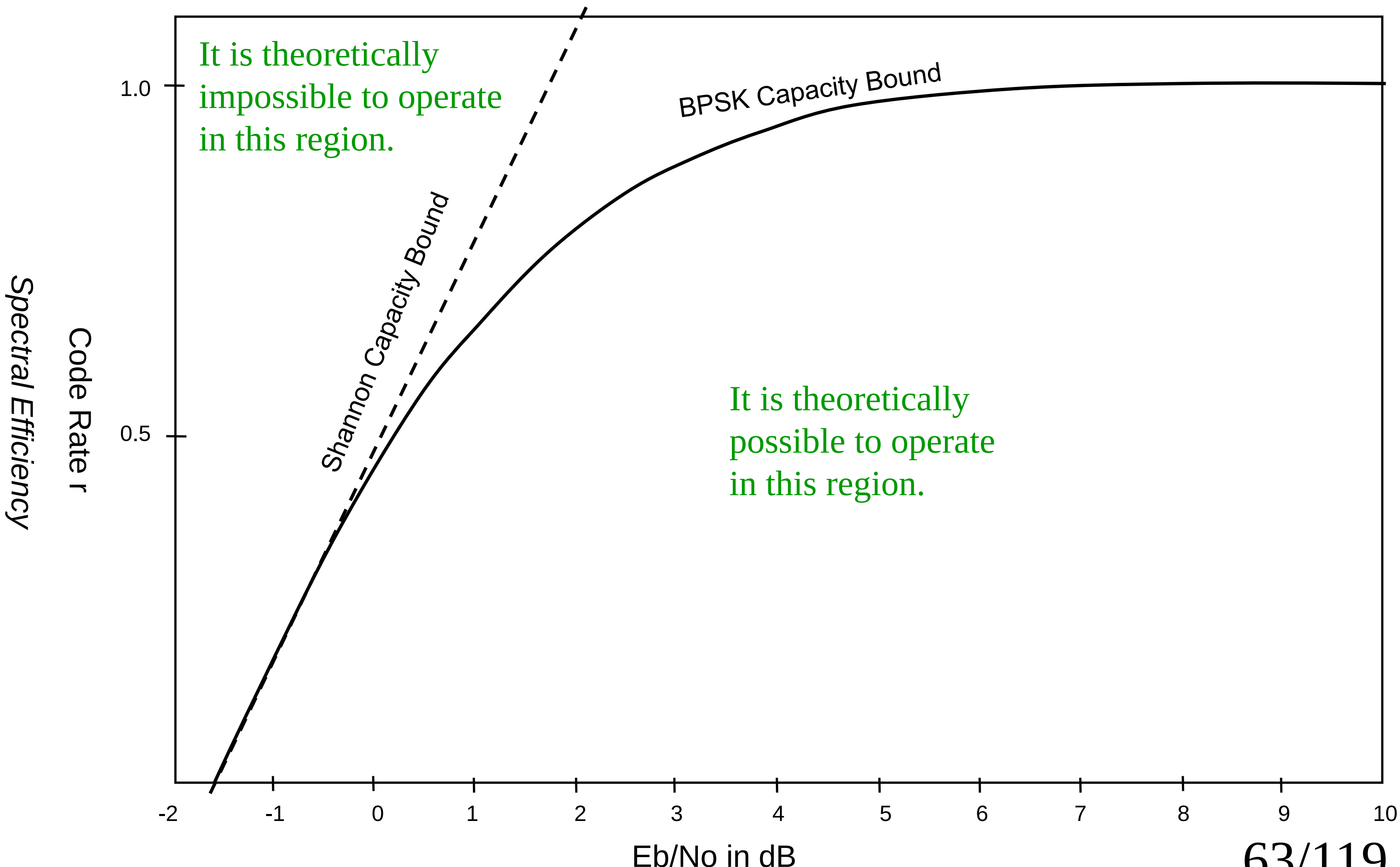
- Suppose that S ={+1,-1} and N has variance $N_0/2E_s$
- Then:

$$\log f(y|x) = -\frac{E_s}{N_o}\|y - x\|^2$$

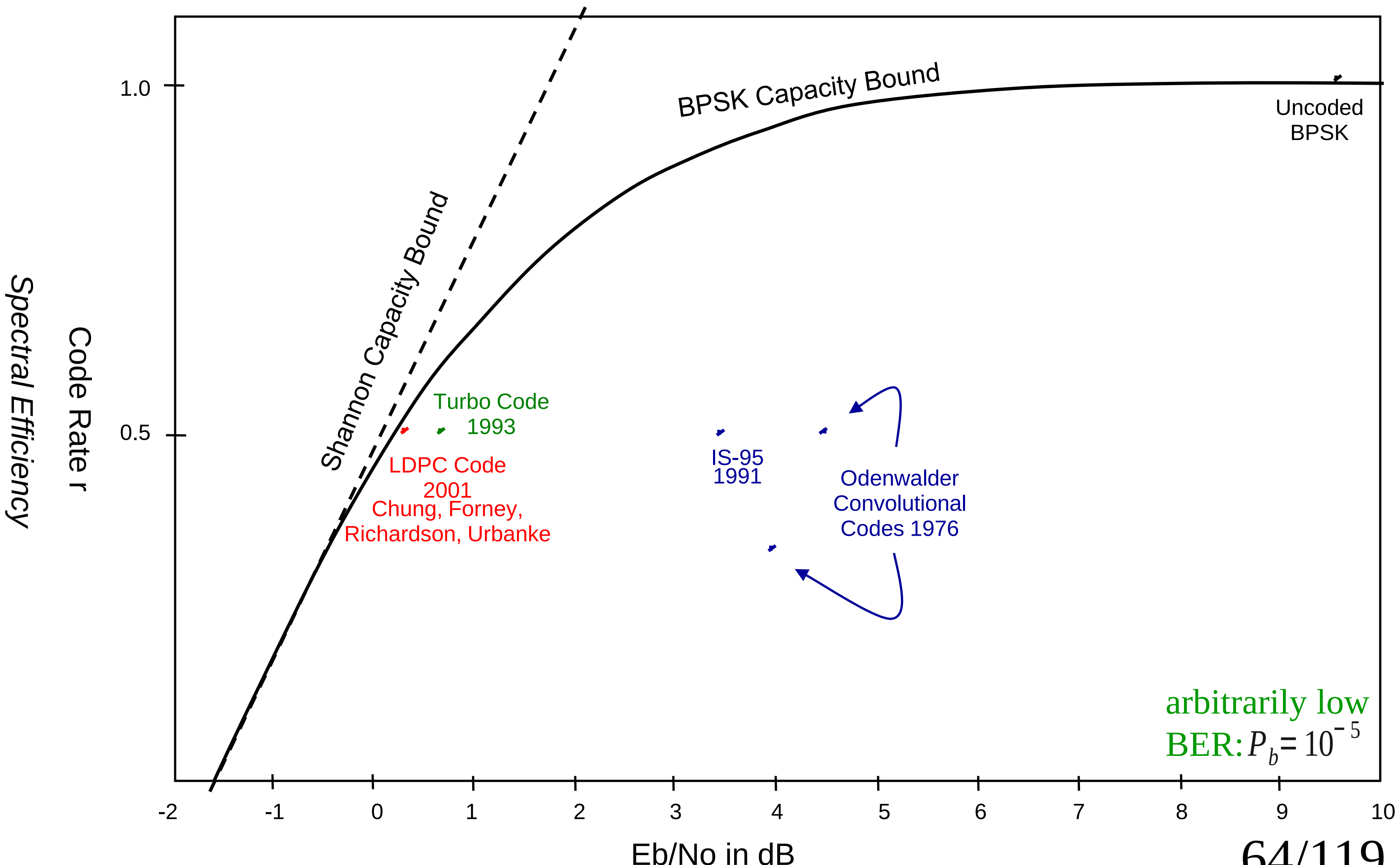
BPSK Capacity as a Function of Number of Simulation Trials

