15-750:Algorithms in the Real World

Data Compression:

Linear Transform Coding

(for both lossless and lossy compression)

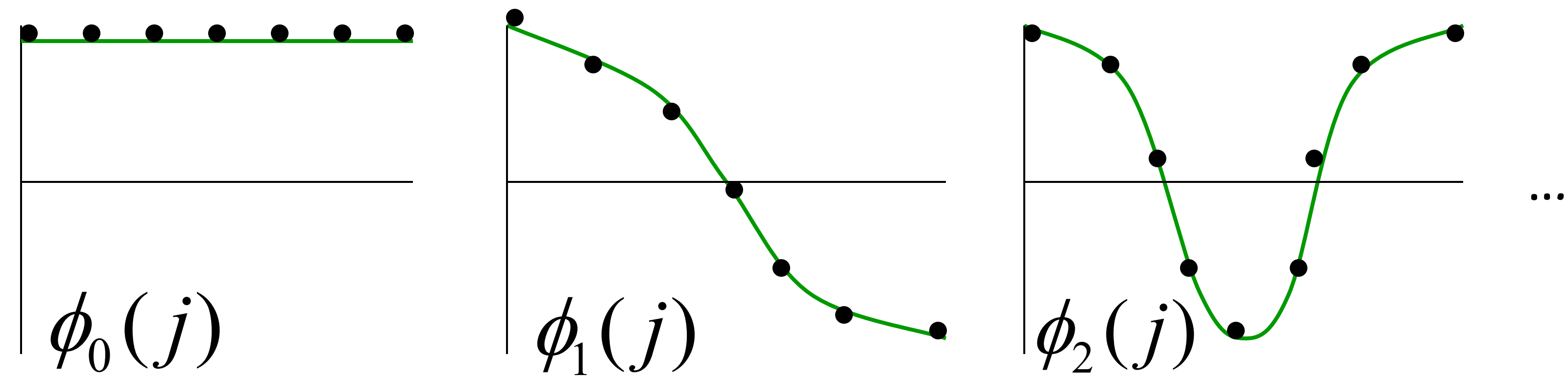
5. Linear Transform Coding

Goal: Transform the data into a form that is easily compressible (through **lossless** or **lossy** compression)

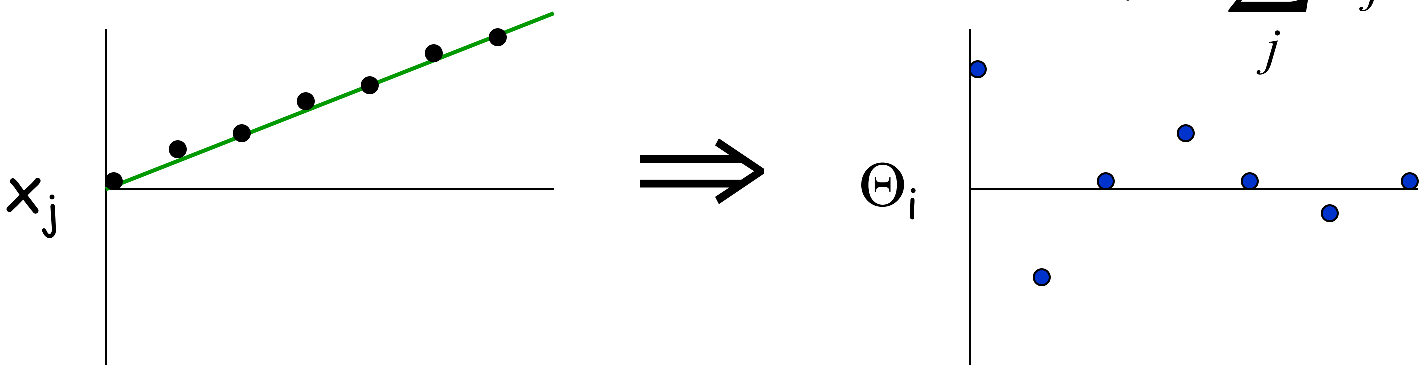
Select a set of linear basis functions ϕ_i that span the space

