# DECODING WITHOUT A DECODER

James Wang, Prof. A. Brinton Cooper III
Department of Electrical and Computer Engineering
Johns Hopkins University

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

## Introduction

### Motivation for GRAND

Digital communications use an error correcting code to recover data transmitted through a noisy channel that causes bit errors. But decoders can require excessive time to decode the data. The recently proposed Guessing Random Additive Noise Decoding(GRAND) algorithm "guesses" the binary noise vector and, after a few attempts, usually produces error-free data. What is exciting about GRAND is that it can "decode" any code. Of people's special interest are channels that cause errors to come in bursts.

### How GRAND works

Inputs: $Y^n$, the observed received vector
Outputs: $\hat{X}$, $Q$

$d \leftarrow 0$, $Q \leftarrow 0$.
while $d = 0$ do
    $z^n \leftarrow$ the next most likely noise vector
    $Q \leftarrow Q + 1$
    if $Y^n \ominus z^n$ is a code-word then
        $\hat{X} \leftarrow Y^n \ominus z^n$
        $d \leftarrow 1$
        return $\hat{X}$, $Q$
    end if
end while

$$Y^n = X^n \oplus N^n$$

Traditional decoder:
identify $X^n$ using the structure of code-book.

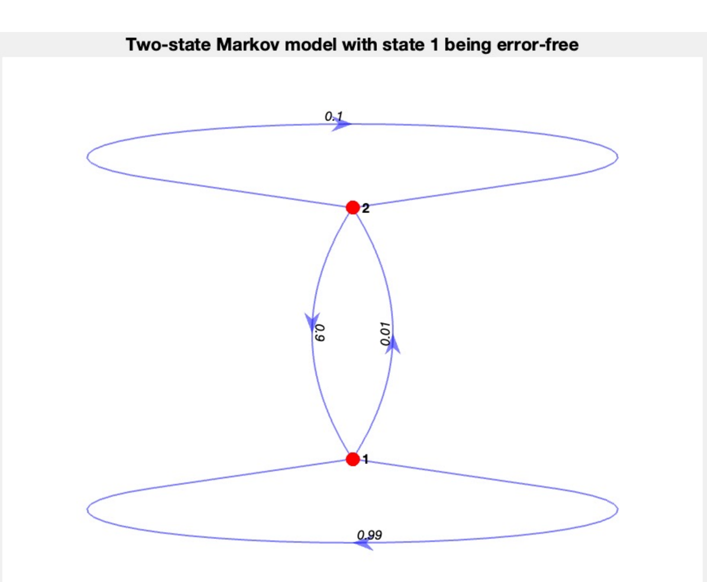
GRAND:
identify $N^n$ by exploiting structure of channel noise.

### Two conundrums of GRAND that our work hopes to solve:

- Model the channel noise in a way that stays true to the bursty nature of a physical channel.
- Order all noise patterns from the most likely to the least likely.

## Solutions

### Modeling the bursty channel: Two-state Markov Chain




When in the good state, G, the channel is error-free and the corresponding entry in $N^n$ is to a 0, and when the channel is in the bad state, B, there is an error and the corresponding entry in $N^n$ is to a 1.

The transition probability from G to B is b and from B to G is g. An error burst is a set of consecutive 1s in $N^n$ with its length following geometry distribution of mean $1/g$ and variance $(1-g)/g^2$.

**Constraint**
Consecutive bursts must be separated by at least one 0.

**Notations**
l: # 1's in the noise vector.
m: # bursts of 1's in the noise vector.

### Approach 1: Using two-state Markov model to find the statistics

Set the transition probability → Set up the Markov Model → Generate a random sequence of bits from the model → Divide the generated sequence into n-length words → Count the frequencies of each n-length word in the sequence → Order them from the highest to the lowest occurrence count → Sorted noise patterns

### Approach 2: Calculate probability and apply integer partitioning

Calculate the probability of different classes of n-length bit sequence → Arrange all the l and m pairs from the highest to the lowest probability → For each l and m pair, generate all the length-n noise patterns using integer partitioning → Sorted noise patterns

- The algorithm to generate all permutations for a given l and m pair

Generate all integer partitioning results for this pair. This gives us all the possible burst patterns. → Based on the basic patterns from last step, generate all the possible permutations, i.e., to permute the positions of the bursts. → For the zeros to be inserted between the bursts, also make use of integer partitioning, since the number of zeros and insertions are known. → Do the generation procedure like the last step for the 0's. → Assemble the 1's and the 0's to obtain our final noise vectors.

- Illustration for the algorithm

Goal: To obtain all possible noise vectors with (n, l, m) = (15, 4, 8).