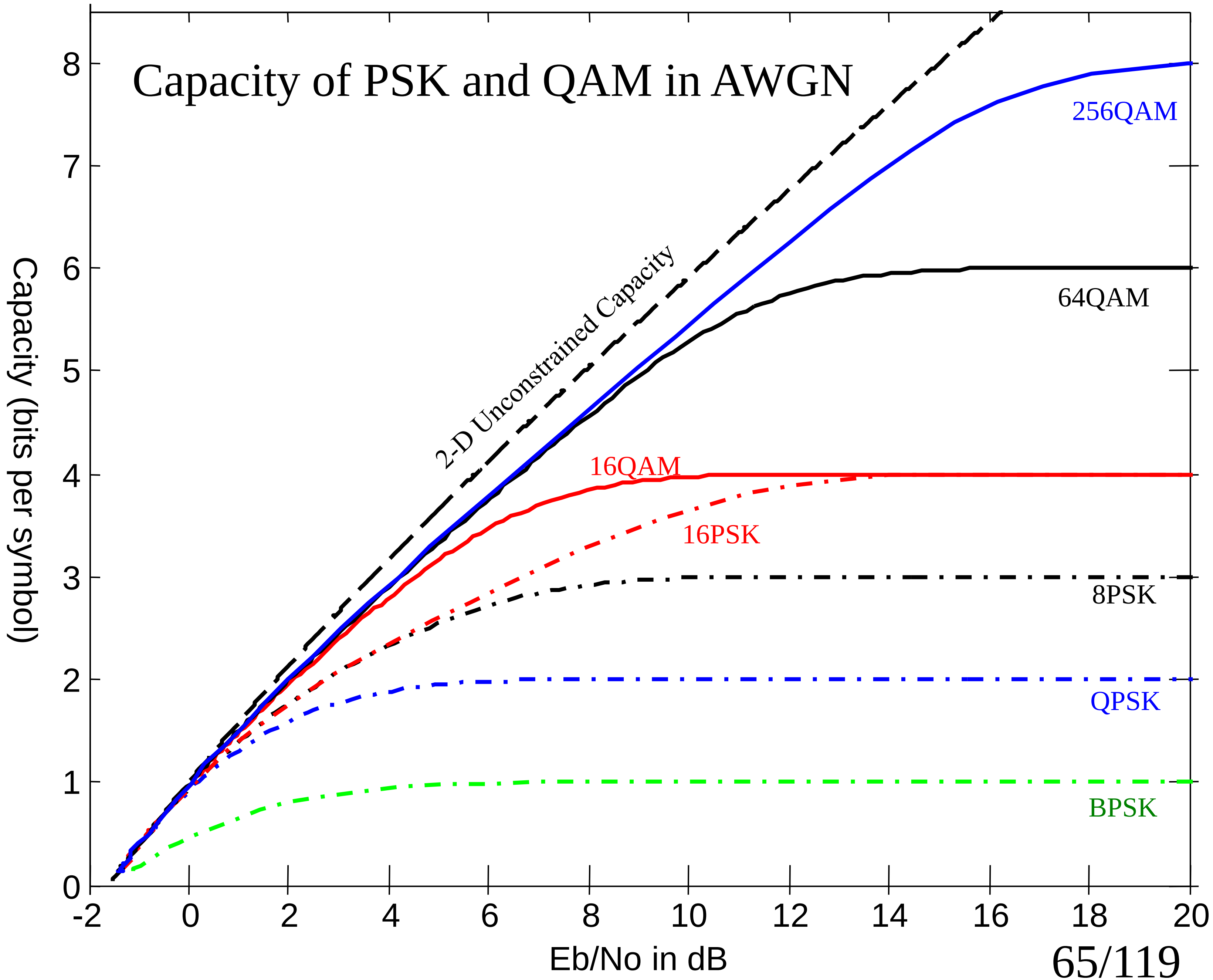


Unconstrained vs. BPSK Constrained Capacity



Power Efficiency of Standard Binary Channel Codes

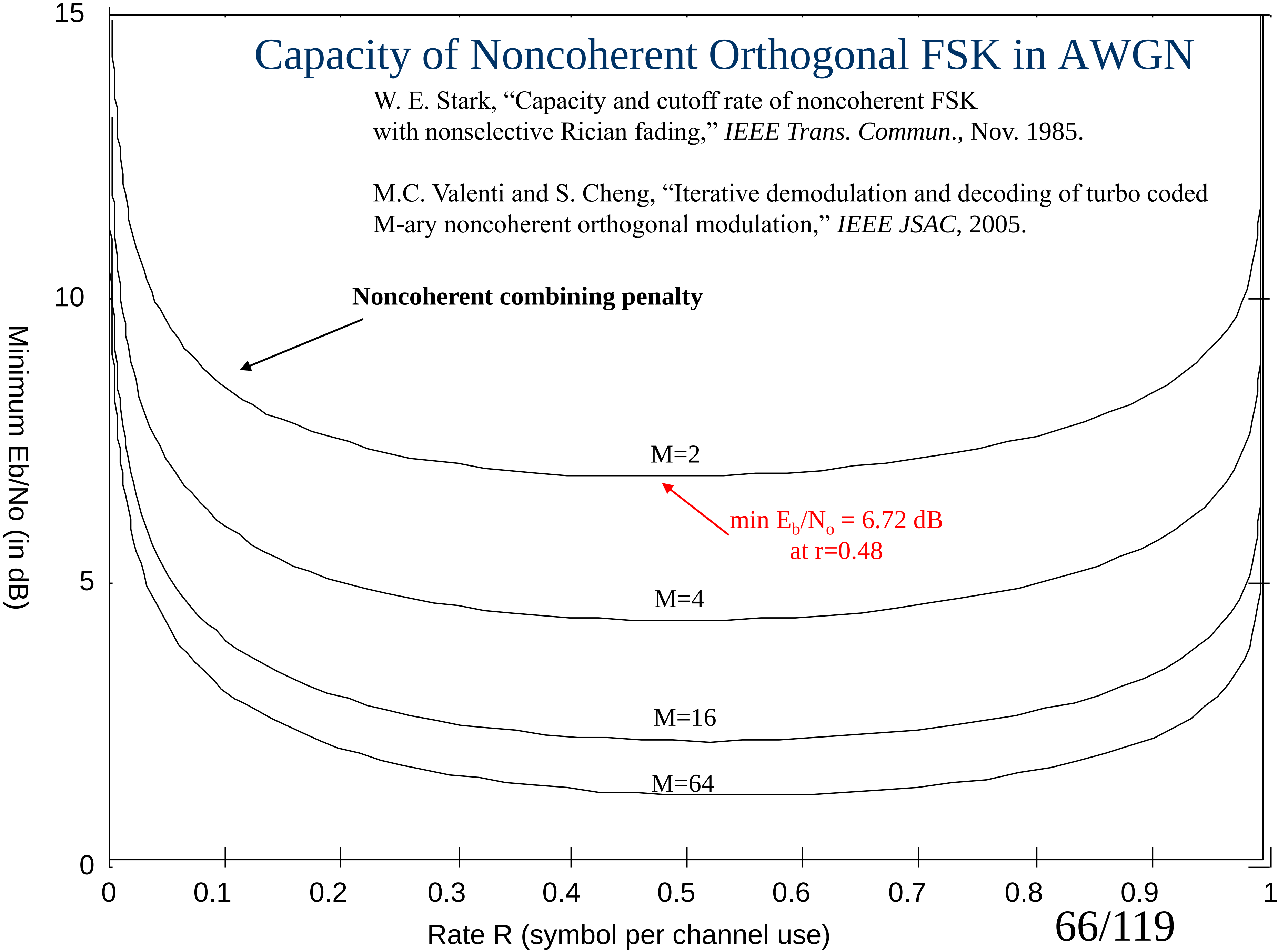



Capacity of PSK and QAM in AWGN
2-D Unconstrained Capacity
256QAM
64QAM
16QAM
16PSK
8PSK
QPSK
BPSK
Capacity (bits per symbol)
Eb/No in dB
0
1
2
3
4
5
6
7
8
-2
0
2
4
6
8
10
12
14
16
18
20

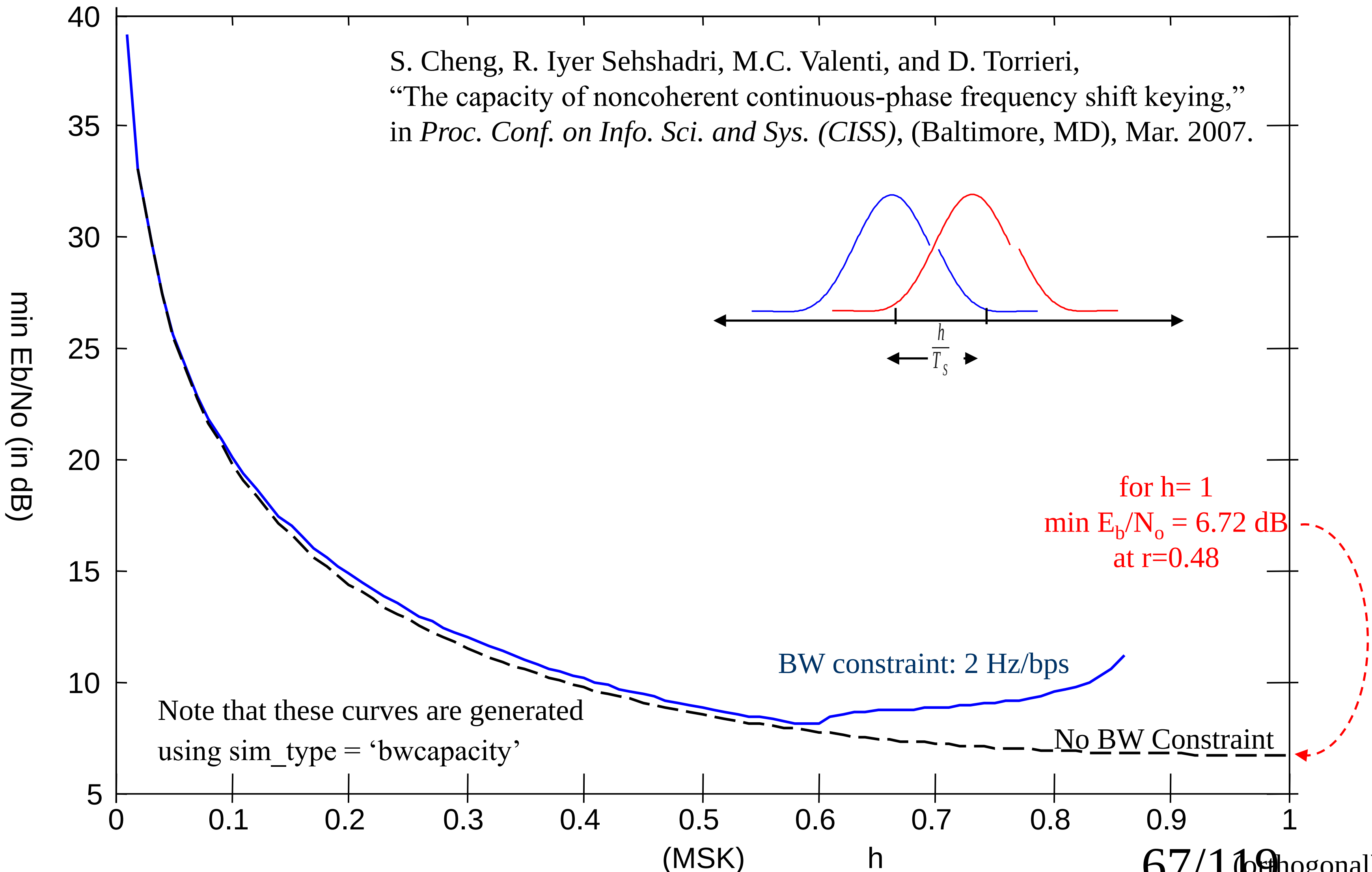
Capacity of Noncoherent Orthogonal FSK in AWGN

W. E. Stark, "Capacity and cutoff rate of noncoherent FSK with nonselective Rician fading," *IEEE Trans. Commun.*, Nov. 1985.

M.C. Valenti and S. Cheng, "Iterative demodulation and decoding of turbo coded M-ary noncoherent orthogonal modulation," *IEEE JSAC*, 2005.