- sin, cos, spherical harmonics, wavelets, …

Example: Cosine Transform

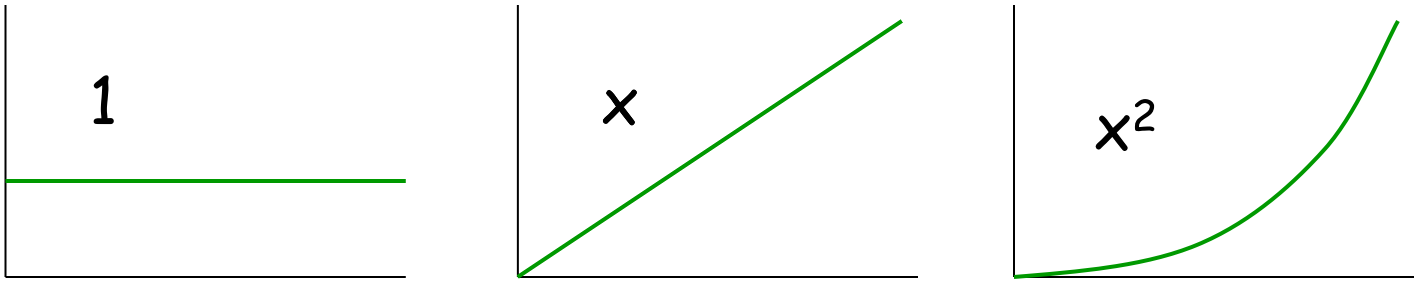


$$\Theta_i = \sum_j x_j \phi_i(j)$$

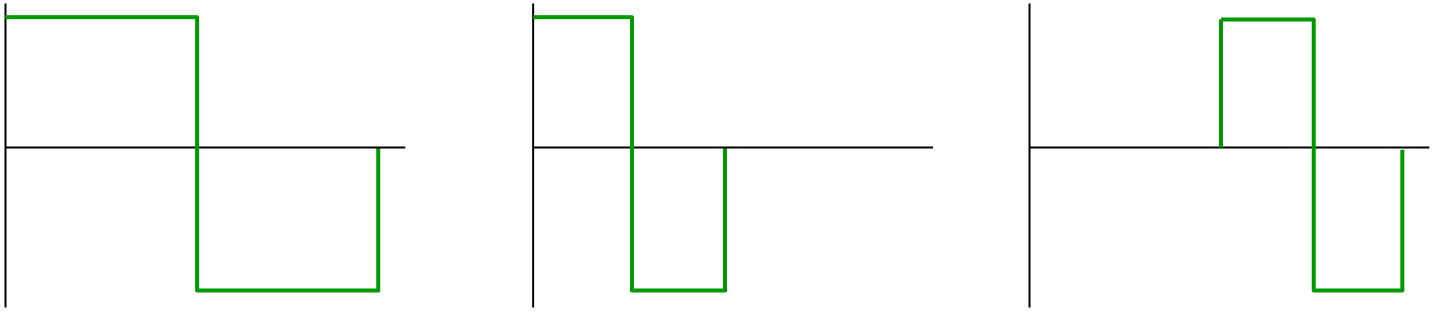


Other Transforms

Polynomial:



Wavelet (Haar):



How to Pick a Transform

Goals:

- Decorrelate the data
- Low coefficients for many terms
- Basis functions that can be ignored from the perception point-of-view

15-750:Algorithms in the Real World

Quantization (lossy)

Scalar Quantization

Quantize regions of values into a single value

E.g. Drop least significant bit

(Can be used to reduce # of bits for a pixel)

