

Soft Decision Channel Optimized Vector Quantization

Introduction

- Our goal is to improve multimedia data transmission over noisy channels
- For example, sending digital photos between cell phones
 - This is a low power and low bandwidth application

Last Year

- Extending work done previously by Andrew MacDonald, Sean Noble, and Dave Richardson

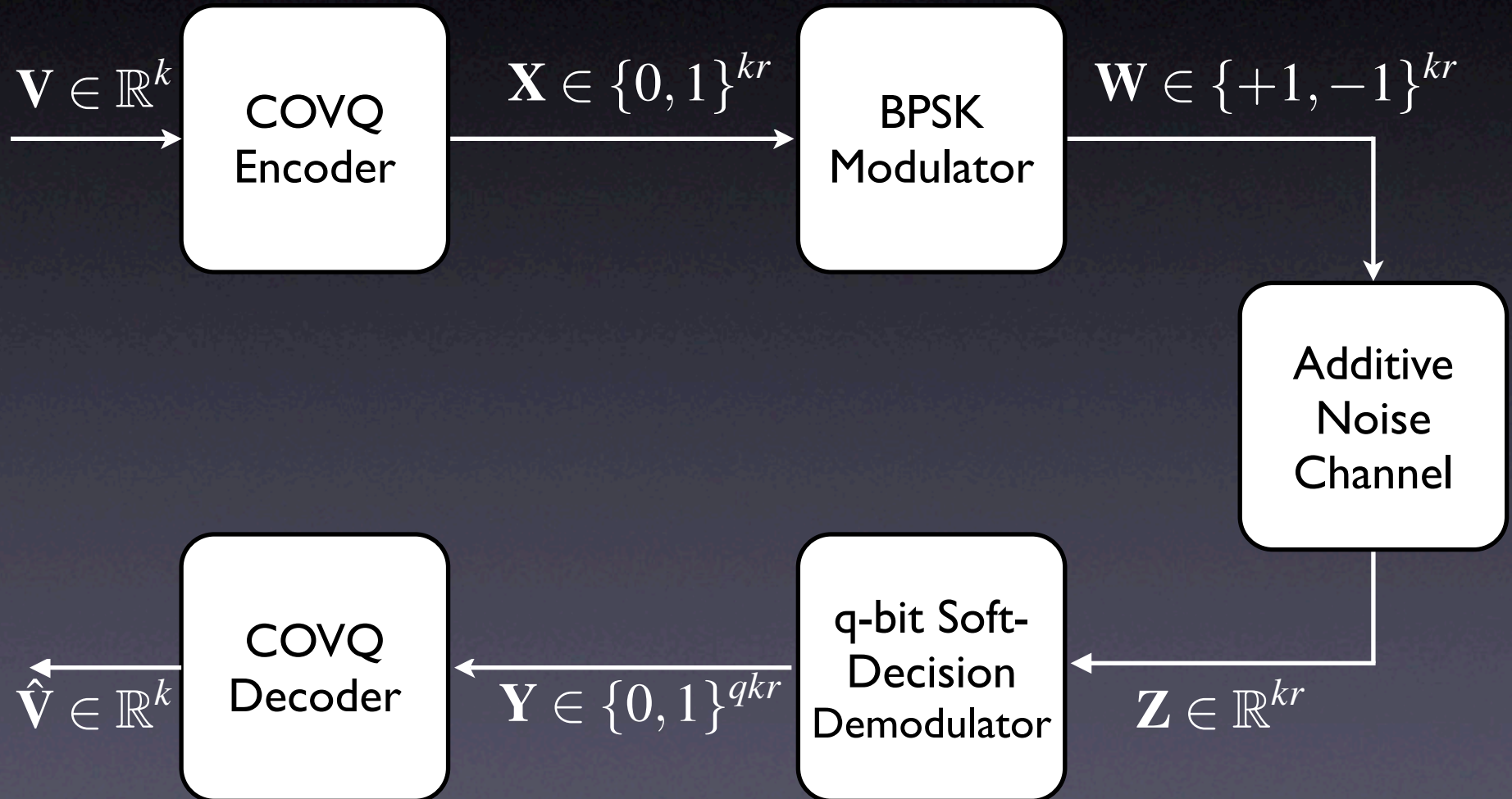
This Year

- Add *soft decision* to the system
- Changes were required in training, transmission, and reception systems

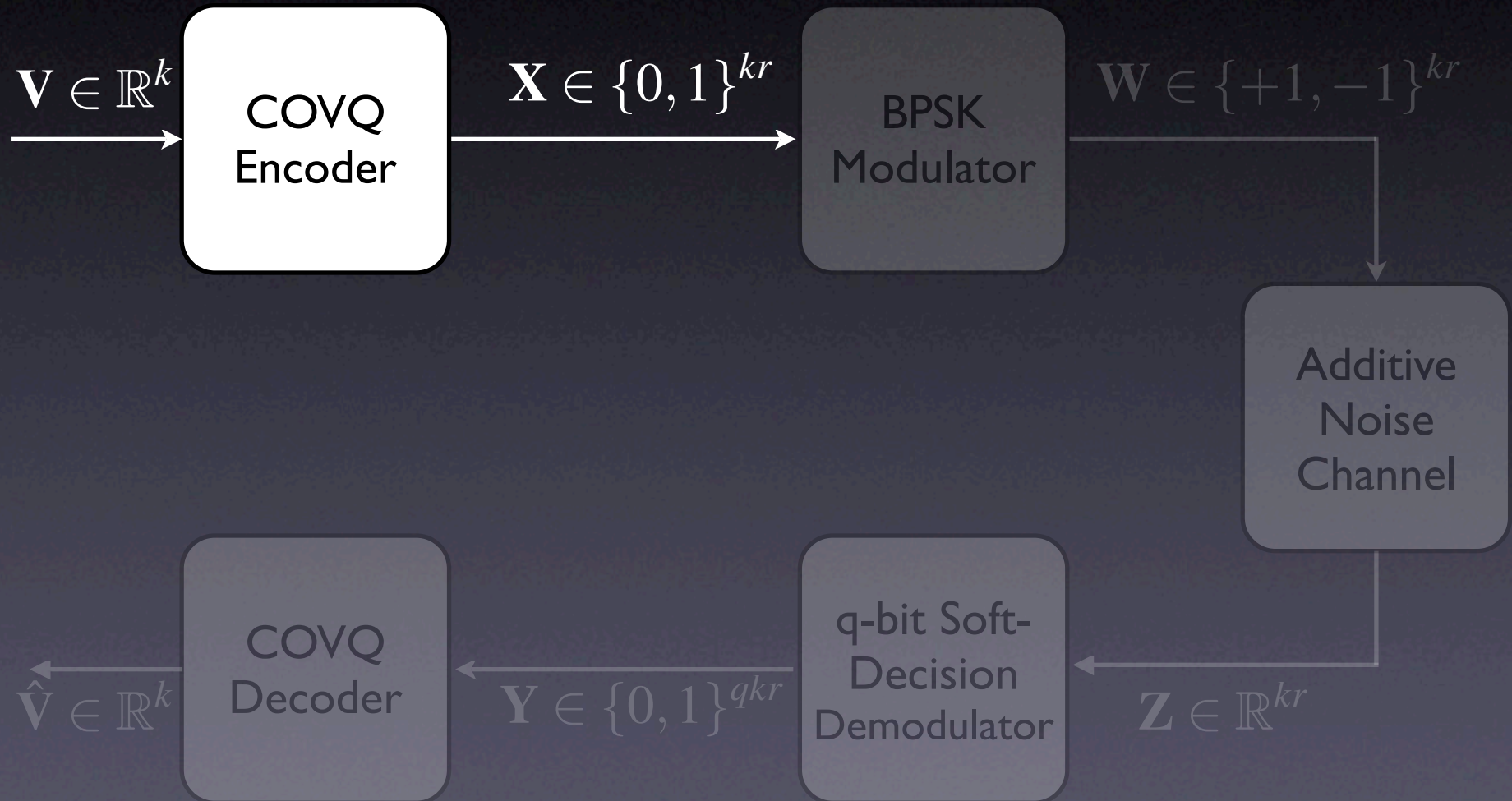
Concepts

- Channel Optimized Vector Quantization
- Soft Decision Demodulation
- LBG Algorithm

COVQ



COVQ

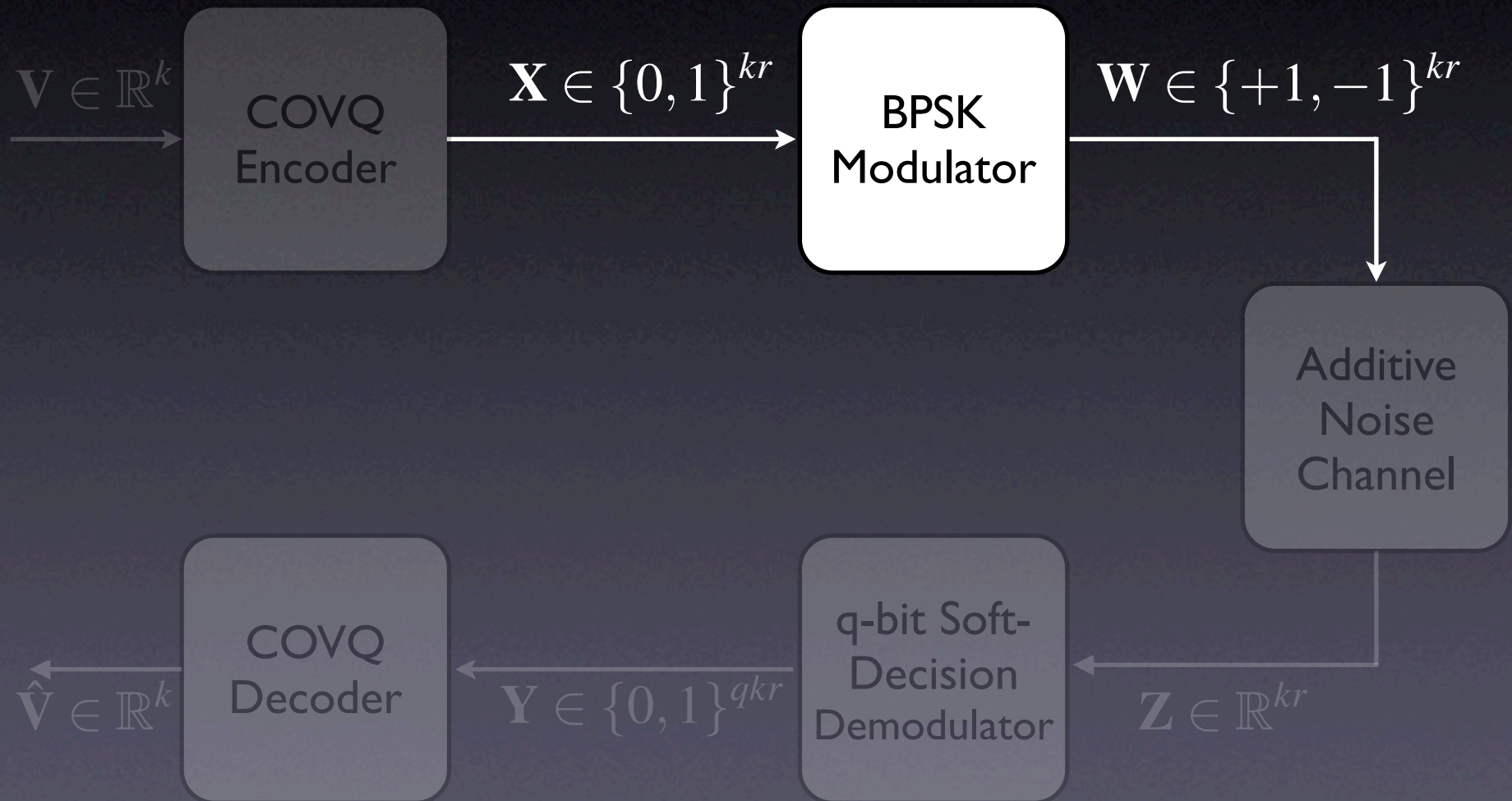


COVQ

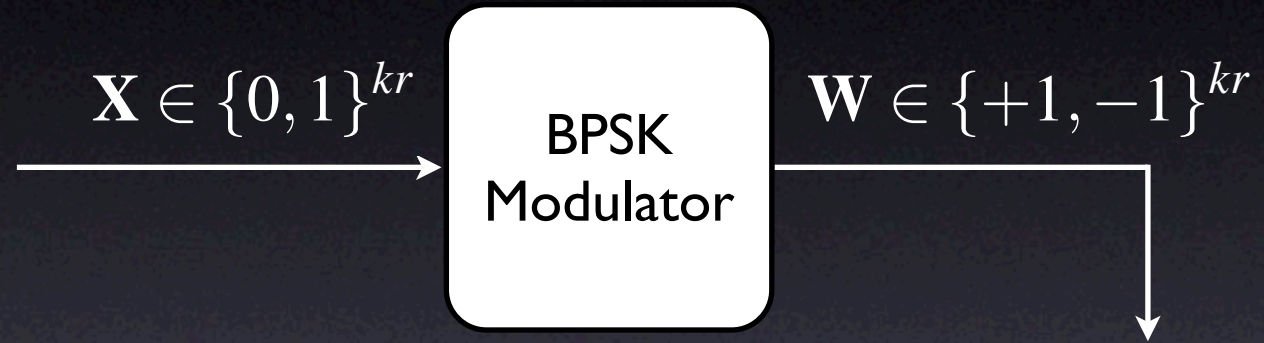


- A vector quantizer takes a vector with k real valued numbers and represents that vector by an index
- That index is between 0 and $2^{kr} - 1$ which can be represented by kr bits