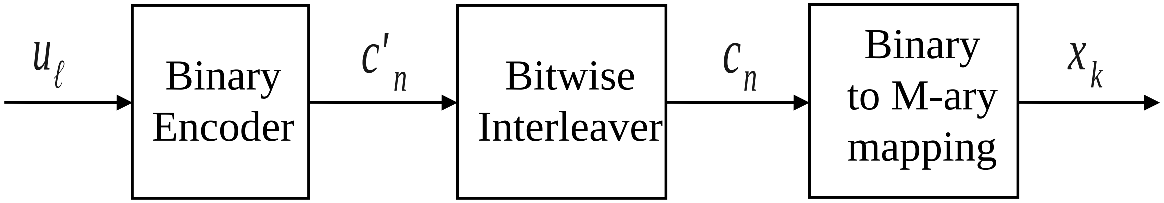


Capacity of Nonorthogonal CPFSK

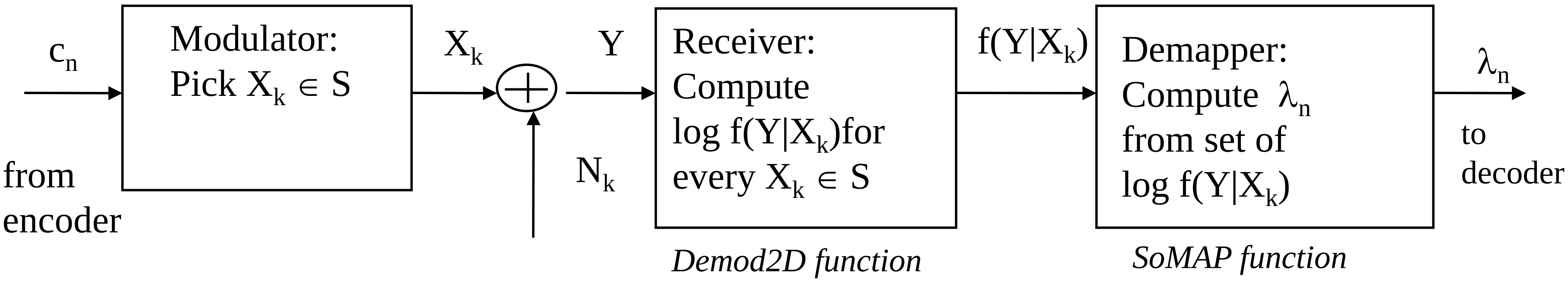


BICM
(Caire 1998)

- Coded modulation (CM) is required to attain the aforementioned capacity.
  - Channel coding and modulation handled jointly.
  - Alphabets of code and modulation are matched.
  - e.g. trellis coded modulation (Ungerboeck); coset codes (Forney)
- Most off-the-shelf capacity approaching codes are binary.
- A pragmatic system would use a binary code followed by a bitwise interleaver and an M-ary modulator.
  - Bit Interleaved Coded Modulation (BICM).

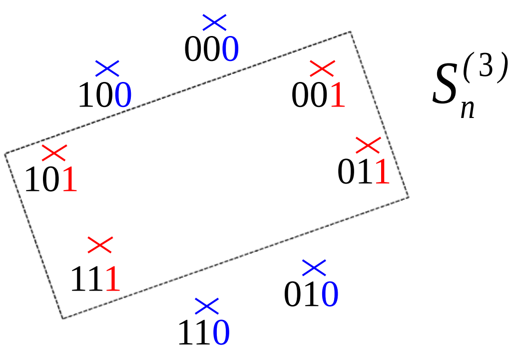


BICM Receiver



- The symbol likelihoods must be transformed into bit log-likelihood ratios (LLRs):

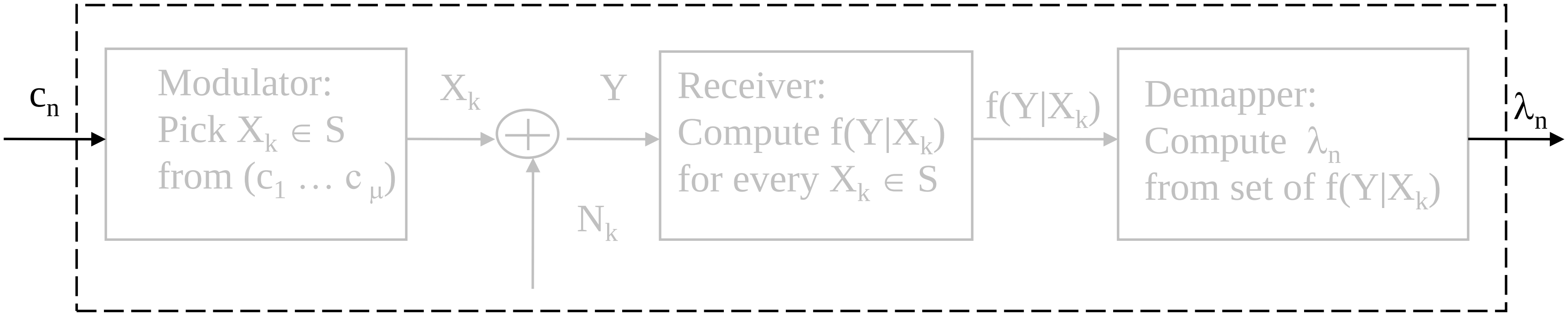
$$\lambda_n = \log \frac{\sum_{X_k \in S_n^{(1)}} f(Y \mid X_k)}{\sum_{X_k \in S_n^{(0)}} f(Y \mid X_k)}$$



– where $S_n^{(1)}$ represents the set of symbols whose nth bit is a 1.

– and $S_n^{(0)}$ is the set of symbols whose nth bit is a 0.

BICM Capacity



- Can be viewed as $\mu=\log_2 M$ binary parallel channels, each with capacity

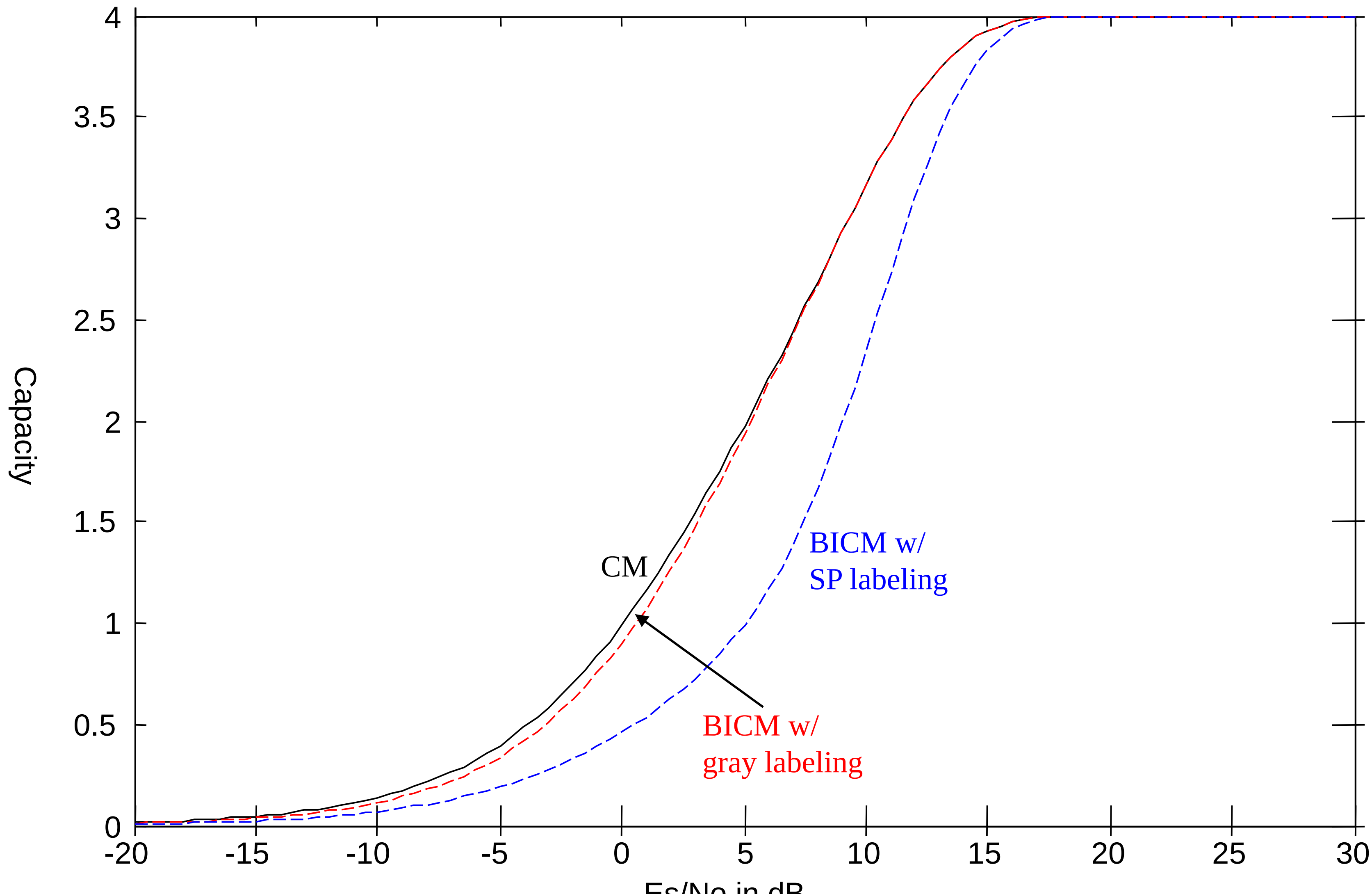
$$C_n = I(c_n, \lambda_n)$$

- Capacity over parallel channels adds:

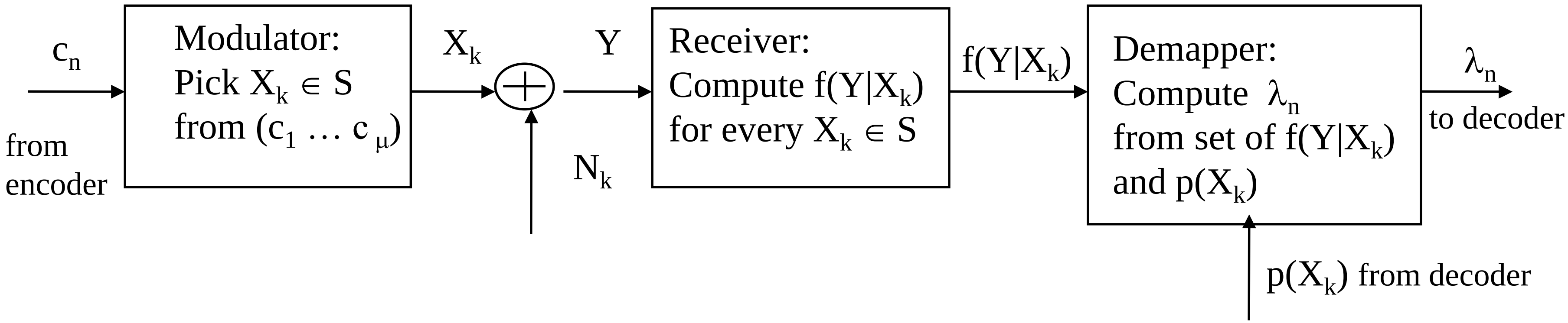
$$C = \sum_{n=1}^{\mu} C_n$$

- As with the CM case, Monte Carlo integration may be used.