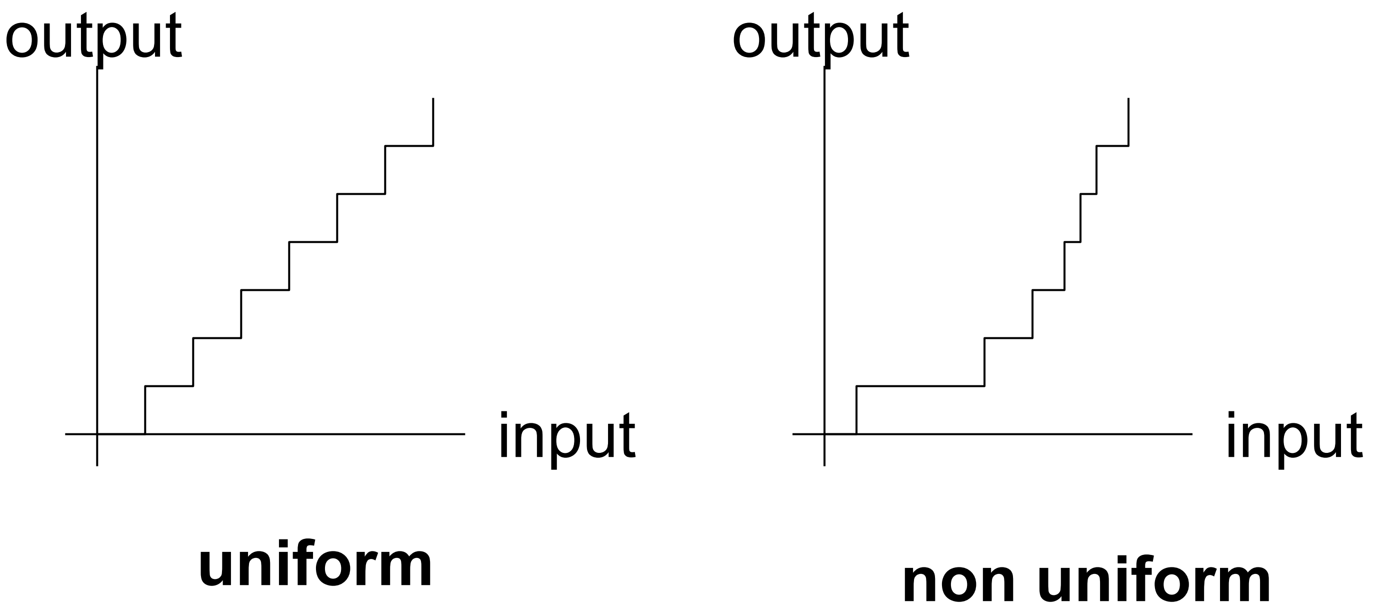
Q: Why is this lossy?