COVQ

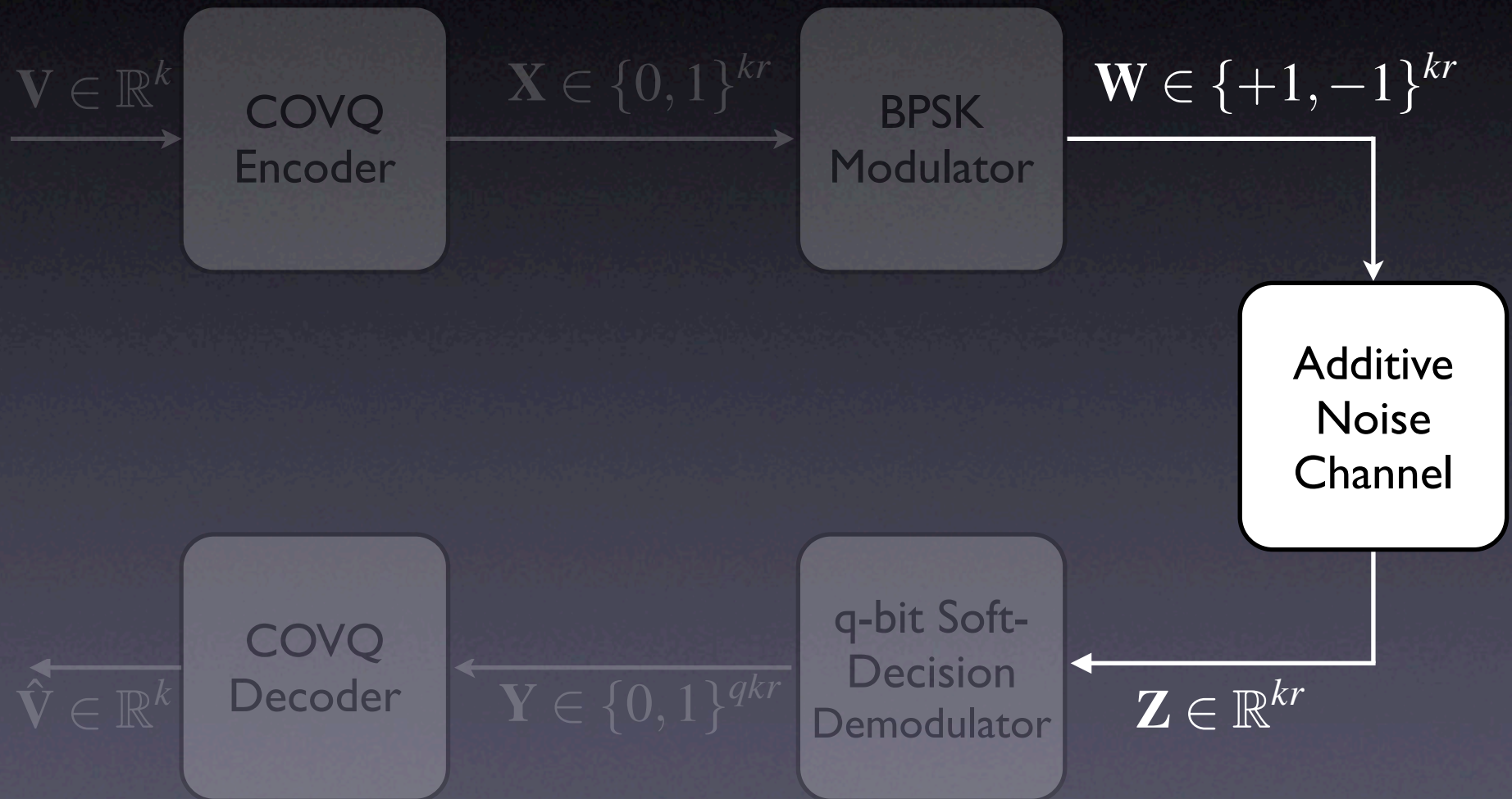


COVQ



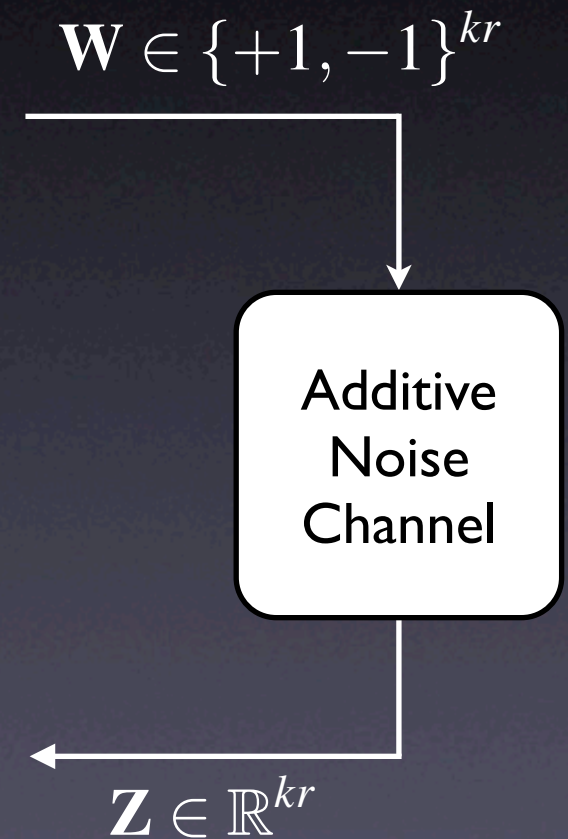
- Takes the kr bits representing our index and *modulates* them
- 0 is modulated to a -1
- 1 is modulated to a +1

COVQ



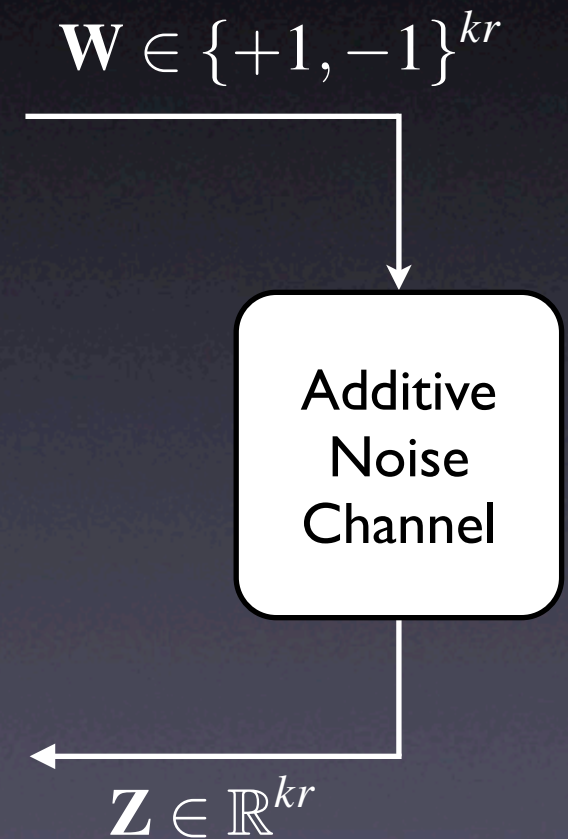
COVQ

- One “bit” at a time is sent over the channel
- This happens kr times for each vector
- Each bit sent is transformed by the channel through the *addition of noise*

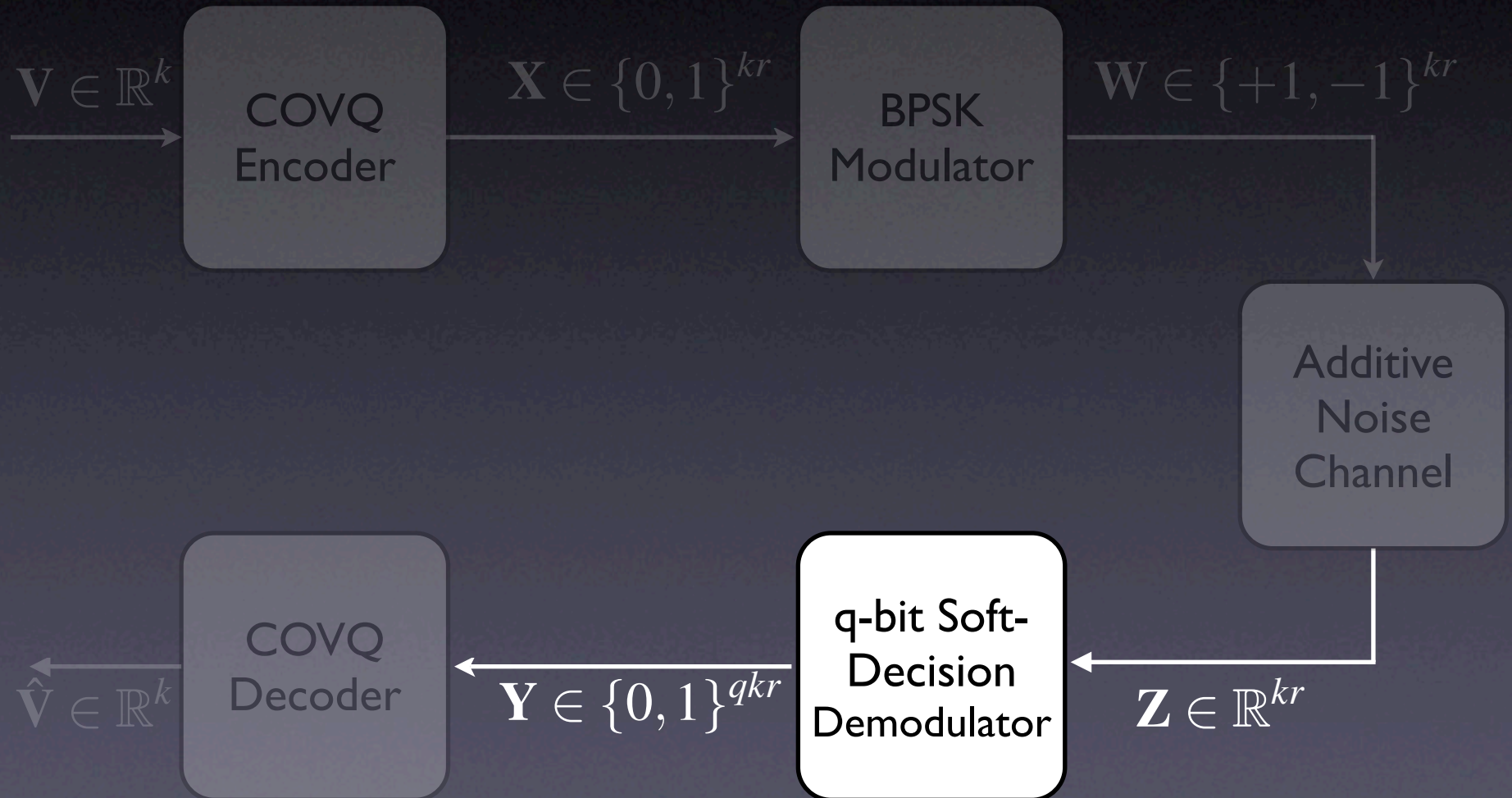


COVQ

- Additive noise means each signal has some random noise added to it
- For each of the kr bits we have :
$$z_i = w_i + n_i$$
where n_i is a zero mean Gaussian random variable