CM vs. BICM Capacity for 16QAM



BICM-ID
(Li & Ritcey 1997)



- A SISO decoder can provide side information to the demapper in the form of a priori symbol likelihoods.
  - BICM with Iterative Detection The demapper's output then becomes

$$\lambda_n = \log \frac{\sum_{x_k \in S_n^{(1)}} f(Y \mid X_k) p(X_k)}{\sum_{x_k \in S_n^{(0)}} f(Y \mid X_k) p(X_k)}$$

Capacity Simulations in CML

- sim_param(record).sim_type = ‘capacity’
- Exact same parameters as for uncoded simulations
  - SNR
  - SNR_type = ‘Es/No in dB’
  - framesize
  - modulation
  - mod_order
  - channel
  - bicm
  - demod_type
  - max_trials

Exercises

- Determine the capacity for
  - BPSK in AWGN
  - 64QAM with gray labeling in AWGN
  - 64QAM with gray labeling in Rayleigh fading
- Setup BICM-ID for
  - 16-QAM with SP mapping in AWGN and (7,5) CC.

Outline

1. CML overview
   - What is it? How to set it up and get started?

2. Uncoded Modulation
   - Simulate uncoded BPSK and QAM in AWGN and Rayleigh fading

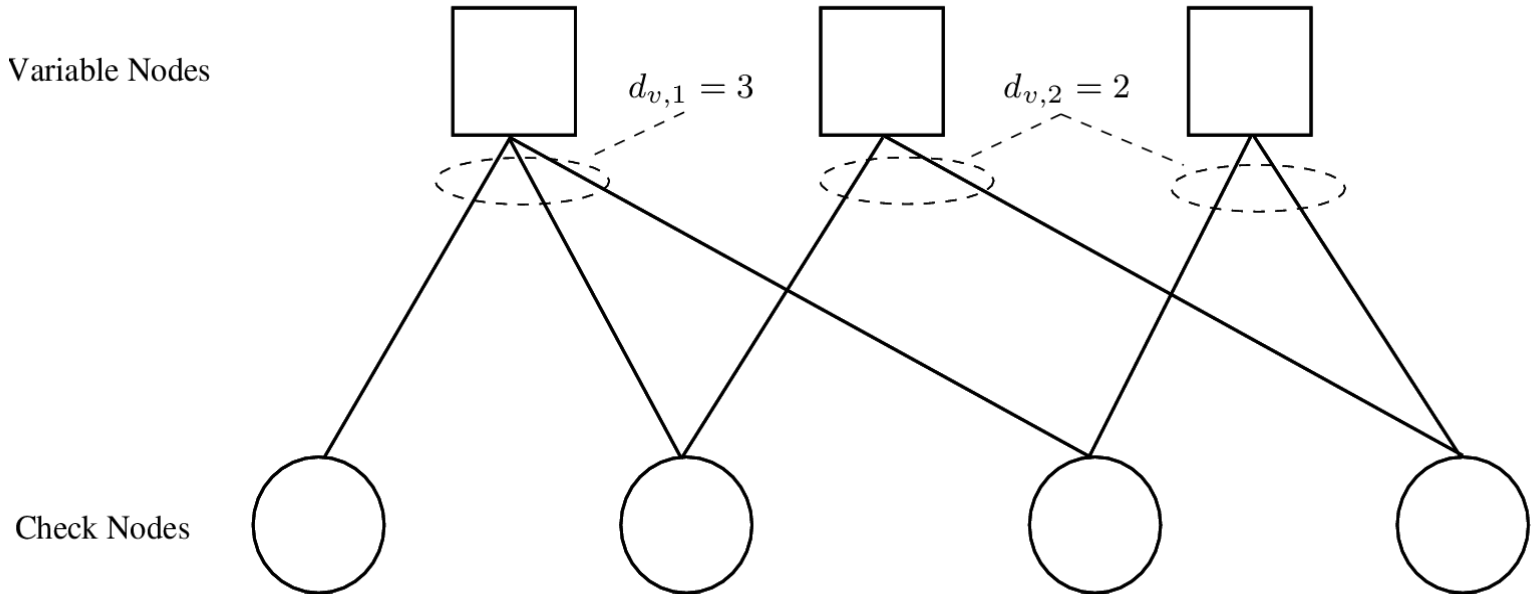
3. Coded Modulation
   - Simulate a turbo code from UMTS 25.212
   - *Specify a custom eIRA LDPC parity check matrix

4. Ergodic (Shannon) capacity analysis
   - Determine the modulation constrained capacity of BPSK and QAM

5. EXIT analysis
   - Determine the EXIT characteristic for BPSK in AWGN for a particular LDPC degree distribution

EXIT Analysis

– EXIT analysis may be applied to design LDPC code degree distributions which optimize decoding performance



EXIT Analysis

- Design goal: match mutual information transfer characteristics of LDPC decoders through selection of variable node degrees
  - VND - variable node decoder
  - CND - check node decoder
- VND decoder captures characteristic of detector and variable node decoder
- CND curve depends only on channel code parameters

EXIT Analysis

– First step of generating VND transfer characteristic is to compute the mutual information at the input and output of the detector

$$I_{E,DET}(I_A, E_S / N_0)$$

– where

I_A mutual information of a-prior LLRs at detector input

– Any detector supported by CML may be analyzed.

EXIT Analysis

– The extrinsic information at the output of the combined variable node decoder/detector is

$$I_{E,VND}(I_A, d_v, E_S / N_0) = \ldots$$

$$J\left(\sqrt{(d_v - 1)[J^{-1}(I_A)^2] + [J^{-1}(I_{E,DET}(I_A, E_S/N_0))]^2}\right)$$

– where

d_v variable node degree

$J()$ function computing the mutual information $I(X, \Lambda(Y))$ [1]

and $Y = X + N$ is an AWGN channel where X is drawn from a BPSK constellation, and $\Lambda(Y)$ is the LLR of the channel output.

[1] S. ten Brink, G. Kramer, and A. Ashikhmin, "Design of low-density parity-check codes for modulation and detection," IEEE Trans. Commun., vol. 52, pp. 670–678, Apr. 2004.

EXIT Analysis

– Considering an LDPC decoder having multiple variable node degrees (irregular)

$$I_{E,\ VND}(I_A, E_S / N_0) = \sum b_i \cdot I_{E,\ VND}(I_A, d_{v,i}, E_S / N_0)$$

– where

D number of distinct variable node degrees

b_i fraction of edges incident on variable nodes of degree $d_{v,i}$

$d_{v,i}$ i-th variable node degree

EXIT Analysis - LDPC

– The extrinsic information at the output of the CND is

$$I_{E,CND}(I_A, d_c) = 1 - J\left(\sqrt{d_c - 1} \cdot J^{-1}(1 - I_A)\right)$$

– where

d_c check node degree (check regular)

– Note that the CND transfer characteristic depends only on the code parameters.

EXIT Simulation Example

- First step: Simulate detector characteristic
  CML Scenario: ExitP2P

Record: 1

- sim_param(record).sim_type = 'exit'
- sim_param(record).exit_param.exit_phase = 'detector'
- sim_param(record).exit_param.exit_type = 'ldpc'
- sim_param(record).exit_param.requested_IA = 0 : 0.01 : 0.9
- sim_param(record).channel = 'awgn'
- sim_param(record).SNR = [0 : 0.5 : 10]
- sim_param(record).modulation = 'psk'
- sim_param(record).mod_order = 4

EXIT Simulation Example

- Second step: Simulate decoder characteristic
  CML Scenario: ExitP2P

Record: 2

- sim_param(record).sim_type = 'exit'
- sim_param(record).exit_param.exit_phase = 'decoder'
- sim_param(record).exit_param.exit_type = 'ldpc'
- sim_param(record).exit_param.requested_IA = 0 : 0.01 : 0.9
- sim_param(record).channel = 'awgn'
- sim_param(record).SNR = [0 : 0.5 : 10]
- sim_param(record).modulation = 'psk'
- sim_param(record).mod_order = 4

EXIT Simulation Example

- Second step: Simulate decoder characteristic
  CML Scenario: ExitP2P

Record: 2

- sim_param(record).exit_param.rate = 0.6
- sim_param(record).exit_param.dv = [2   4   19]
- sim_param(record).exit_param.dv_dist = [0.4   0.52 0.08]
- sim_param(record).exit_param.dc = 11
- sim_param(record).exit_param.det_scenario = 'ExitP2P'
- sim_param(record).exit_param.det_record = 1

EXIT Simulation Example

User Simulation Steps

Start MATLAB and initialize CML

```
>> CmlStartup
```

Execute detector simulation

```
>> CmlSimulate('ExitP2P', 1)
```

Execute decoder simulation

```
>> CmlSimulate('ExitP2P', 2)
```

Plot results

```
>> CmlPlot('ExitP2P', 2,1)   % Red argument: SNR to plot
```

Outline

6. Outage analysis

- **Determine the outage probability over block fading channels**
- **Determine the outage probability of finite-length codes**

7. Digital Network Coding

- Describe CML implementation of digital network coding
- Simulate error rate of digital network coding for non-channel-coded 4-FSK in Rayleigh fading

8. The internals of CML

9. Throughput calculation

- Convert BLER to throughput for hybrid-ARQ

Ergodicity vs. Block Fading

- Up until now, we have assumed that the channel is ***ergodic***.
  - The observation window is large enough that the time-average converges to the statistical average.
- Often, the system might be ***nonergodic***.
- Example: ***Block fading***

b=1	b=2	b=3	b=4	b=5
γ_1	γ_2	γ_3	γ_4	γ_5

The codeword is broken into B equal length blocks
The SNR changes randomly from block-to-block
The channel is conditionally Gaussian
The instantaneous Es/No for block b is γ_b