Step 1: e.g., One integer partitioning result for the 1's is: _111__11_1__11__

Step 2: e.g., One integer partitioning result for the 0's is: ____00__0_00__00

Step 3: Assemble the two parts: _111__11_1__11__ ____00__0_00__00

Results: 111001101001100 ⋮

### Approach 3: Analytical calculation of two-state Markov probability

$$1^{\text{st}} \text{ symbol} = \begin{cases} 1, & p = \frac{b}{b+g} \\ 0, & p = \frac{g}{b+g} \end{cases}$$

↓

$$\text{Current state} = \text{G, the next symbol} = \begin{cases} 1, & p = b \\ 0, & p = 1-b \end{cases}$$

↓

$$\text{Current stage} = \text{B, the next symbol} = \begin{cases} 1, & p = 1-g \\ 0, & p = g \end{cases}$$

→ And so on⋯

When large length n, calculating every possible sequence using this approach would be intractable.

However, this approach is perfect for generating a "true" noise vector for the GRAND simulation, as we will see in the next section.

## Results

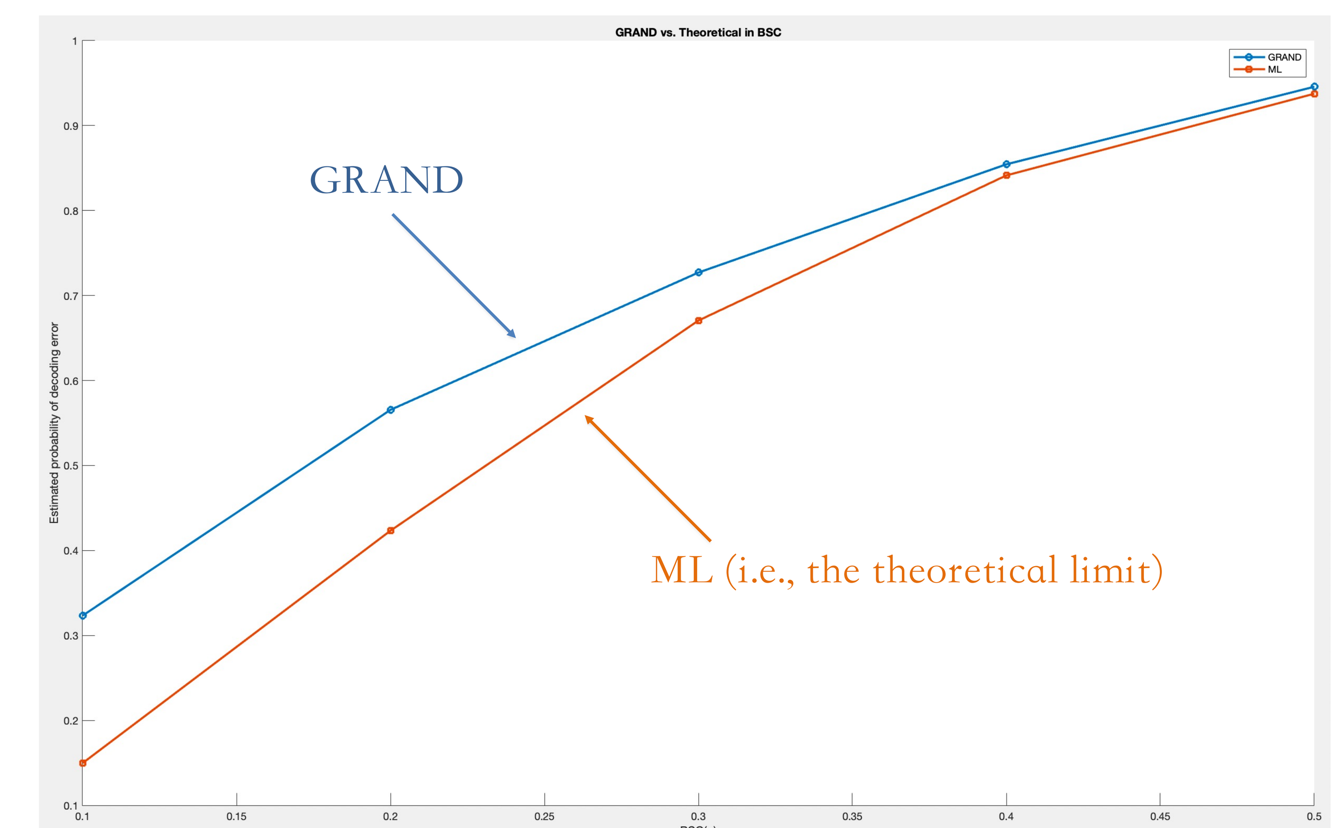
### Simulation Algorithm

1. Generate a random message
2. Encode the message using any code-book
3. Generate the sorted noise patterns matrix (Using Approach 1 or 2)
4. Generate the Markov channel noise (Using Approach 3)
5. Add the Markov channel noise to the transmitted bits
6. Decode the received code-word using the GRAND algorithm
7. Compare the decoded code-word with the actual code-word sent