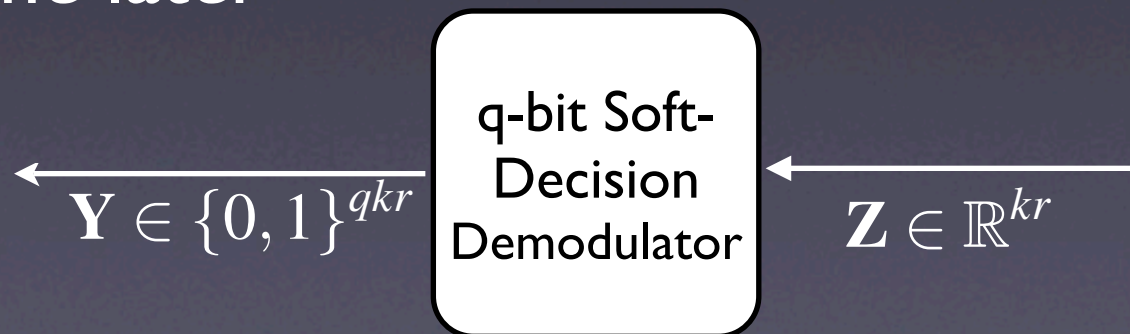


COVQ

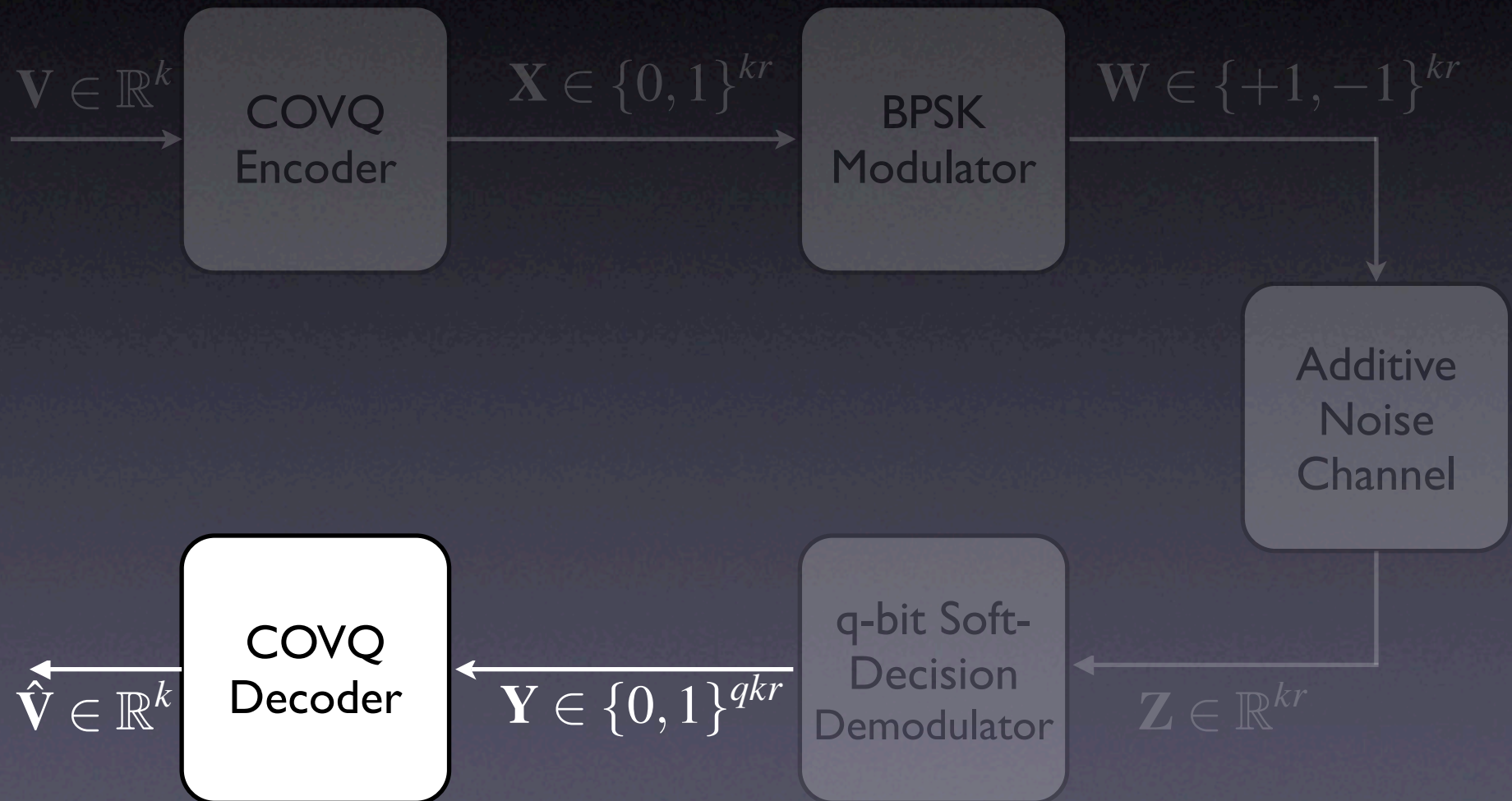


COVQ

- For each kr long vector sent over the channel, we receive kr real values
- Each of these kr real values are quantized using a uniform quantizer with 2^q levels
- This is the key addition over last year's project, more to come later



COVQ



COVQ

- Using a simple lookup table, the index (represented by the qkr bit string \mathbf{Y}) produces an output vector $\hat{\mathbf{V}}$ the estimate of the original input vector



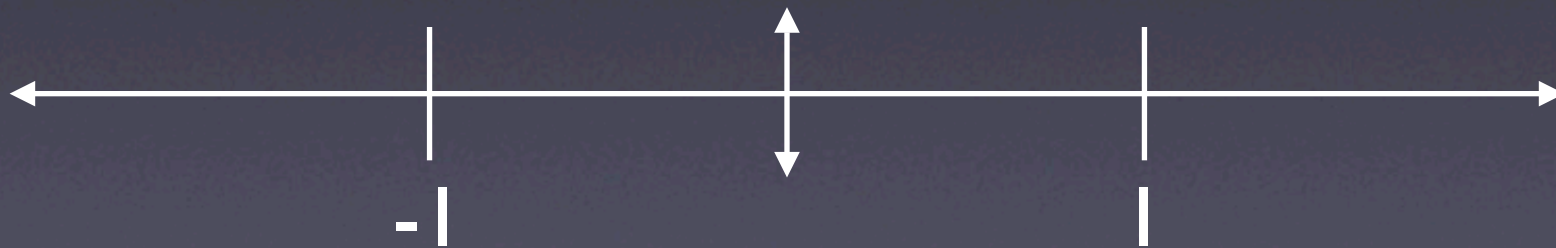
Hard & Soft Decision for BPSK

- Hard decision:
if received value < 0 detect as '-1'
if received value > 0 detect as '+1'
- Soft decision:
now there are 2^q soft decision regions
("bins")