Accumulating Mutual Information

- The SNR γ_b of block b is a random.
- Therefore, the mutual information I_b for the block is also random.
  - With a complex Gaussian input, $I_b = \log(1+\gamma_b)$
  - Otherwise the modulation constrained capacity can be used for I_b

b=1	b=2	b=3	b=4	b=5
$I_1 = \log(1+\gamma_1)$	I_2	I_3	I_4	I_5

The mutual information of each block is $I_b = \log(1+\gamma_b)$
Blocks are conditionally Gaussian
The entire codeword's mutual info is the sum of the blocks'

$$I_1^B = \sum_{b=1}^{B} I_b$$

(Code combining)

Information Outage

- An ***information outage*** occurs after B blocks if

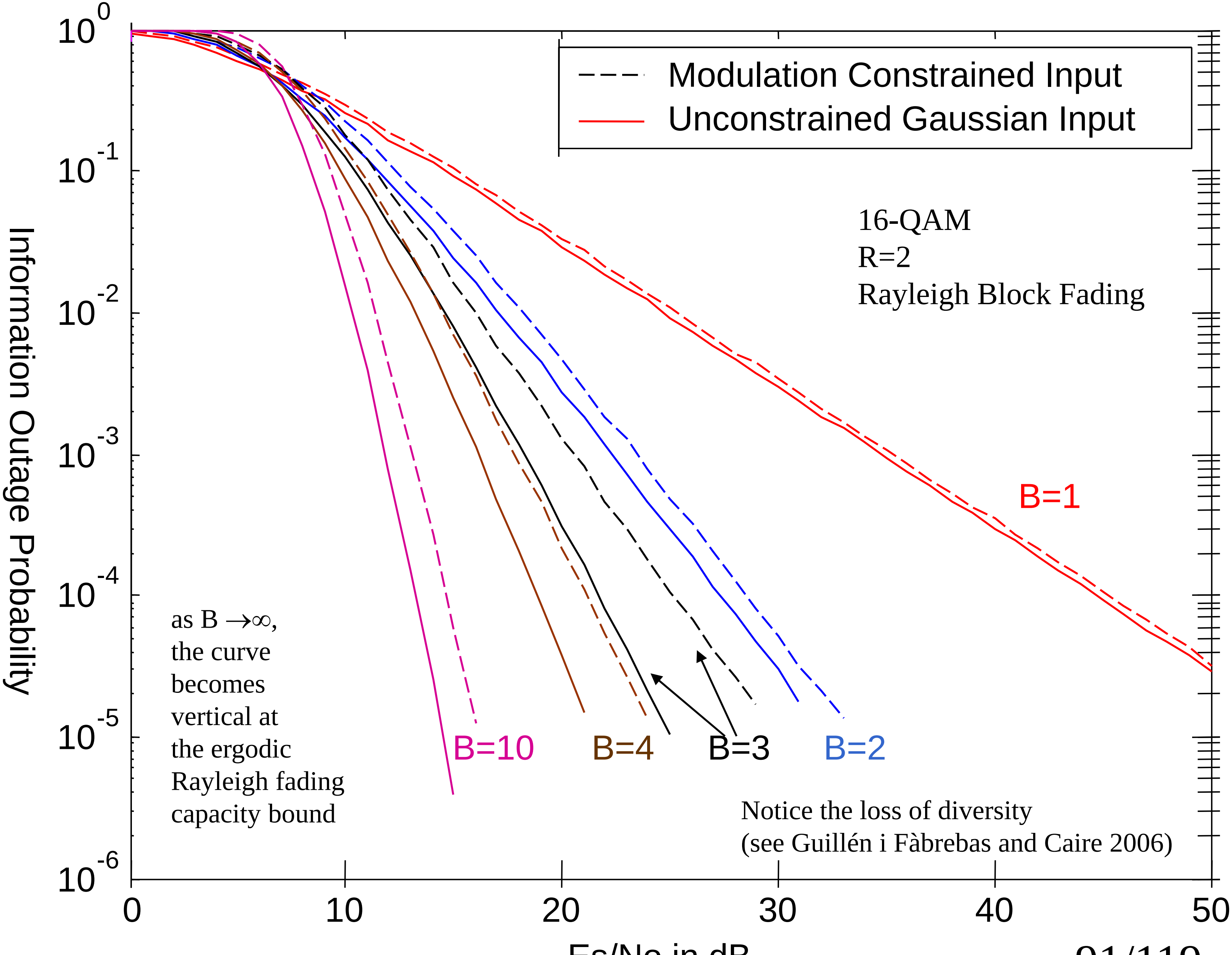
$$I_1^B < R$$

  – where $R \le \log_2 M$ is the rate of the coded modulation

- An outage implies that no code can be reliable for the particular channel instantiation
- The information outage probability is

$$P_0 = P\left[I_1^B < R\right]$$

  – This is a practical bound on FER for the actual system.