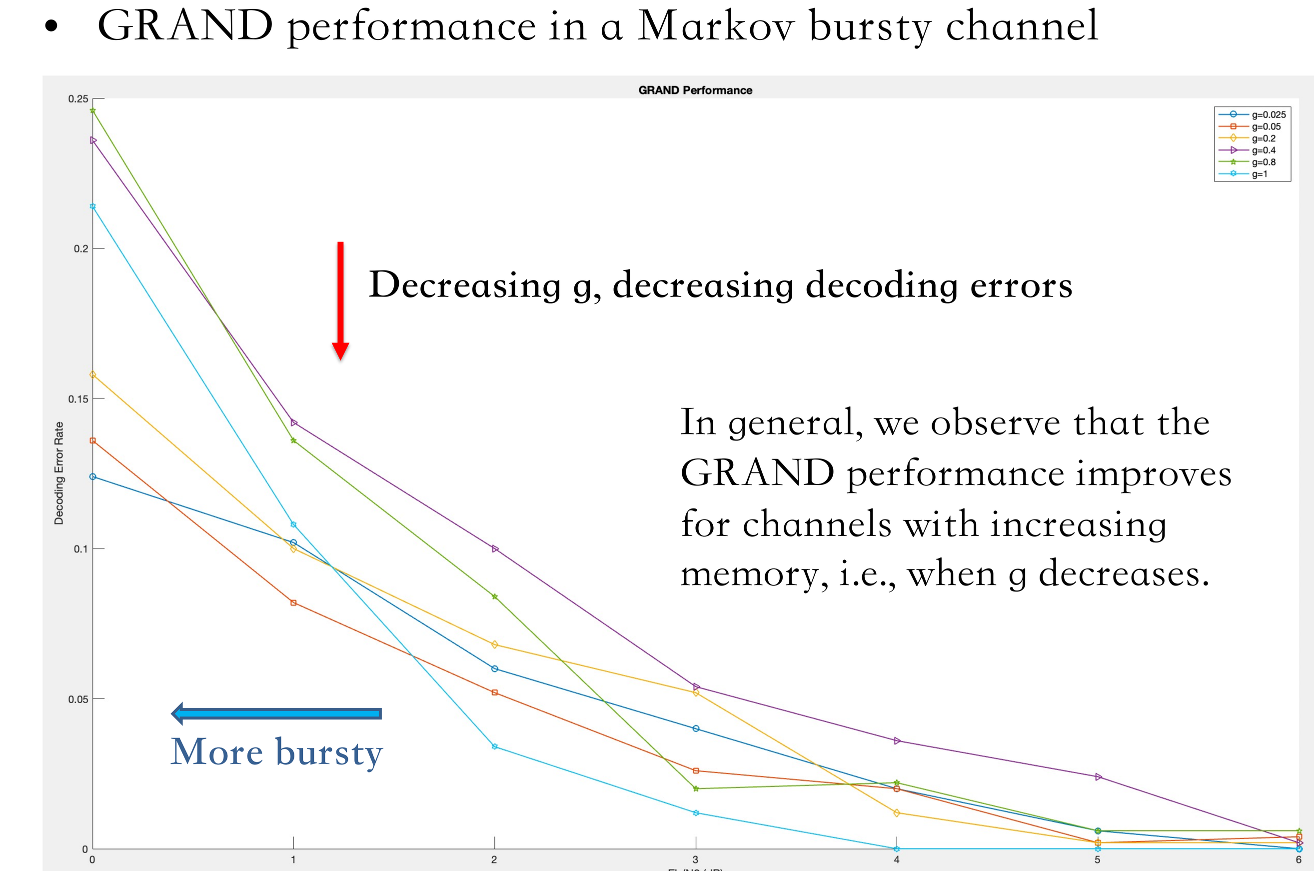
Key steps: Generate the sorted noise patterns matrix; Generate the Markov channel noise

### Simulation Result

- GRAND vs. Theoretical (ML) in an idealized BSC channel




- GRAND performance in a Markov bursty channel




## Reference

Wei An, Muriel Médard, Ken R. Duffy. "Keep the bursts and ditch the interleavers". IEEE Global Communications Conference, 2020.