Many-to-one mapping

Two types

- Uniform: Mapping is linear
- Non-uniform: Mapping is non-linear

Scalar Quantization

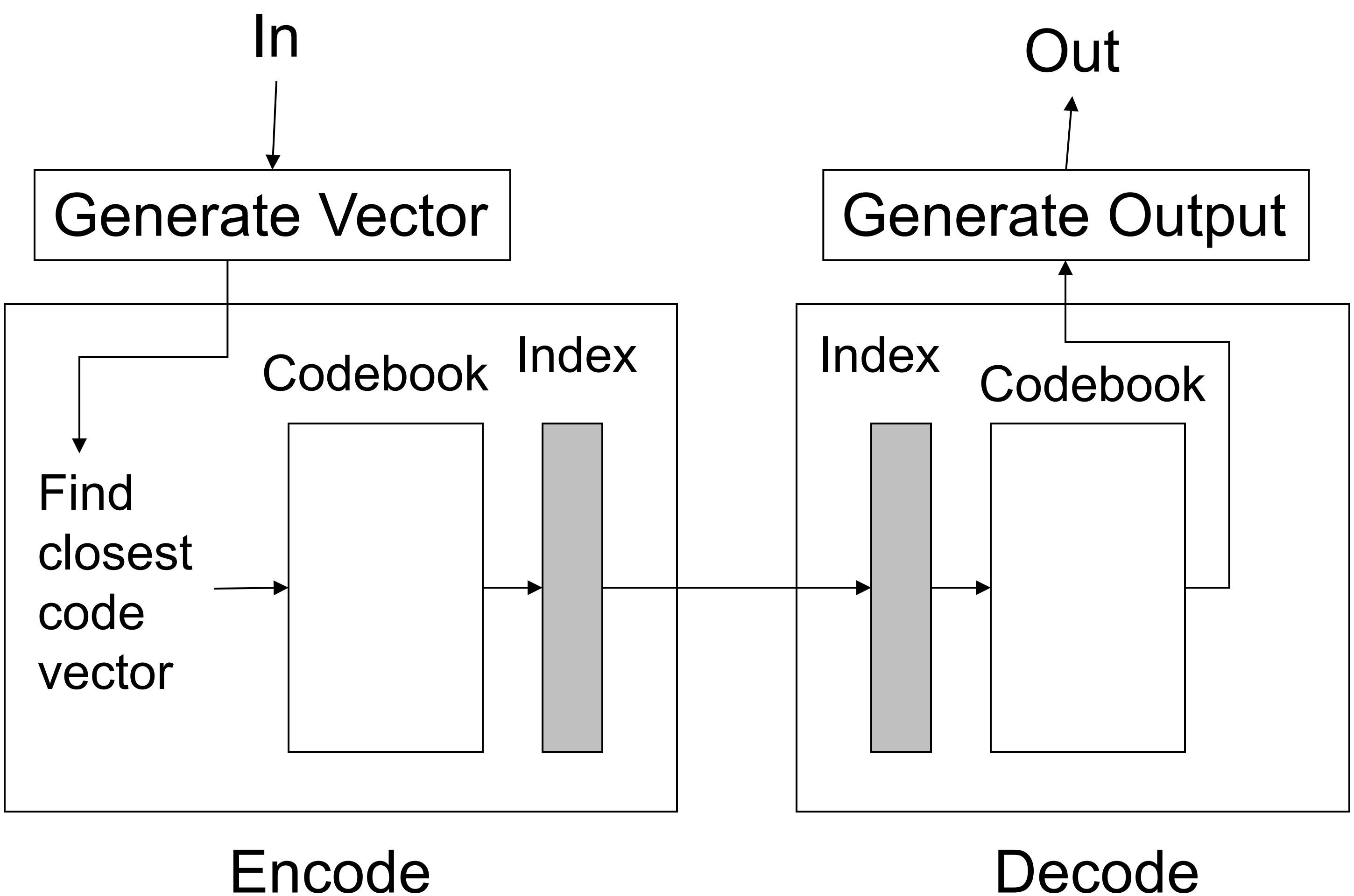


Q: Why use non-uniform?
Error metric might be non-uniform.
E.g. Human eye sensitivity to specific color regions

Can formalize the mapping problem as an optimization problem

Vector Quantization

Mapping a multi-dimensional space into a smaller set of messages



Vector Quantization (VQ)

What do we use as vectors?