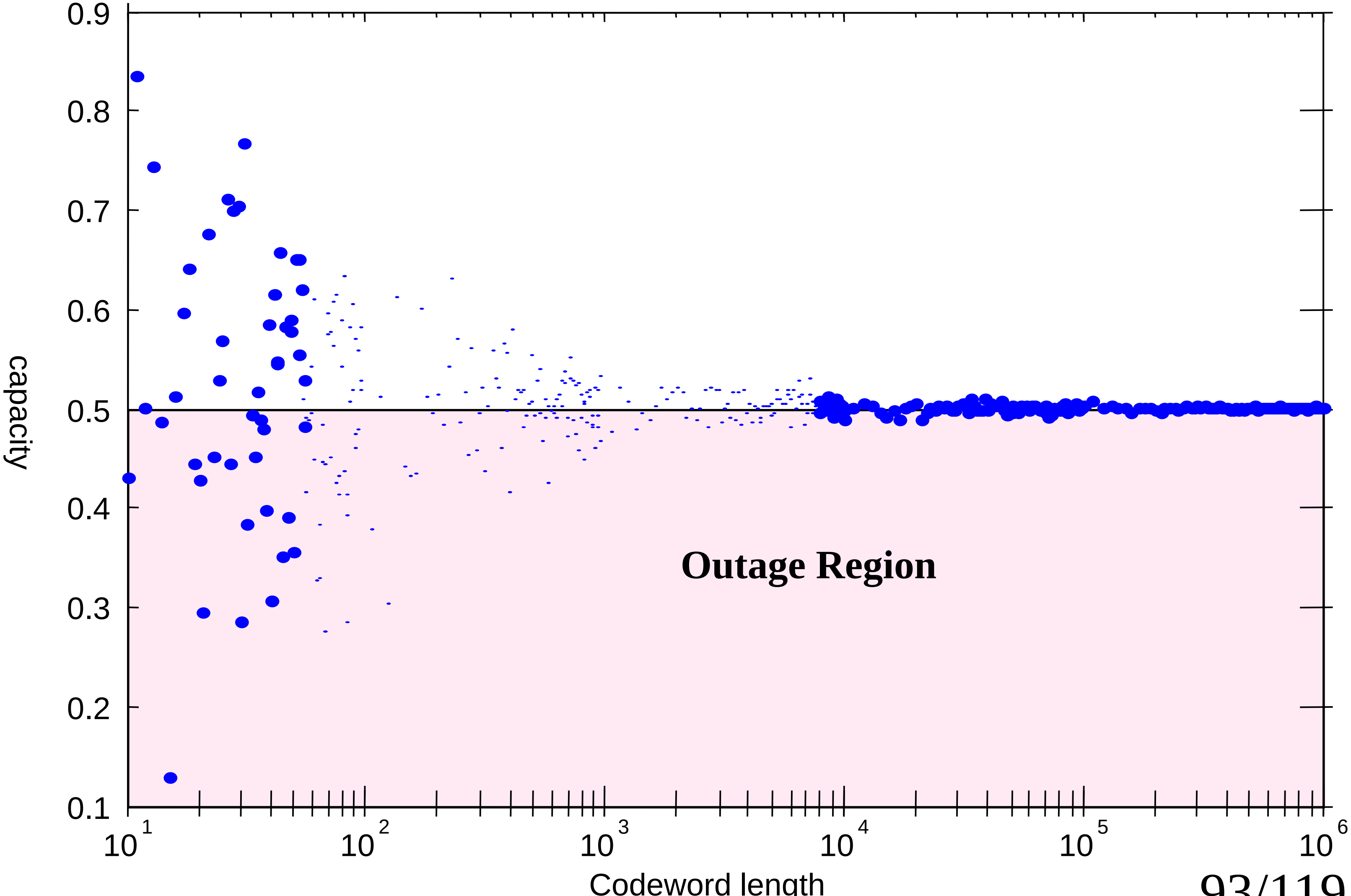


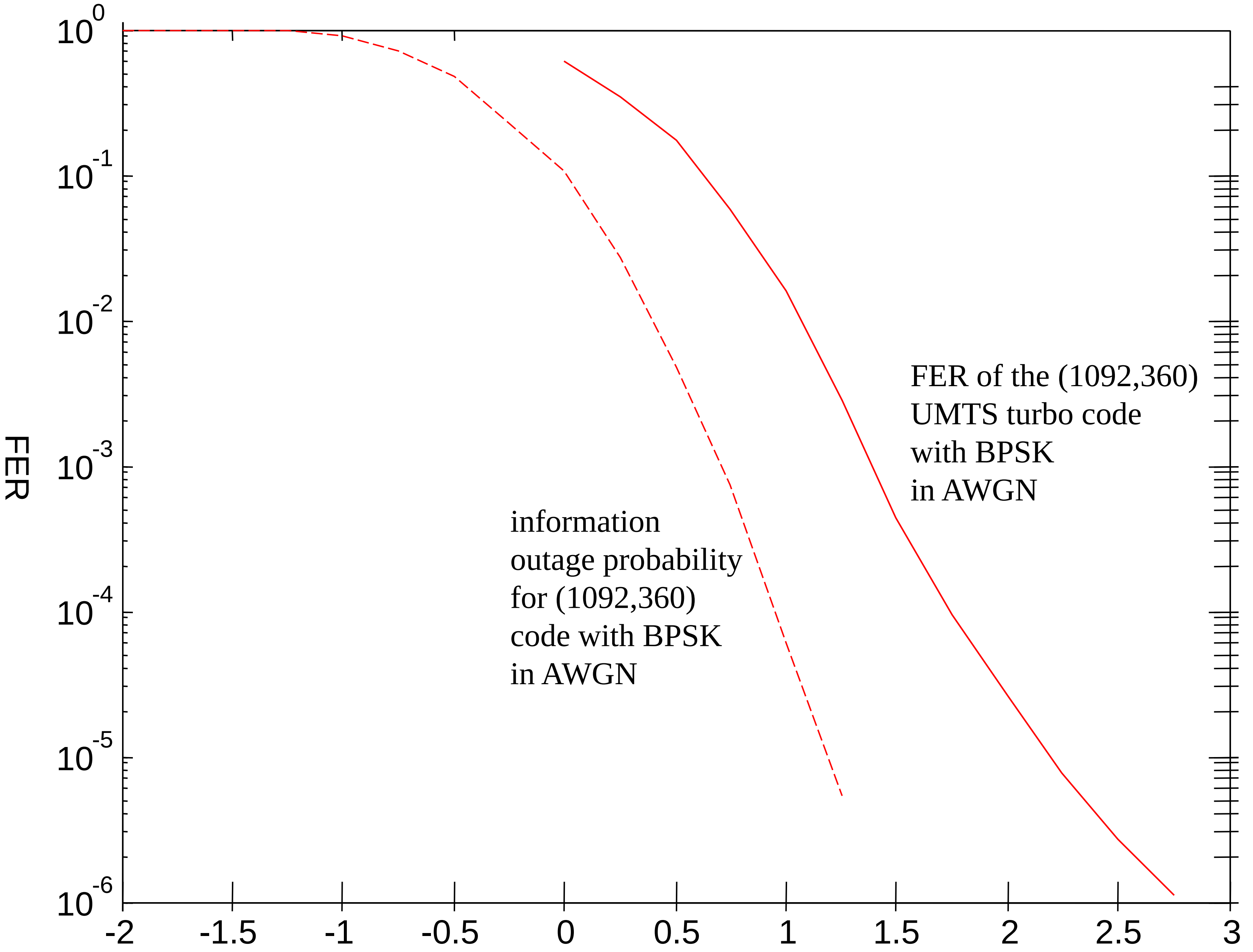
Modulation Constrained Input
Unconstrained Gaussian Input
16-QAM
R=2
Rayleigh Block Fading
B=1
B=10
B=4
B=3
B=2
as B →∞,
the curve
becomes
vertical at
the ergodic
Rayleigh fading
capacity bound
Notice the loss of diversity
(see Guillén i Fàbrebas and Caire 2006)
Information Outage Probability
Es/No in dB
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
0
10
20
30
40
50

Outage Simulation Type

- sim_param(record).
  - blocks_per_frame
    - Assumes block fading channel
  - mod_order
    - 0 for Gaussian input case
  - rate
    - Code rate.
    - Outage whenever MI < rate
  - combining_type = {‘code’, ‘diversity’}
  - input_filename
    - Required if mod_order > 0
    - Contains results of a capacity simulation.
    - Used for a table look-up operation

Finite Length Codeword Effects





FER
Eb/No in dB
information outage probability for (1092,360) code with BPSK in AWGN
FER of the (1092,360) UMTS turbo code with BPSK in AWGN
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5
3

Bloutage Simulation Type

- Set up like an uncoded simulation
  - framesize
  - specify the modulation
    - Set mod_order = 0 for unconstrained Gaussian input
  - specify the channel (AWGN, Rayleigh, etc.)
- Also requires the rate
- Saves FER, not BER

Outline

6. Outage analysis
    - Determine the outage probability over block fading channels
    - Determine the outage probability of finite-length codes

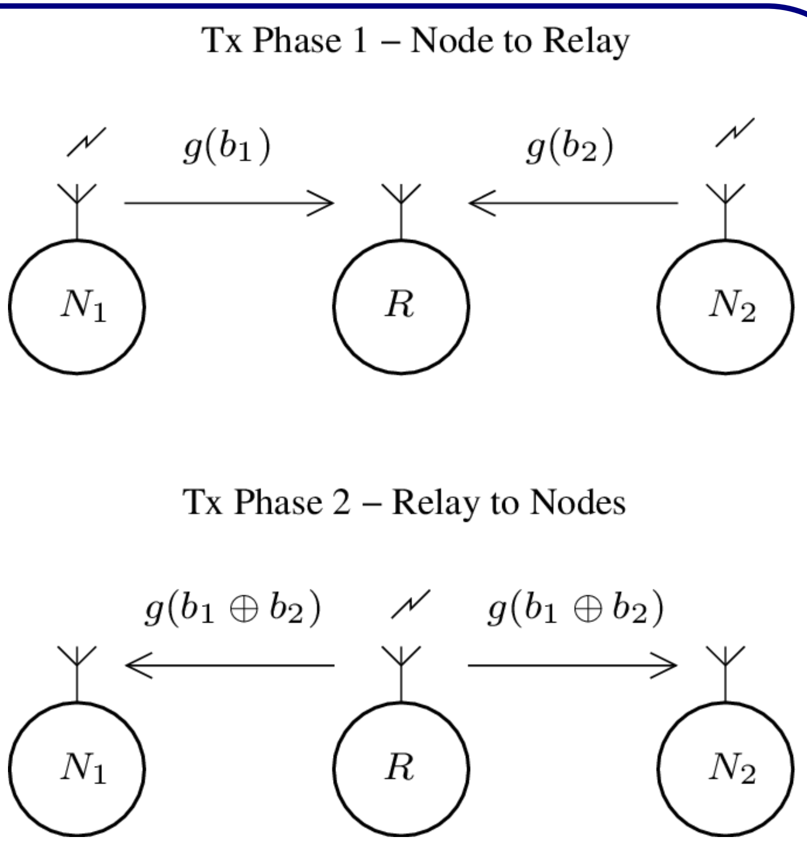
7. Digital Network Coding
    - Describe CML implementation of digital network coding
    - Simulate error rate of digital network coding for non-channel-coded 4-FSK in Rayleigh fading

8. The internals of CML

9. Throughput calculation
    - Convert BLER to throughput for hybrid-ARQ

Digital Network Coding

- Digital network coding (DNC) is a relaying scheme which improves the capacity of multiple-access relay channels.
- Two or more nodes transmit in the same time and band to a relay, deliberately interfering.
- Relay receives interferes signal, encodes and broadcasts such that nodes may decode desired information.



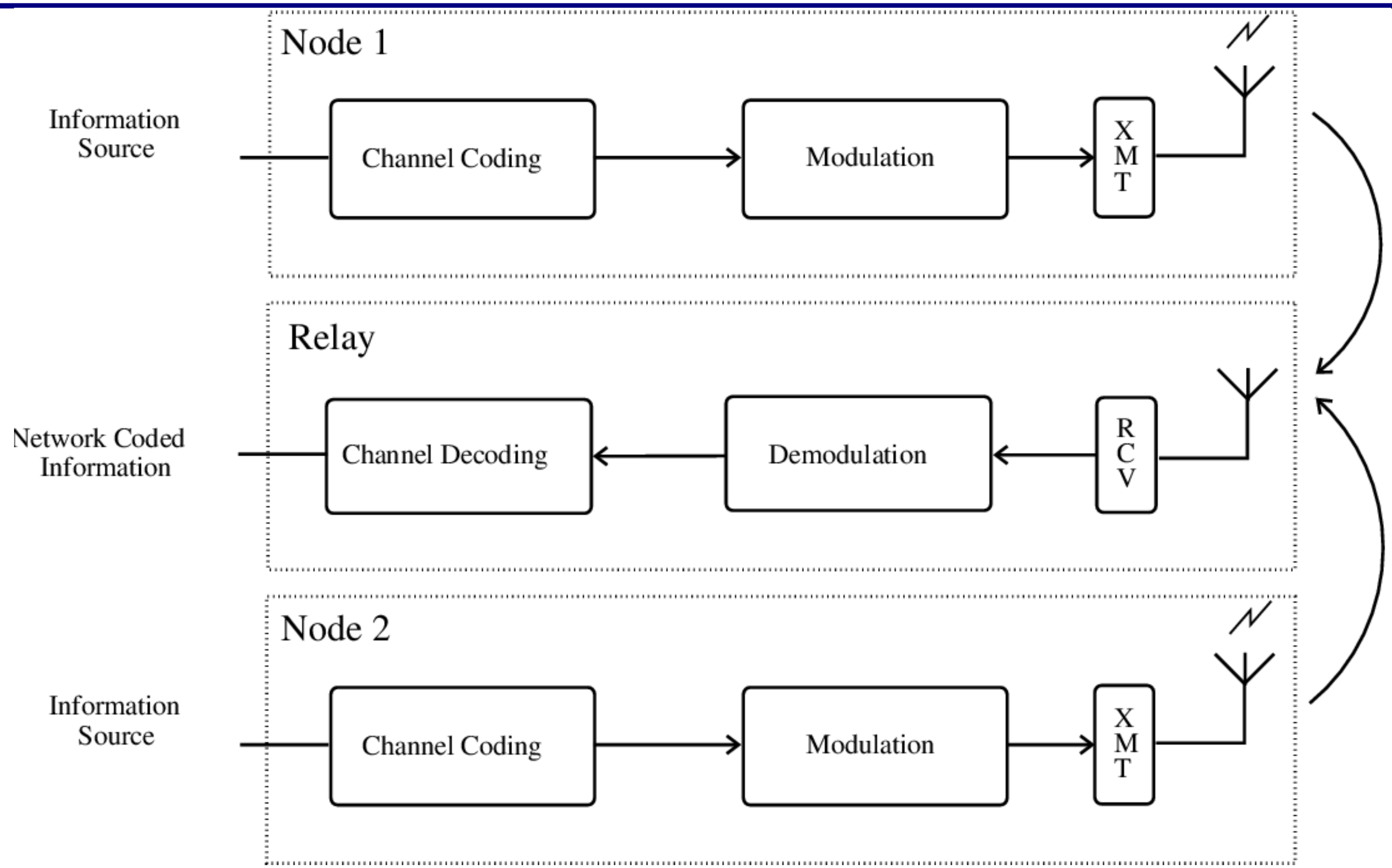
Digital Network Coding Example
Two-way Relay Channel