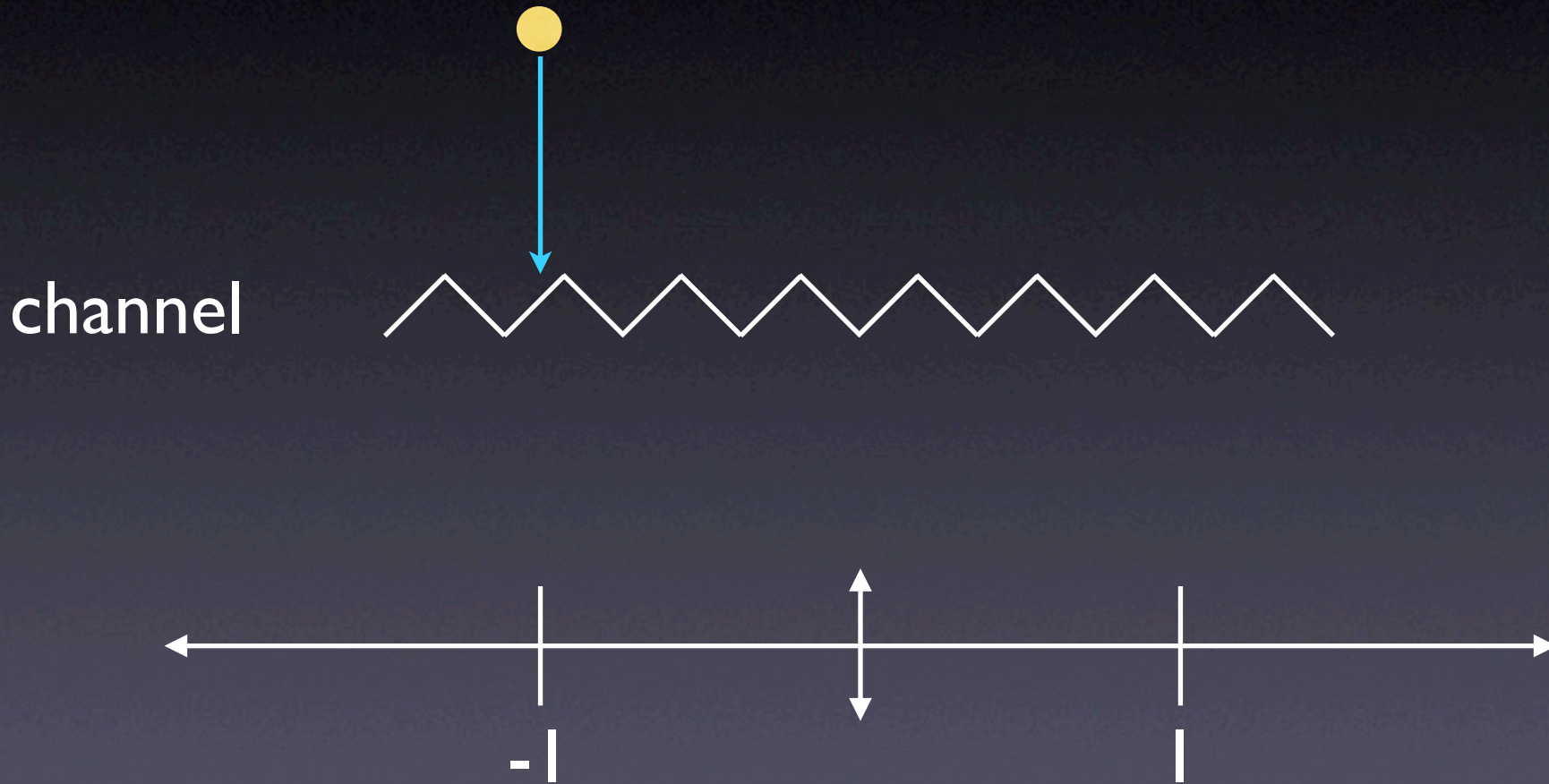
Hard Decision Decoding



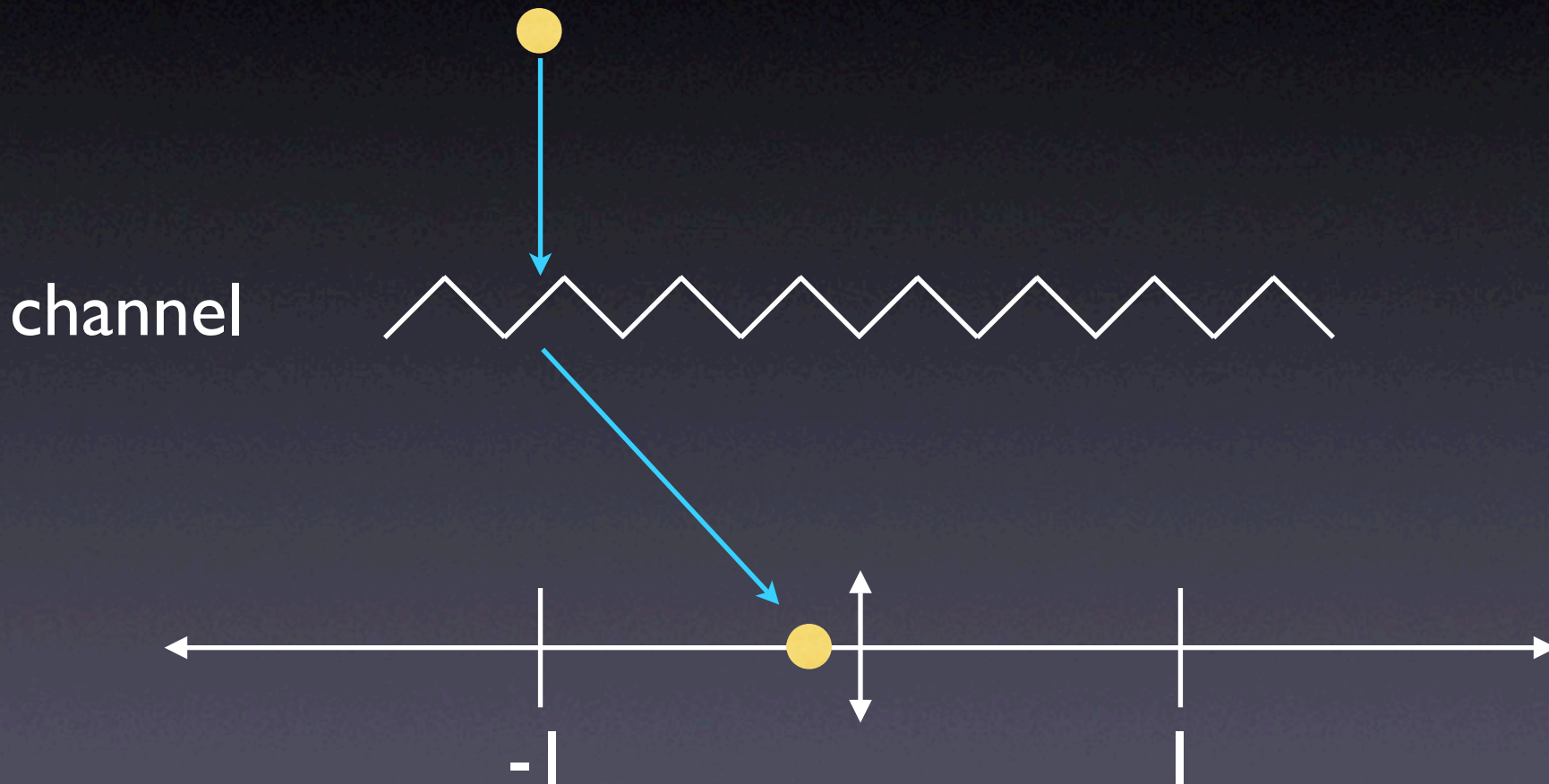
channel



Hard Decision Decoding

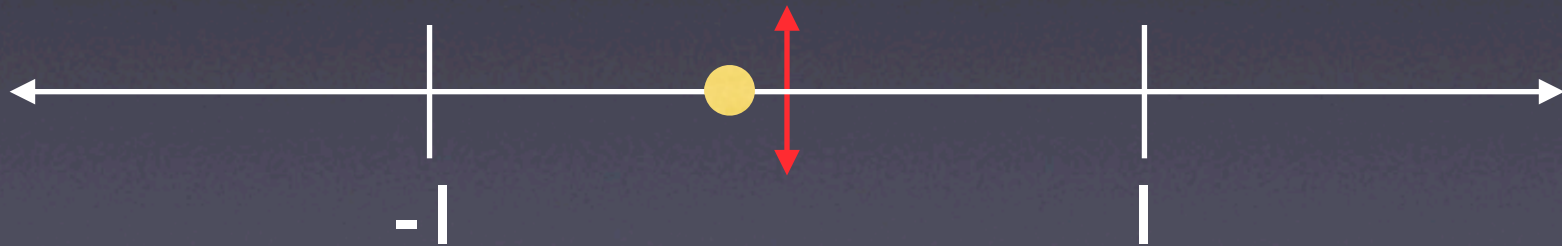


Hard Decision Decoding



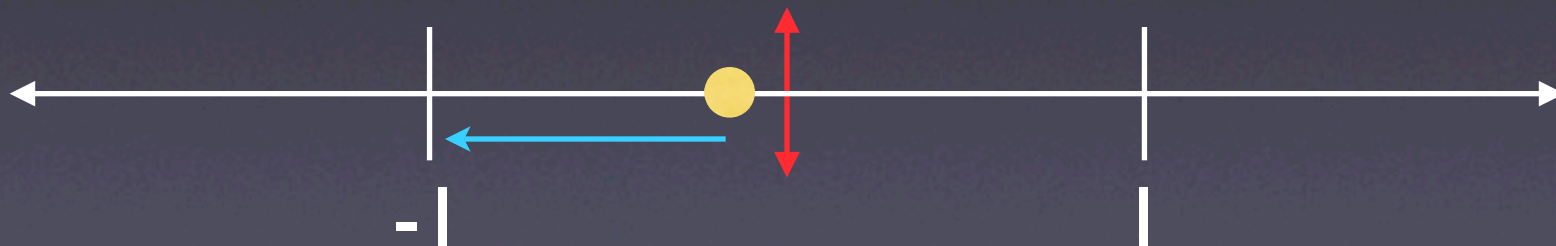
Hard Decision Decoding

channel



Hard Decision Decoding

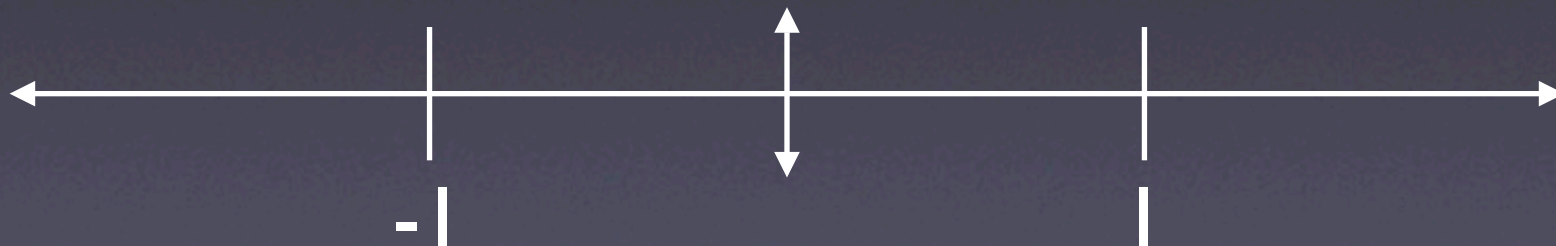
channel



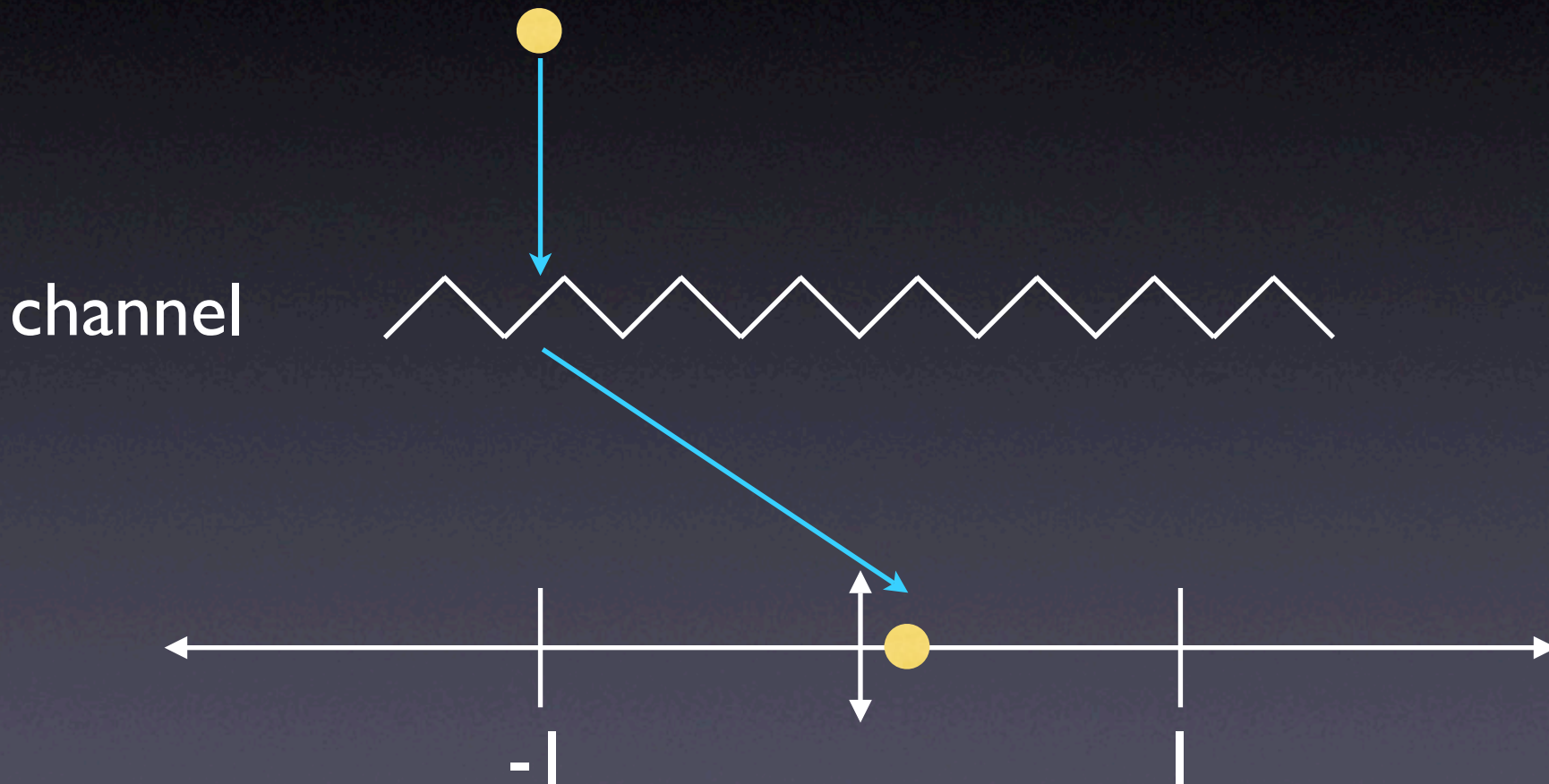
Hard Decision Decoding



channel

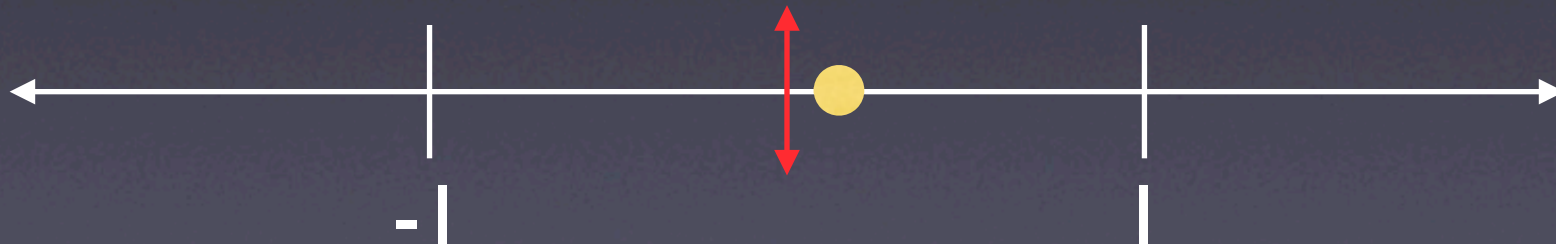


Hard Decision Decoding



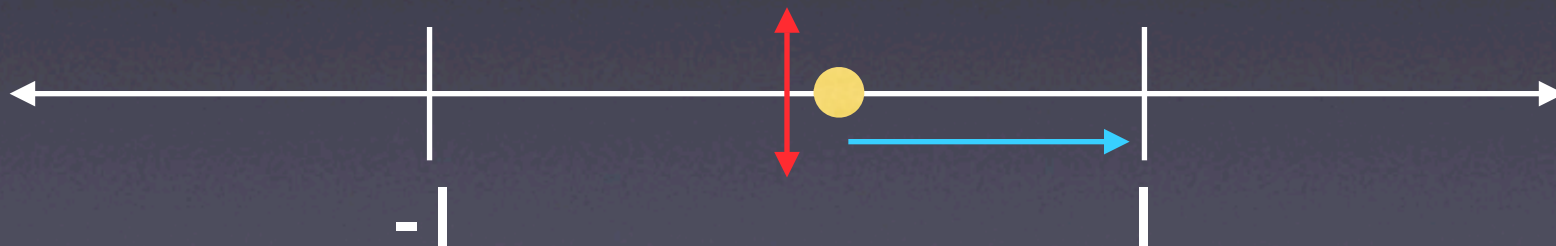