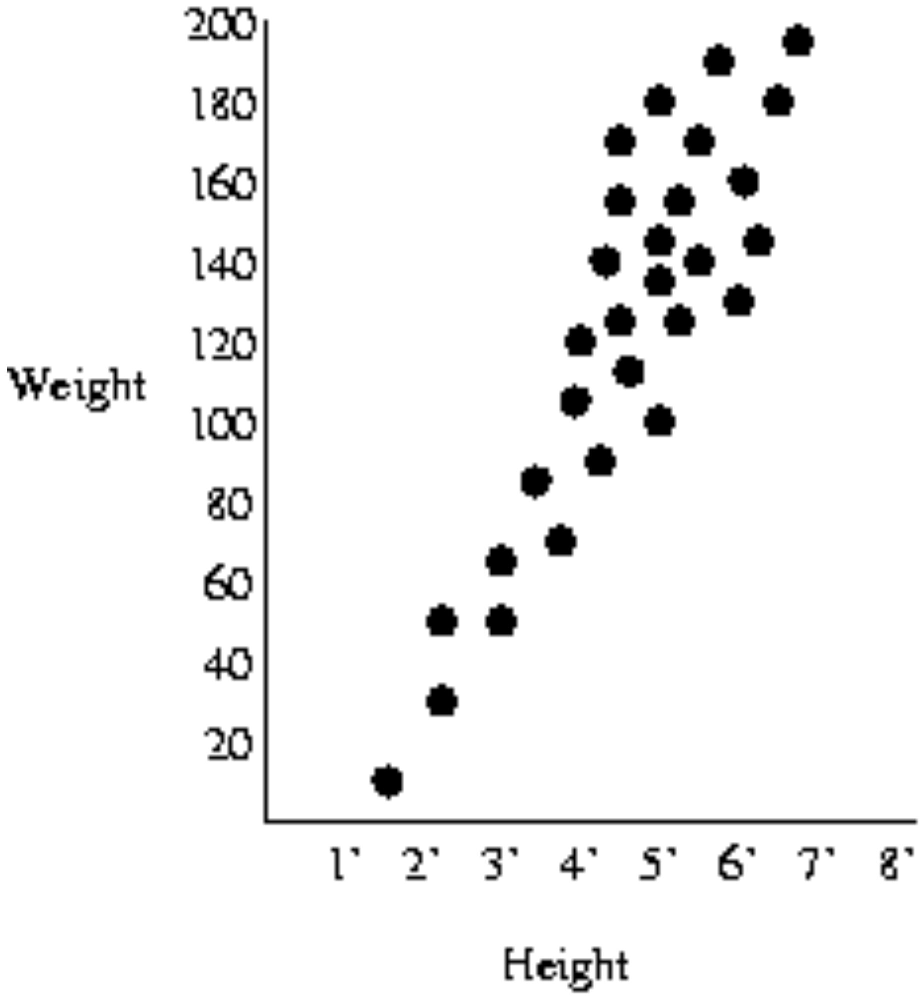
- Color (Red, Green, Blue)
  - Can be used, for example to reduce 24bits/pixel to 8bits/pixel
  - Used in some monitors to reduce data rate from the CPU (colormaps)
- K consecutive samples in audio
- Block of K pixels in an image

How do we decide on a codebook

- Typically done with **clustering**

VQ most effective when the variables along the dimensions of the space are correlated

Vector Quantization: Example



Observations:

1. Highly correlated:
   Concentration of representative points

2. Higher density is more common regions.

Case Study: JPEG

A nice example since it uses many techniques:

- Transform coding (Cosine transform)
- Scalar quantization
- Residual coding
- Run-length coding
- Huffman or arithmetic coding

15-750: Algorithms in the Real World

Algorithms for coding

(Error Correcting Codes)

Welc**e t* t*is clas* o* c*d*ng .
Y*u a** in f*r a f*n rid*!