Digital Network Coding

Technical Requirements

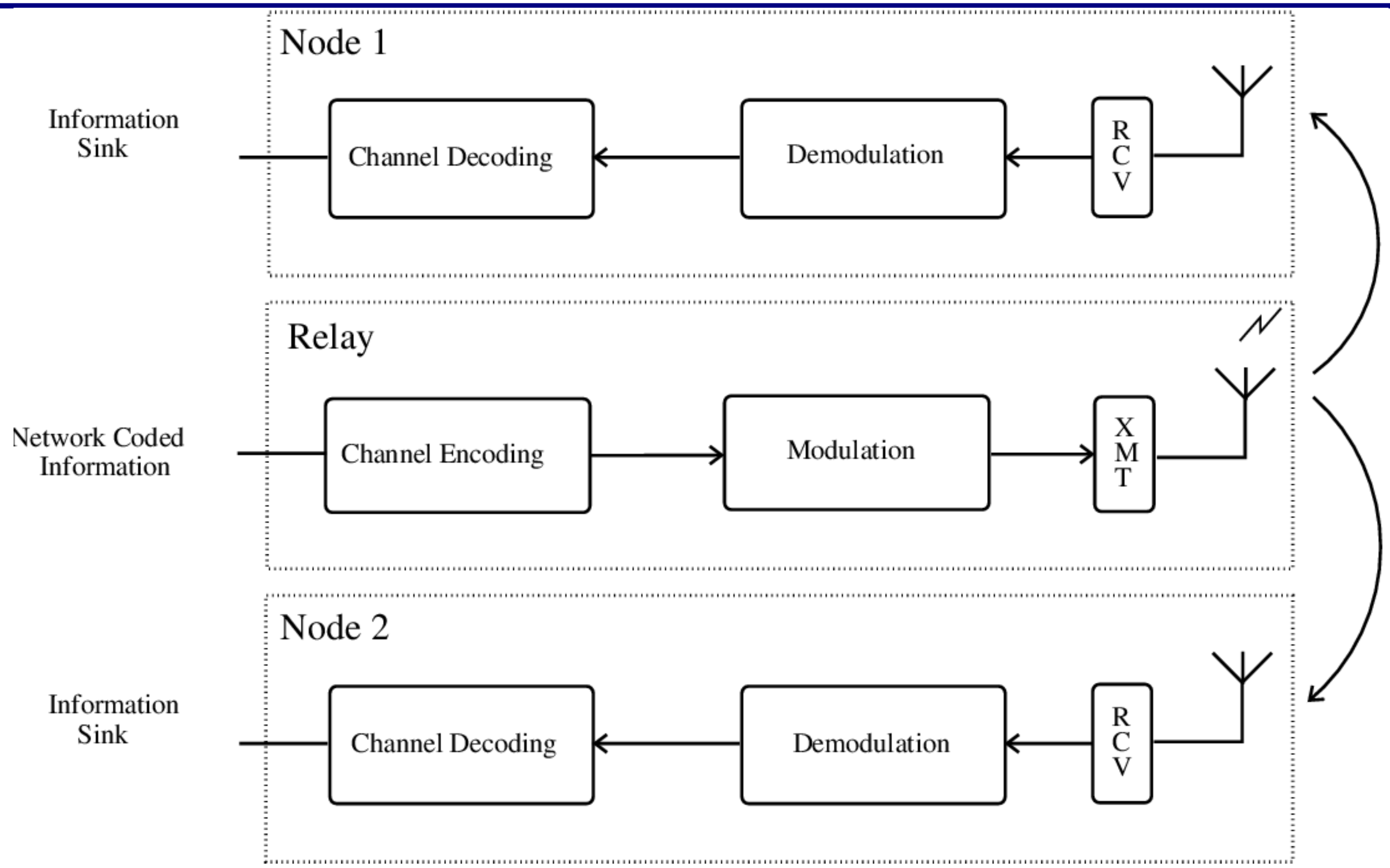
- Network encoding scheme at relay.
- Demodulation scheme at relay which performs network encoding.
- Demodulator producing soft-output channel observations suitable for capacity-approaching iterative decoding.

Digital Network Coding



Tx Phase 1 - Block Diagram

Digital Network Coding



Tx Phase 2 - Block Diagram

Digital Network Coding

CML DNC Implementation [2]

- Topology: Two-way Relay Channel
- Network encoding scheme at relay.
  - Exclusive-or $b = b_1 \oplus b_2$
- Demodulation scheme at relay which performs network encoding.
  - Noncoherent FSK
- Demodulator producing soft-output channel observations suitable for capacity-approaching iterative decoding.
  - Codes supported: Turbo and LDPC

[2] M. C. Valenti, D. Torrieri, and T. Ferrett, "Noncoherent physical-layer network coding using binary CPFSK modulation," Proc. IEEE Military Commun. Conf., Oct. 2009.

Digital Network Coding

CML DNC Performance Simulation

- Compute relay error rate as a function of
  - Modulation order {2, 4, 8, 16 …}
  - Channel Type {Rayleigh, AWGN}
  - Channel State Information {full, partial, none}
  - Channel code {Turbo, LDPC}

DNC Simulation Example

- CML Scenario: DncTwrc

  Record: 1

  - sim_type = 'uncoded'
  - topology = 'twrc'
  - twrc_param.protocol = 'dnc'
  - twrc_param.energy_ratio = 1   % sources use same Tx energy
  - modulation = 'FSK'
  - mod_order = 4
  - channel = 'Rayleigh'
  - csi_flag = 1 % Partial CSI
  - SNR = [0 : 0.5 : 50]

DNC Simulation Example

User Simulation Steps

Start MATLAB and initialize CML

```
>> CmlStartup
```

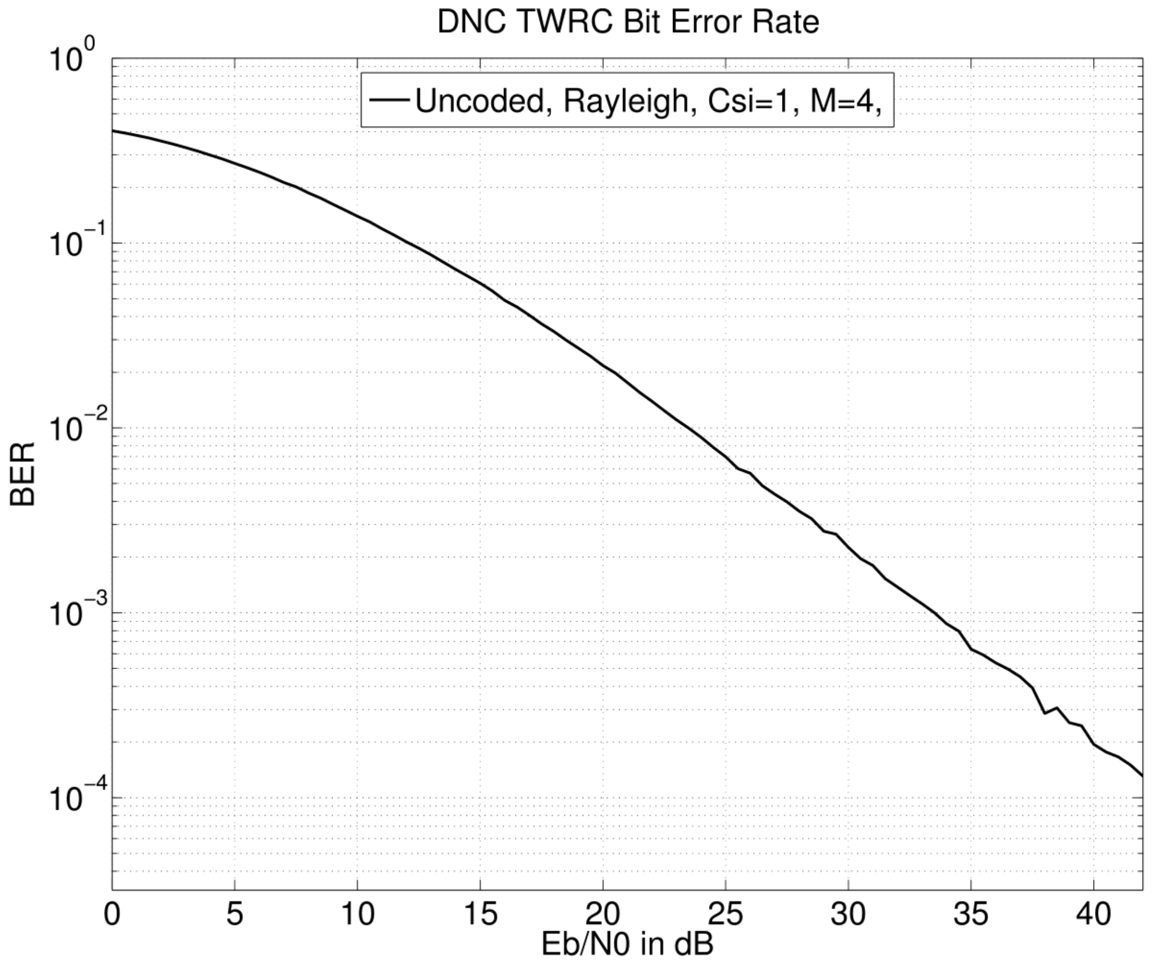
Execute DNC simulation

```
>> CmlSimulate('DncTwrc', 1)
```

Plot results

```
>> CmlPlot('DncTwrc', 1)
```

DNC Simulation Example



Outline

6. Outage analysis
   - Determine the outage probability over block fading channels
   - Determine the outage probability of finite-length codes

7. Digital Network Coding
   - Describe CML implementation of digital network coding
   - Simulate error rate of digital network coding for non-channel-coded 4-FSK in Rayleigh fading

8. The internals of CML

9. Throughput calculation
   - Convert BLER to throughput for hybrid-ARQ

Main Program Flow

- CmlSimulate
  - ReadScenario
    - Runs SingleRead for each record
    - Performs sanity check on sim_param structure
    - Initializes or restores the sim_state structure
  - For each record
    - SingleSimulate if a simulation
    - Otherwise, runs one of the analysis functions:
      - CalculateThroughput
      - CalculateMinSNR
      - CalculateMinSNRvsB

SingleSimulate

- Seeds random number generator
- Branches into
  - SimulateMod
    - For uncoded, coded, and bloutage point-to-point channel
  - SimulateTwrc
    - For uncoded, coded, two-way relay channel
  - SimulateUGI
    - For a blocklength-constrained outage simulation with unconstrained Gaussian input.
  - SimulateCapacity
  - SimulateCapacityTwrc
  - SimulateOutage

SimulateMod

- Main subfunctions (coded/uncoded) cases:
  - CmlEncode
  - CmlChannel
  - CmlDecode
- For bloutage, replace CmlDecode with
  - Somap
  - capacity

SimulateTwrc

- Main subfunctions (coded/uncoded) cases:
  - CmlEncode
  - CmlTwrcRelayChannel
  - CmlTwrcRelayComputeSymbolLh
  - CmlInitSomap
  - CmlTwrcRelaySomap
  - CmlTwrcRelayDecode

SimulateCapacity

- Operates like SimulateMod with sim_type = ‘bloutage’
  - However, instead of comparing MI of each codeword against the rate, keeps a running average of MI.