Hard Decision Decoding

channel



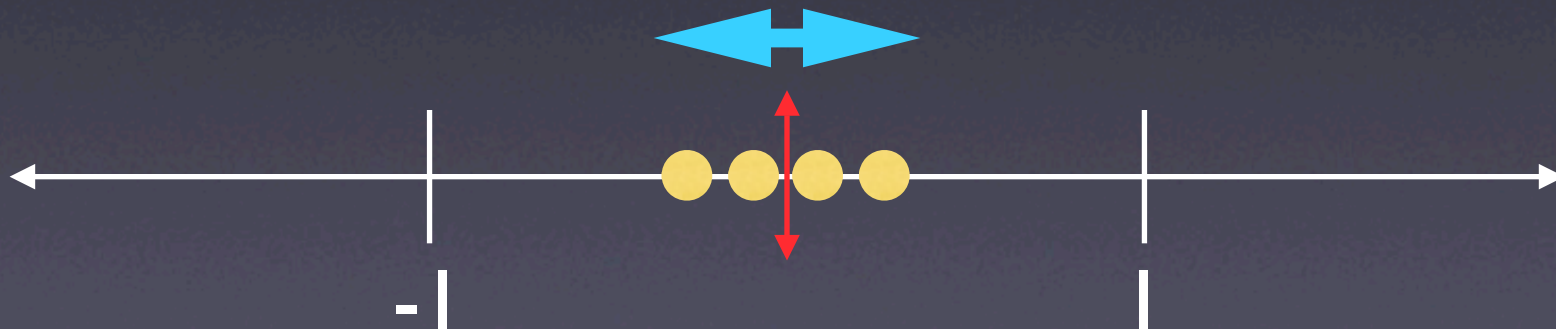
Hard Decision Decoding

channel



Hard Decision Decoding

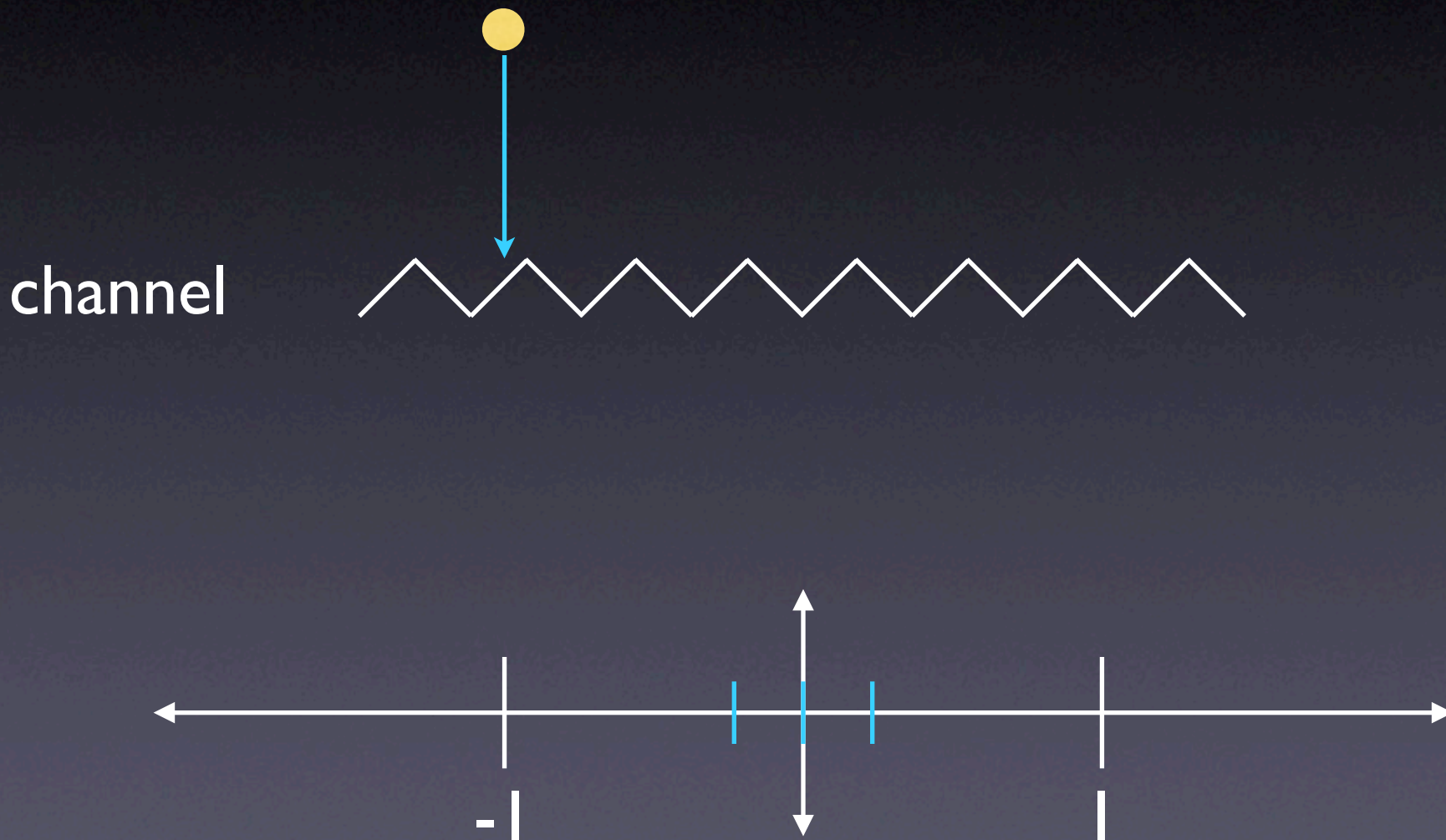
- It is a close call for bits detected near the centre
- Just as likely to have been sent as either bit originally



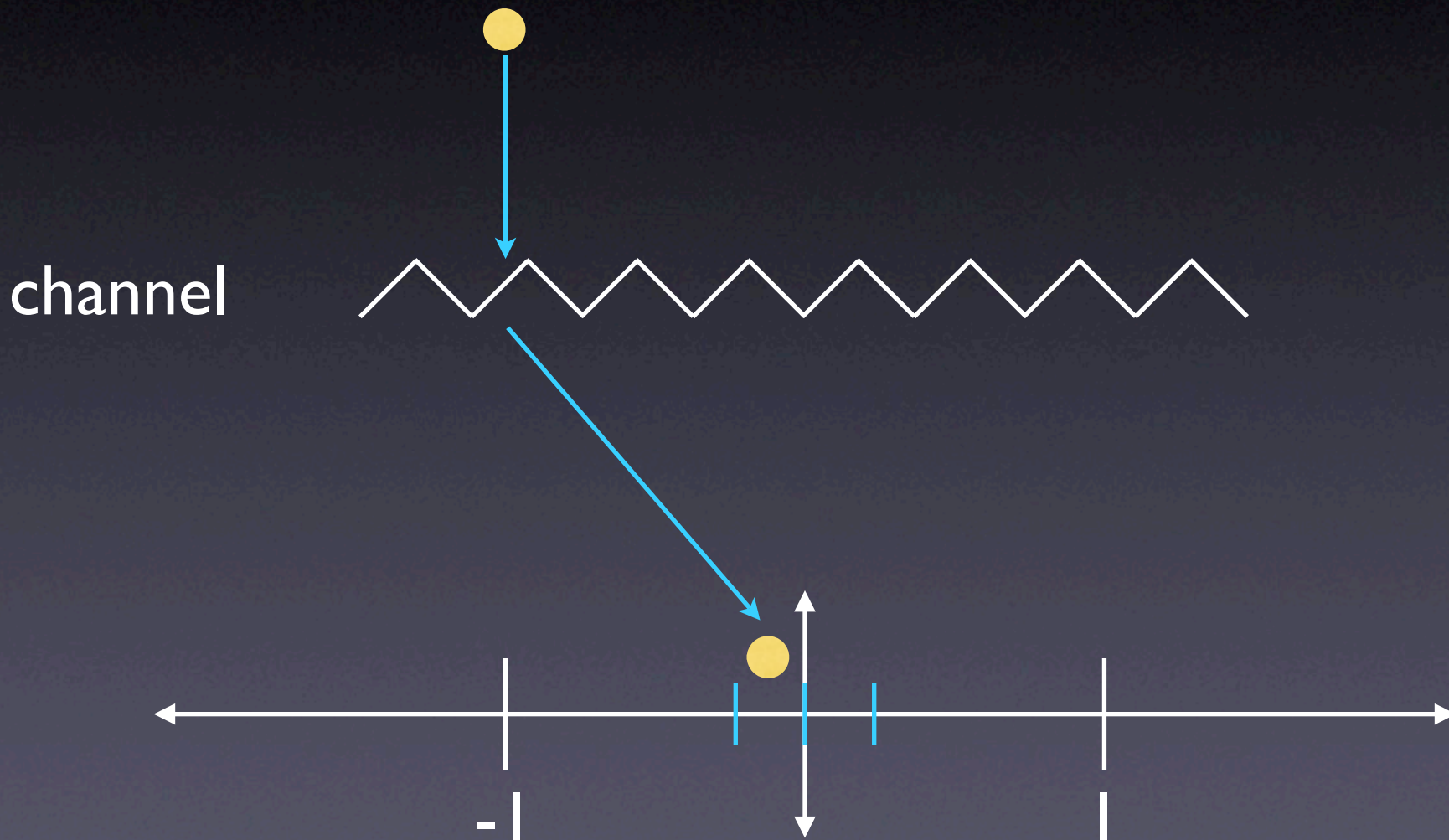
Soft Decision Decoding

- The solution is soft decision decoding
- This means that a '-1' sent over the channel could be decoded as an intermediate value between '-1' and '+1'