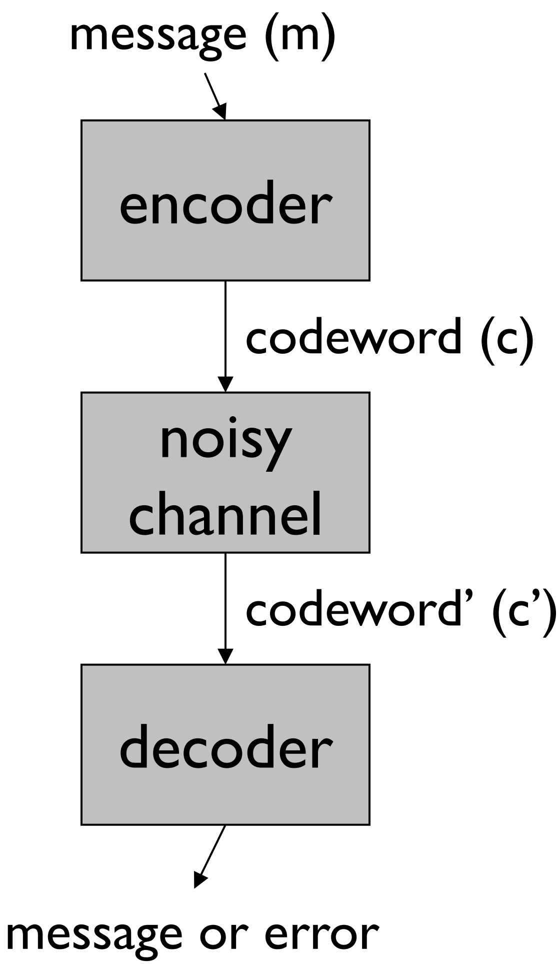
What do these sentences say?

Why did this work?

Redundancy!

Codes are clever ways of judiciously adding redundancy to enable recovery under “noise”.

General Model



"Noise" introduced by the channel:

- changed fields in the codeword vector (e.g. a flipped bit).
  - Called **errors**
- missing fields in the codeword vector (e.g. a lost byte).
  - Called **erasures**

How the decoder deals with errors and/or erasures?

- **detection** (only needed for errors)
- **correction**

Applications

Numerous applications:

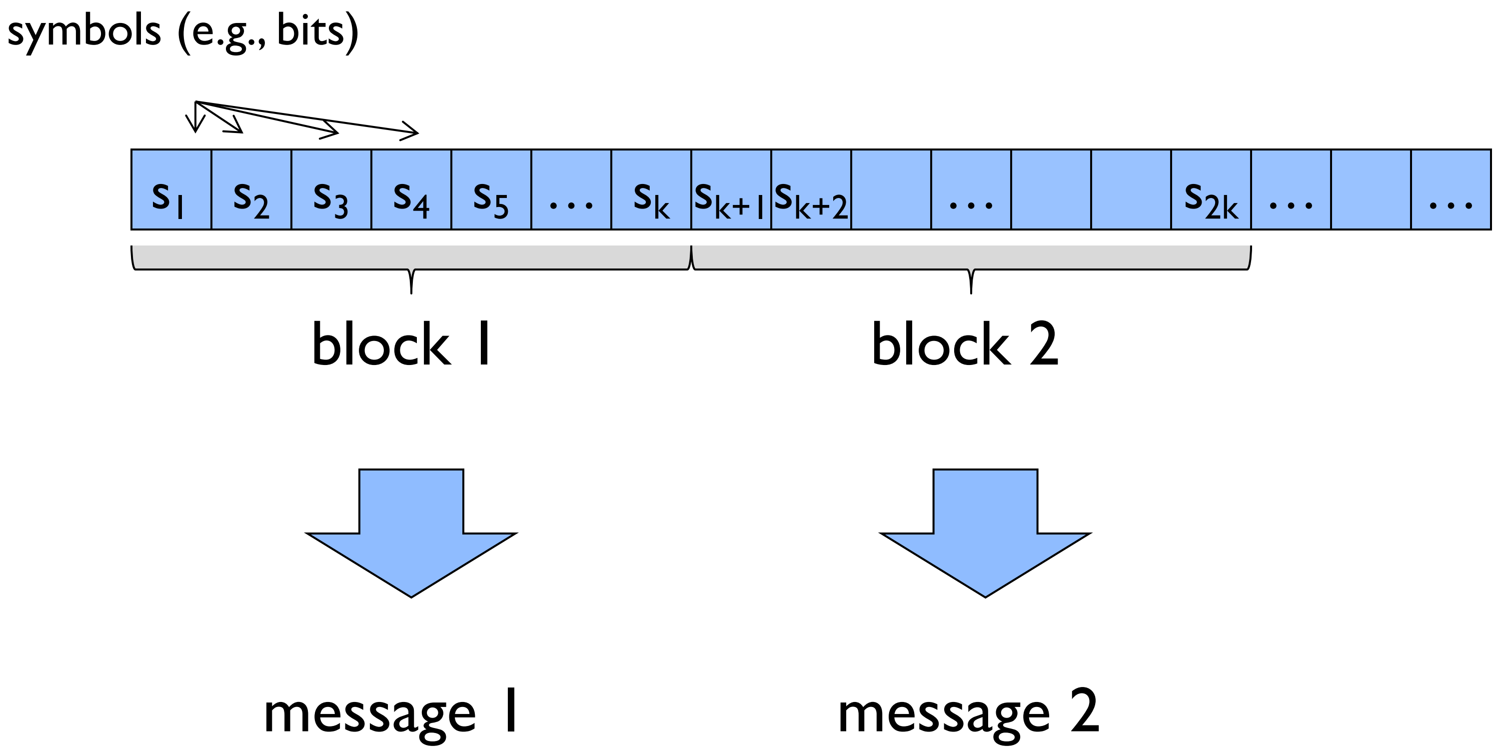
Some examples

- **Storage**: Hard disks, cloud storage, NAND flash…
- **Wireless**: Cell phones, wireless links,
- **Satellite and Space**: TV, Mars rover, …

Reed-Solomon codes are by far the most used in practice.

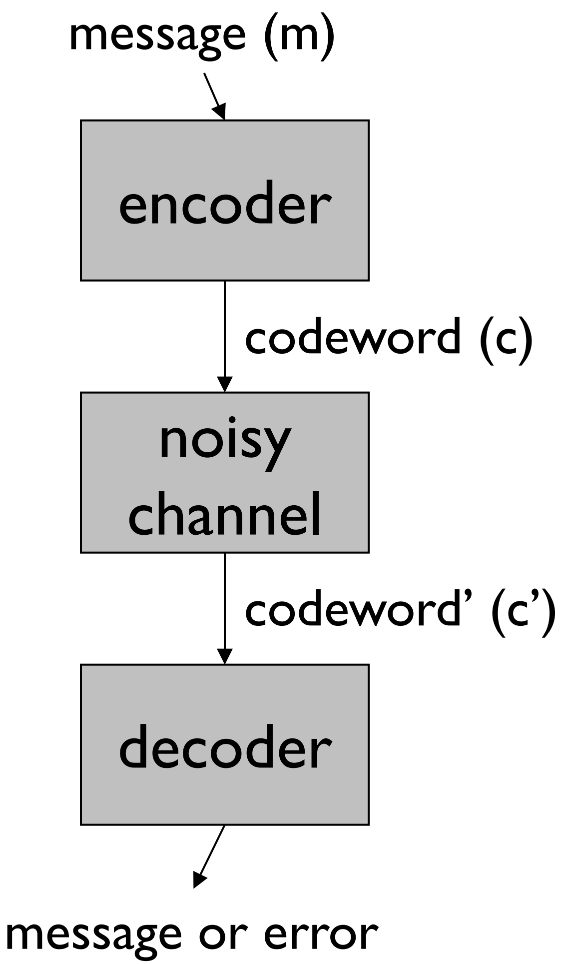
Low density parity check codes (LDPC) codes used for 4G (and 5G) communication and NAND flash

Block Codes



Other kind: convolutional codes (we won't cover it)…

Block Codes



- Each message and codeword is of fixed size
- Notation:

k = |m| ← "dimension of the code"

length of the message

n = |c| ← "length of the code"

length of the codeword

C = "code" = set of codewords

Simple Examples

3-Repetition code: k=1, n=3

Message		Codeword
0	->	000
1	->	111

- How many **erasures** can be recovered?
- How many **errors** can be **detected**?
- Up to how many **errors** can be **corrected**?

Errors are much harder to deal with than erasures.

Why?