- In case of exit simulation, computes detector and decoder mutual information characteristics

SimulateCapacityTwrc

- Similar to SimulateTwrc but computes mutual information at output of detector and Somap rather than error rates
- In case of exit simulation, computes detector and decoder mutual information characteristic

SimulateOutage

- Randomly generates SNR for each block
- Performs table lookup to get MI from SNR
- Compares MI against threshold

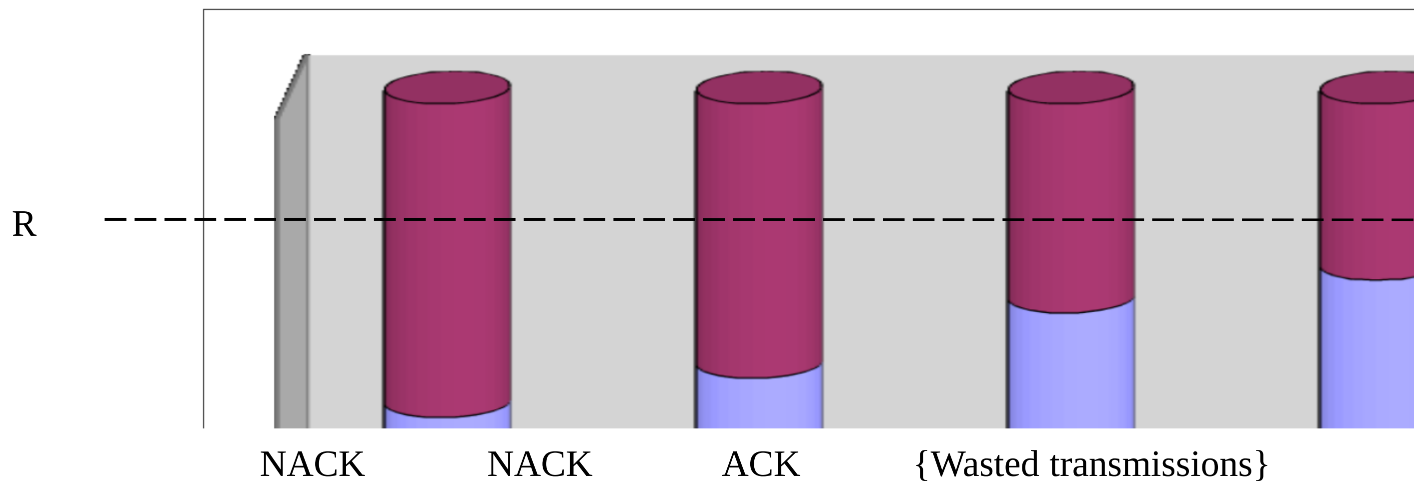
Outline

1. CML overview
   - What is it? How to set it up and get started?
2. Uncoded modulation
   - Simulate uncoded BPSK and QAM in AWGN and Rayleigh fading
3. Coded modulation
   - Simulate a turbo code from UMTS 25.212
4. Ergodic (Shannon) capacity analysis
   - Determine the modulation constrained capacity of BPSK and QAM
5. Outage analysis
   - Determine the outage probability over block fading channels.
   - Determine the outage probability of finite-length codes
6. The internals of CML
7. Throughput calculation
   - Convert BLER to throughput for hybrid-ARQ

Hybrid-ARQ
(Caire and Tunnineti 2001)

- Once $I_1^B > R$ the codeword can be decoded with high reliability.
- Therefore, why continue to transmit any more blocks?
- With hybrid-ARQ, the idea is to request retransmissions until $I_1^B > R$
  - With hybrid-ARQ, outages can be avoided.
  - The issue then becomes one of latency and throughput.

b=1 $I_1 = \log(1+\gamma_1)$	b=2 I_2	b=3 I_3	b=4 I_4	b=5 I_5



Outline

6. Outage analysis
   - Determine the outage probability over block fading channels
   - Determine the outage probability of finite-length codes

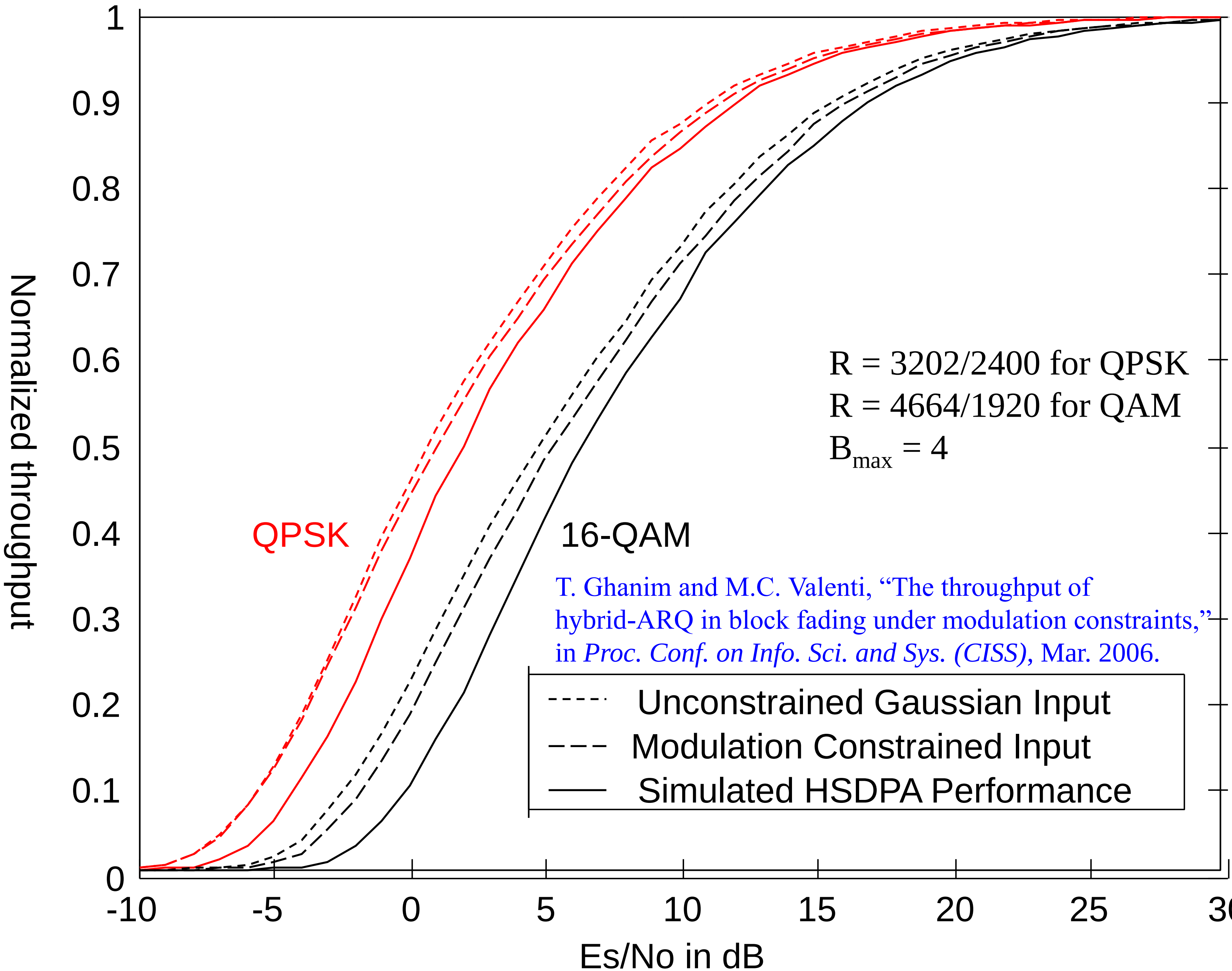
7. Digital Network Coding
   - Describe CML implementation of digital network coding
   - Simulate error rate of digital network coding for non-channel-coded 4-FSK in Rayleigh fading

8. The internals of CML

9. Throughput calculation
   - Convert BLER to throughput for hybrid-ARQ

Latency and Throughput of Hybrid-ARQ

- With hybrid-ARQ B is now a random variable.
  - The average ***latency*** is proportional to E[B].
  - The average ***throughput*** is inversely proportional to E[B].
- Often, there is a practical upper limit on B
  - Rateless coding (e.g. Raptor codes) can allow $B_{max} \rightarrow \infty$
- An example
  - HSDPA: High-speed downlink packet access
  - 16-QAM and QPSK modulation
  - UMTS turbo code
  - HSET-1/2/3 from TS 25.101
  - $B_{max} = 4$



Normalized throughput
Es/No in dB
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
-10
-5
0
5
10
15
20
25
30
QPSK
16-QAM
R = 3202/2400 for QPSK
R = 4664/1920 for QAM
B_{max} = 4
T. Ghanim and M.C. Valenti, "The throughput of hybrid-ARQ in block fading under modulation constraints," in *Proc. Conf. on Info. Sci. and Sys. (CISS)*, Mar. 2006.
Unconstrained Gaussian Input
Modulation Constrained Input
Simulated HSDPA Performance

Conclusions: Design Flow with CML

- When designing a system, first determine its capacity.
  - Only requires a slight modification of the modulation simulation.
  - Does not require the code to be simulated.
  - Allows for optimization with respect to free parameters.
- After optimizing with respect to capacity, design the code.
  - BICM with a good off-the-shelf code.
  - Optimize code with respect to the EXIT curve of the modulation.
- Information outage analysis can be used to characterize:
  - Performance in slow fading channels.
  - Delay and throughput of hybrid-ARQ retransmission protocols.
  - Finite codeword lengths.