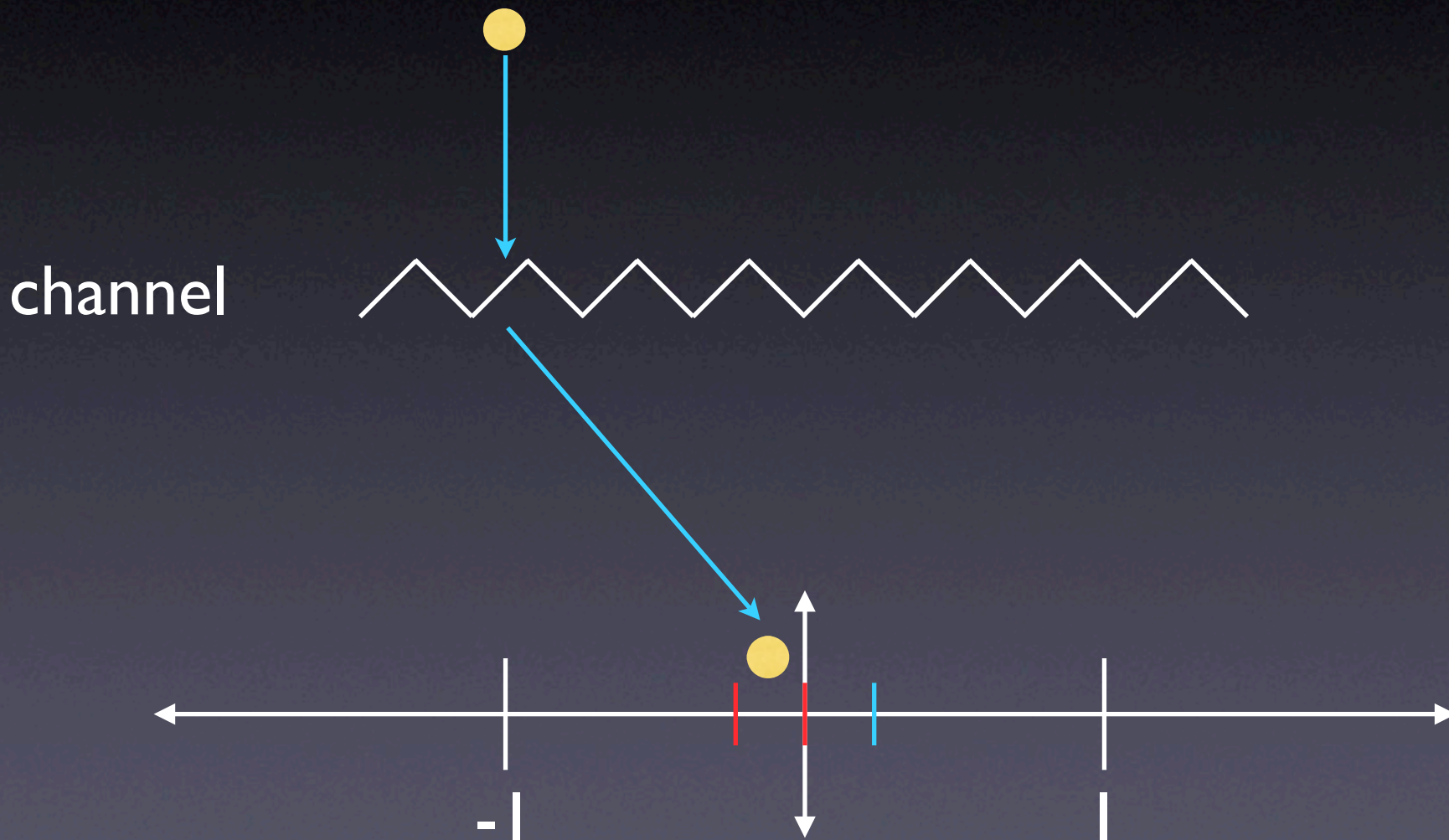
Soft Decision Decoding



Soft Decision Decoding



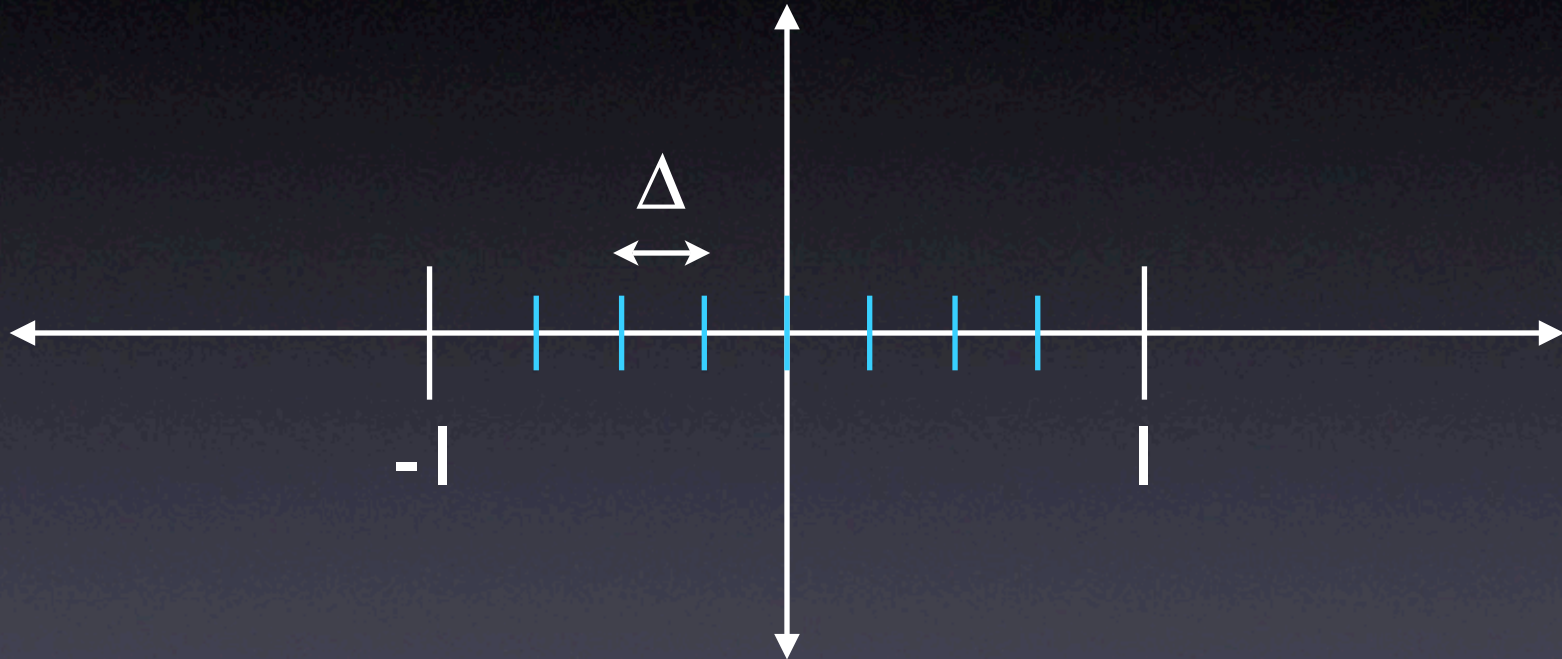
Soft Decision Decoding



Soft Decision Decoding

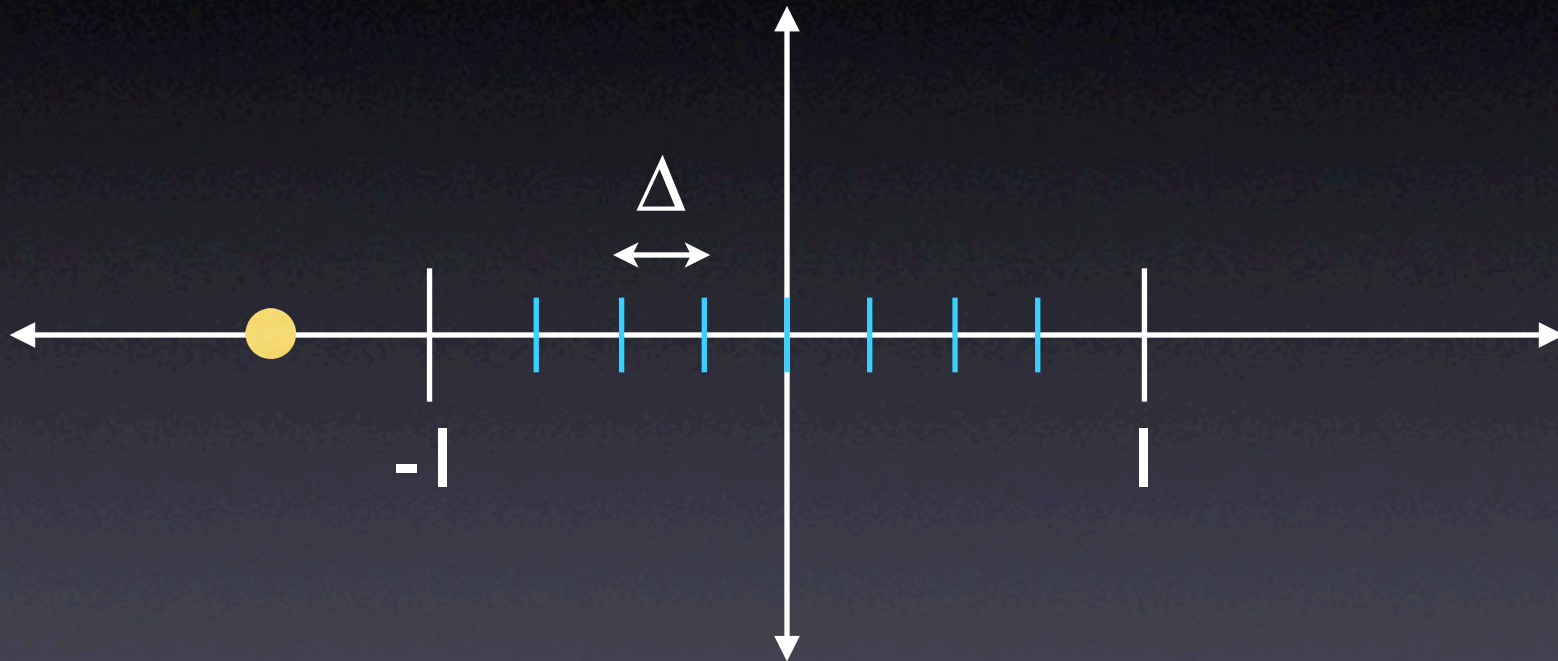
- What about these bins?
- They depend on how noisy the channel is
- The more noise on the channel, the more error we have → wider bins

Soft Decision Decoding



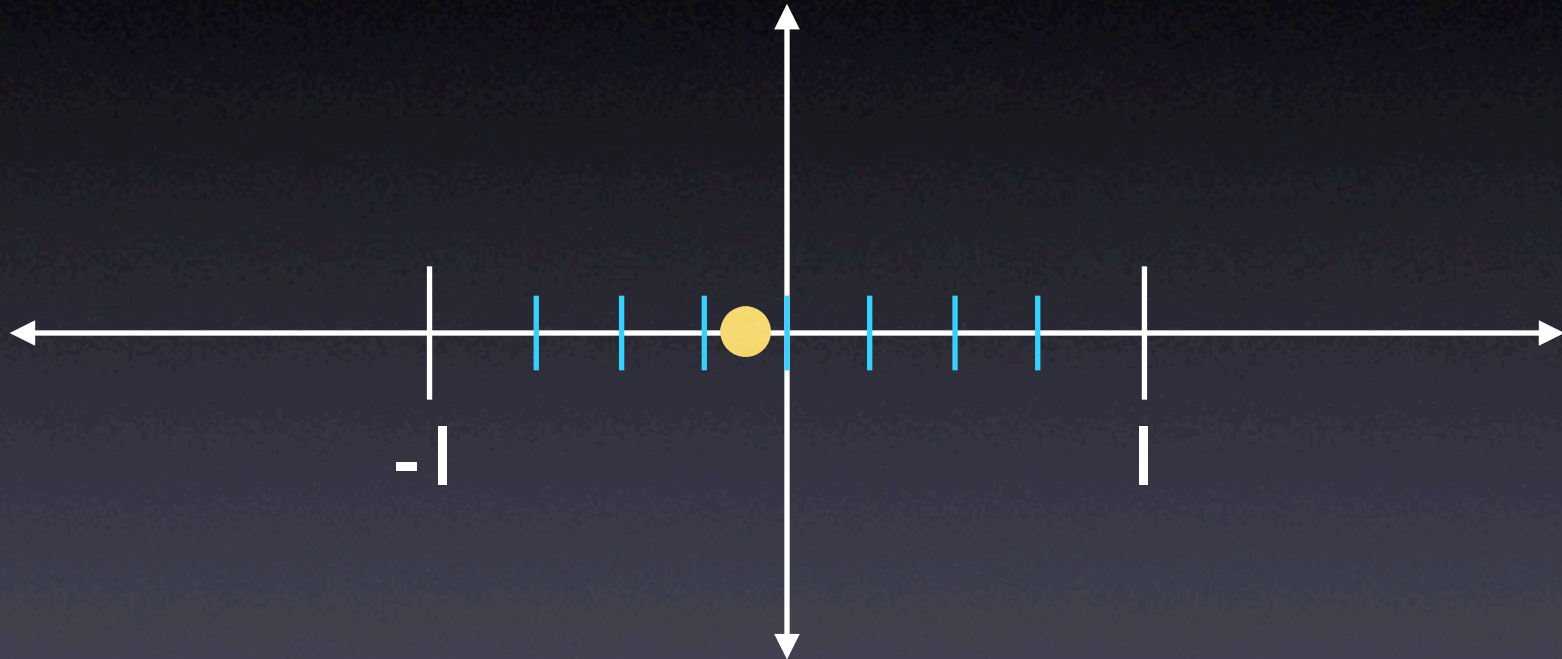
example: $q = 3$ (8 bins) CSNR = 5, $\Delta = .2340$

Soft Decision Decoding



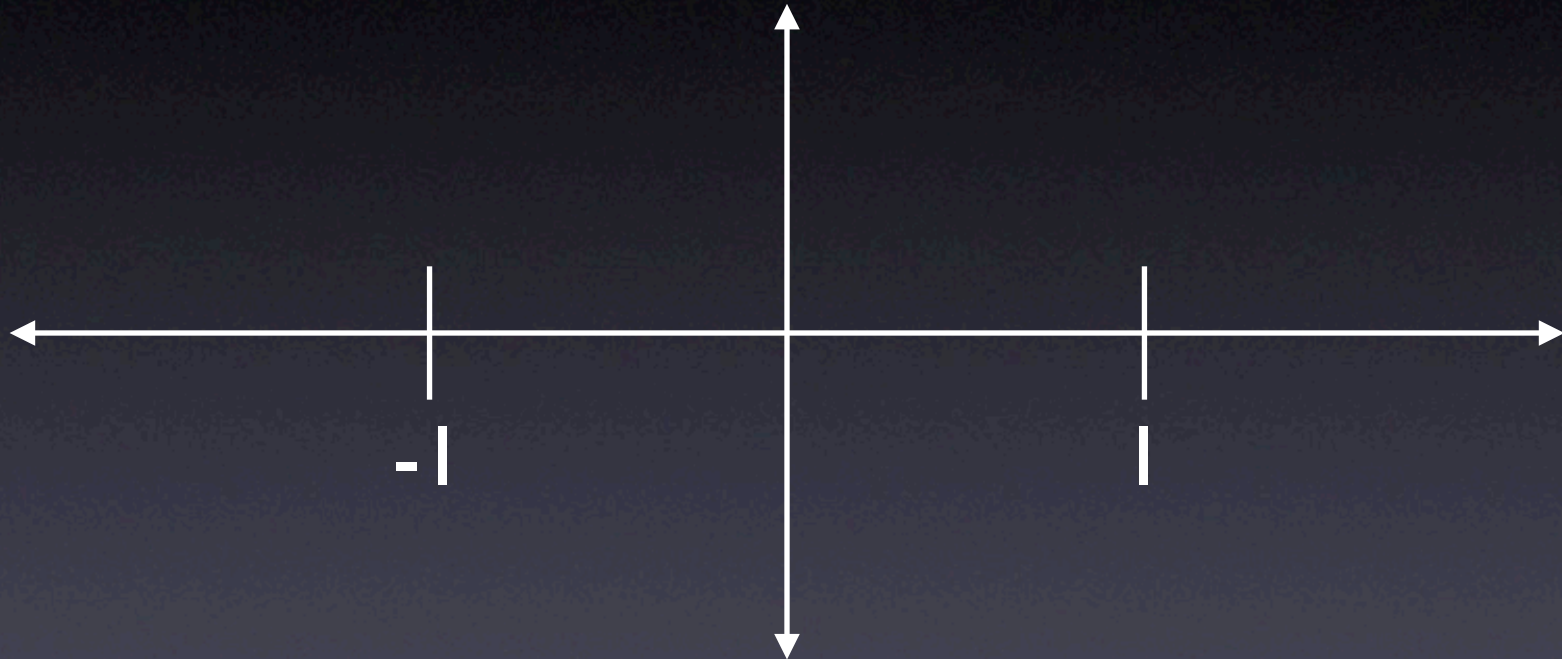
example: $q = 3$ (8 bins) CSNR = 5, $\Delta = .2340$

Soft Decision Decoding

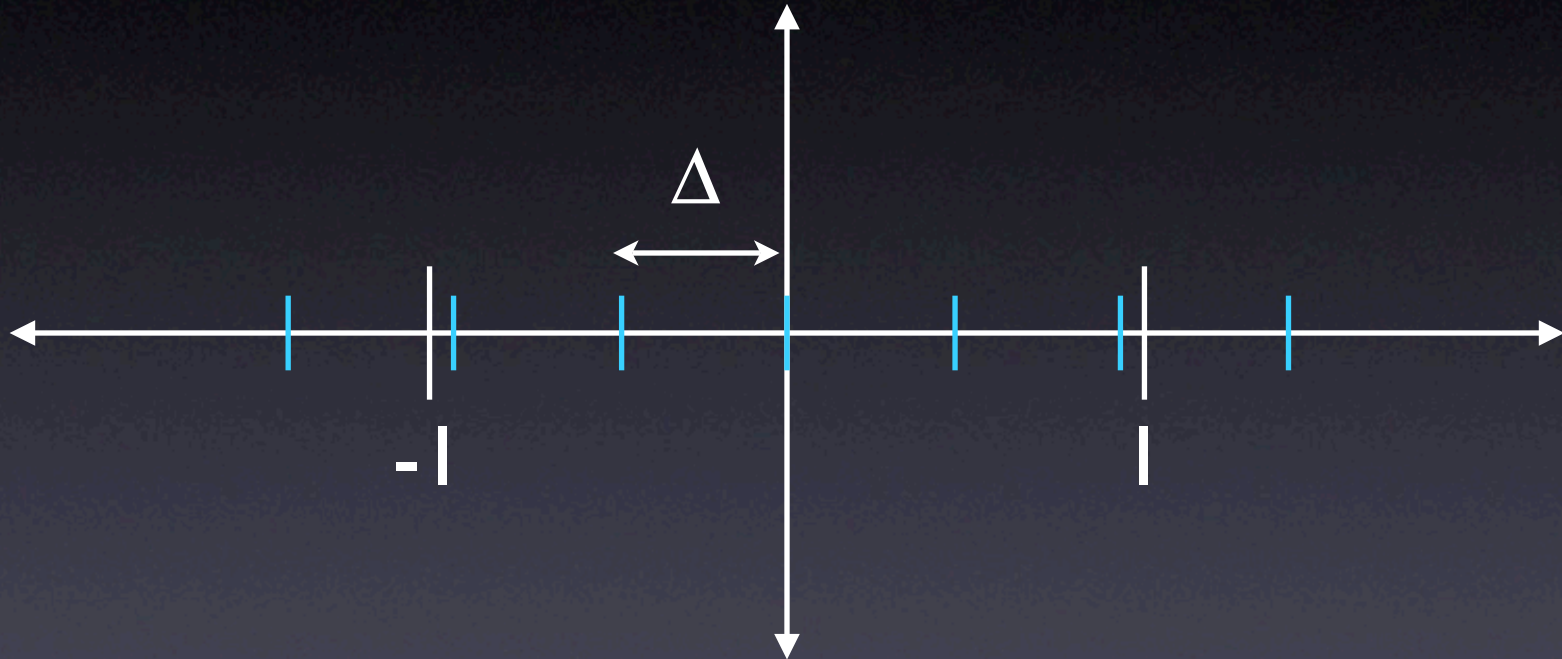


example: $q = 3$ (8 bins) CSNR = 5, $\Delta = .2340$

Soft Decision Decoding

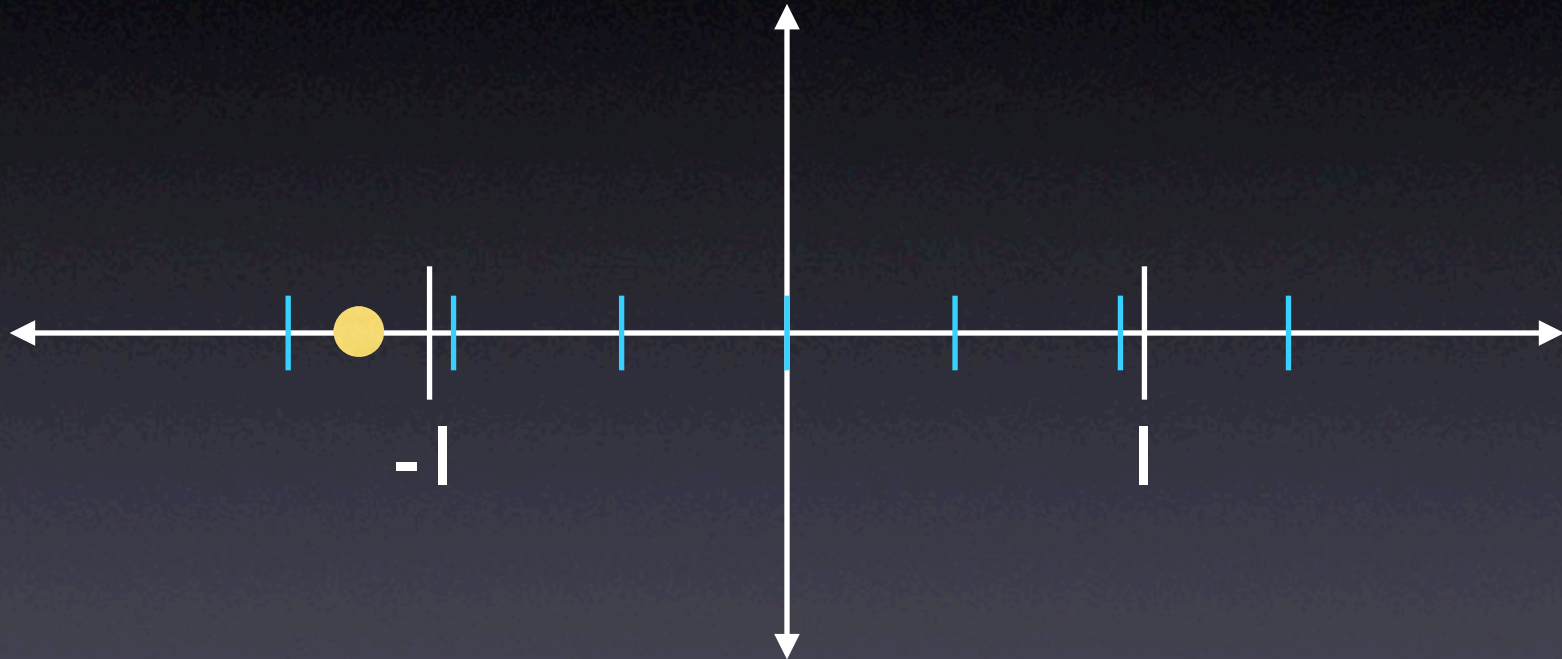


Soft Decision Decoding



example: $q = 3$ (8 bins), $CSNR = 0$, $\Delta = .4815$

Soft Decision Decoding



example: $q = 3$ (8 bins), CSNR = 0, $\Delta = .4815$

Soft Decision

- How does this help us send pictures?

Soft Decision

- Since we have the statistics of the channel we can calculate the probabilities of sending an index i and receiving an index j

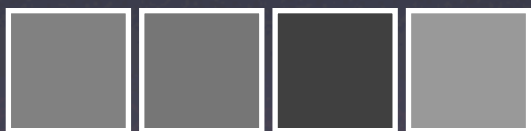
Soft Decision

- What are these indices? They represent a vector of pixel values.

Soft Decision



channel

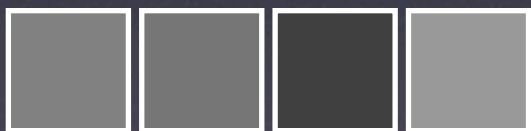


The idea is that we want to reconstruct these pixels in a way that they are as close as possible to the original ones

Soft Decision

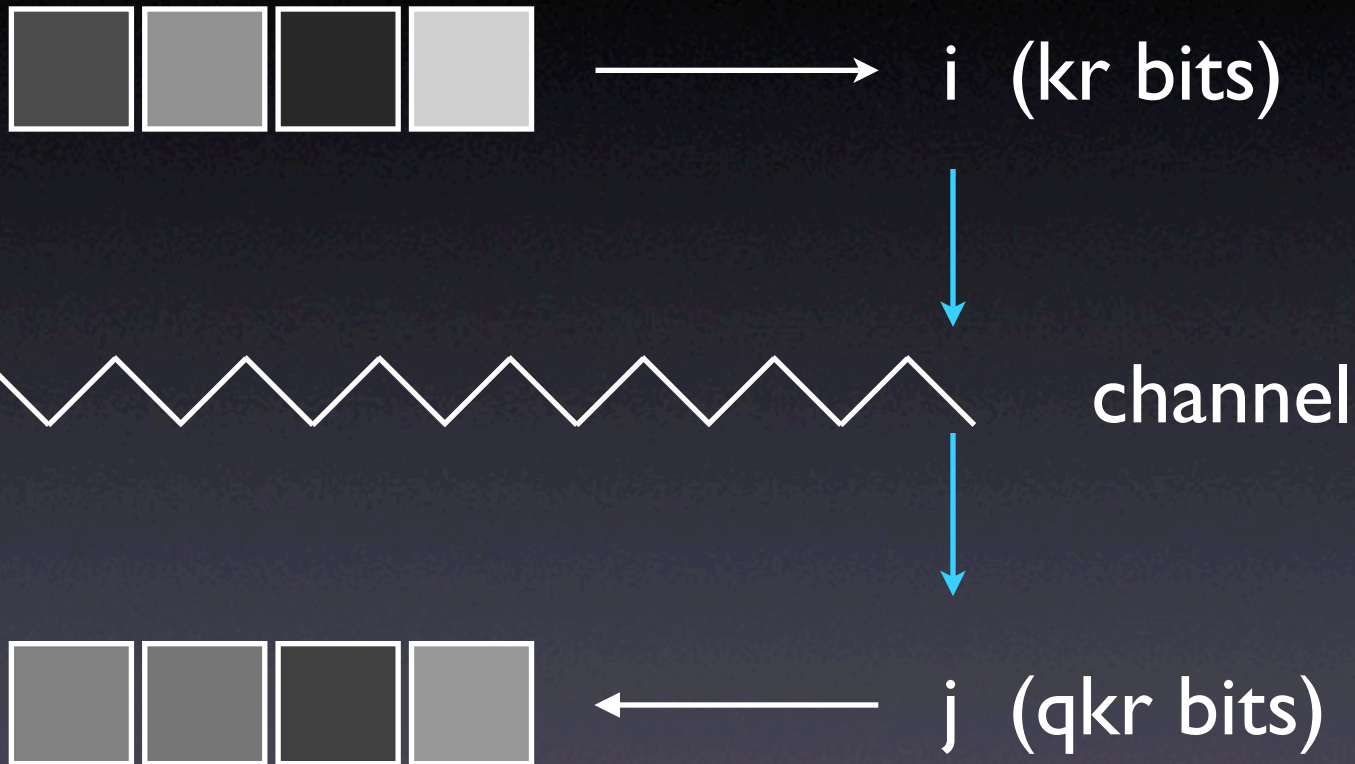


channel



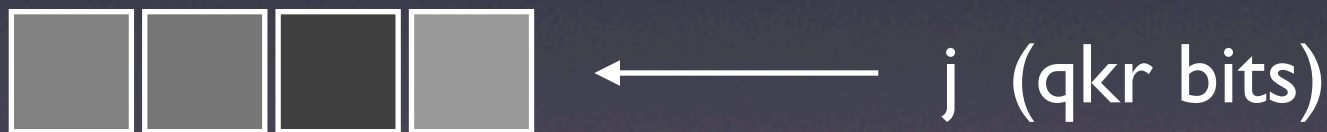
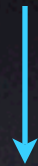
We only have a fixed collection of vectors to decode to, ones that are in our 'codebook'

Soft Decision



But this is vector quantization. We don't send pixels, we send indices.

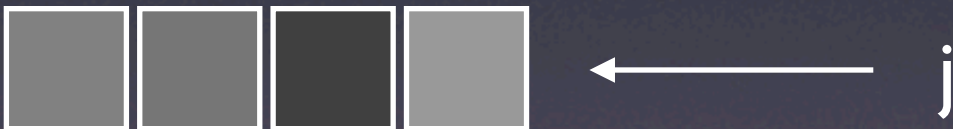
Soft Decision



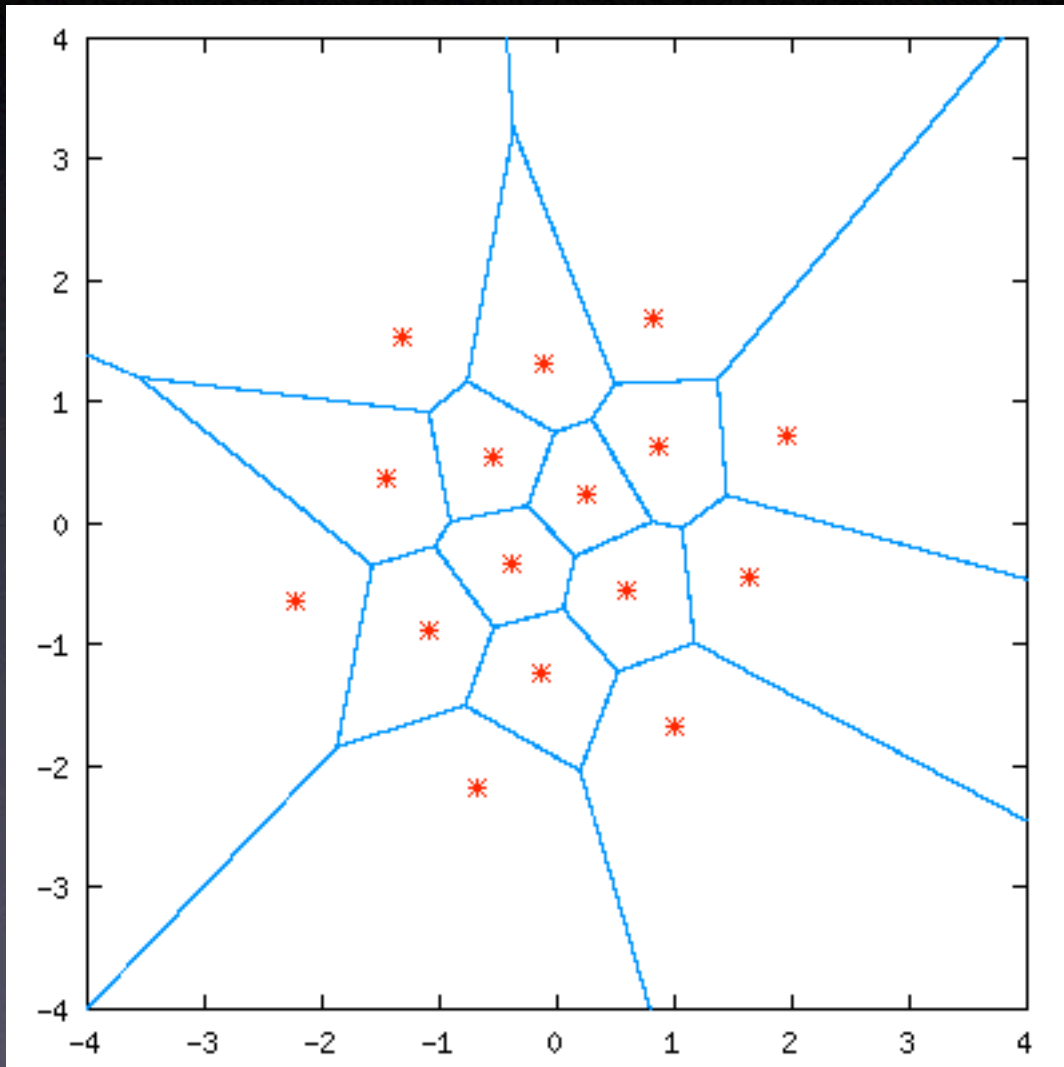
Our goal is to pick an i to send that minimizes the difference between the sent group of pixels and the decoded group of pixels represented by j

Soft Decision

- Where do these j 's come from?
- In our case they represent groups of pixels. These groups form our codebook
- We run the LBG algorithm to construct our codebook



What's a Quantizer?



- If a sample you want to quantize falls in a region, you send the index of the region across the channel
- On the other side of the channel, your estimate of the original sample is the representative of that region

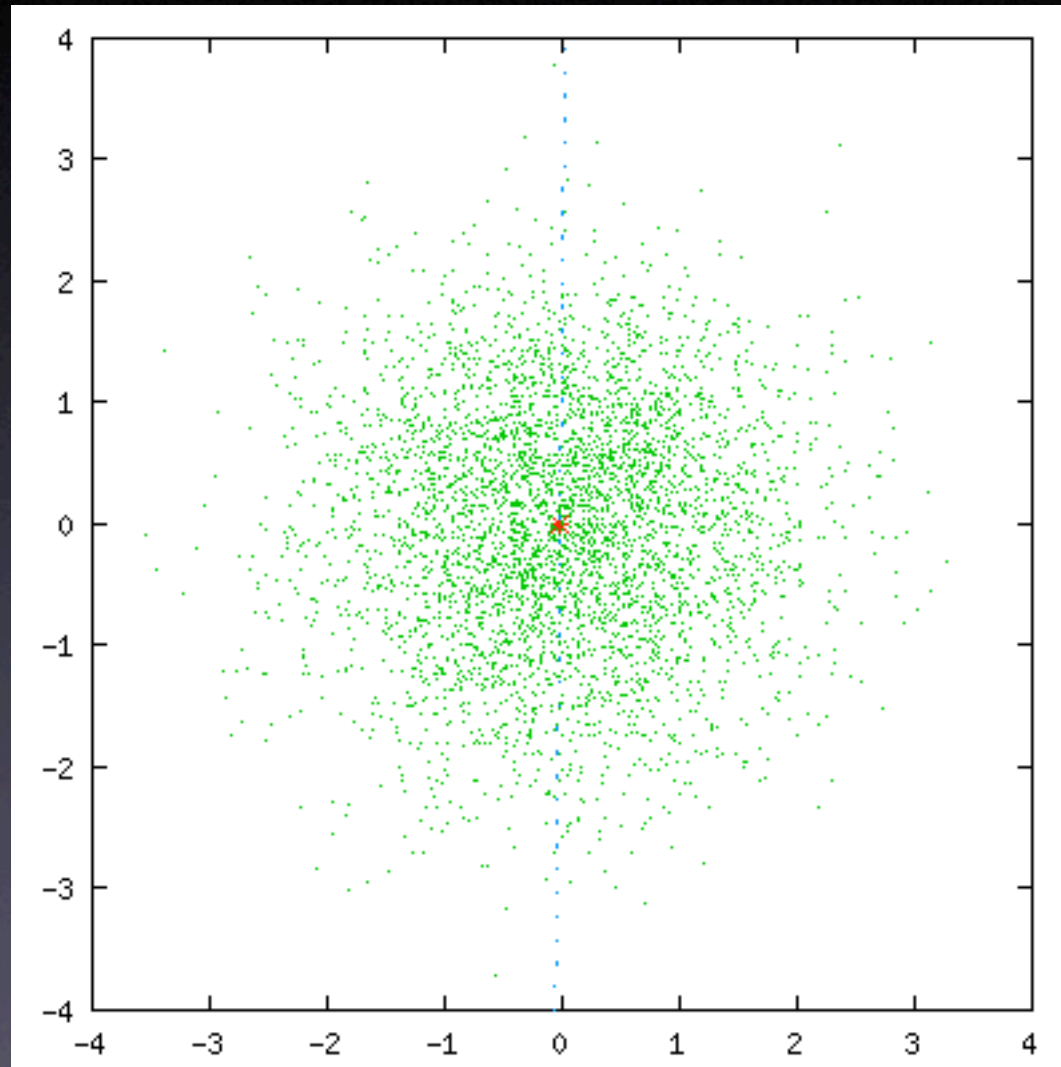
LBG Algorithm

- Nearest Neighbour:
 - Finds the regions
- Centroid Condition:
 - Finds the representative of each region

LBG Algorithm

- We will look at a 2D case
- This is like taking pixels in groups of 2
- They can be represented by an ordered pair (x,y) , with x and y numerically representing how dark the pixel is

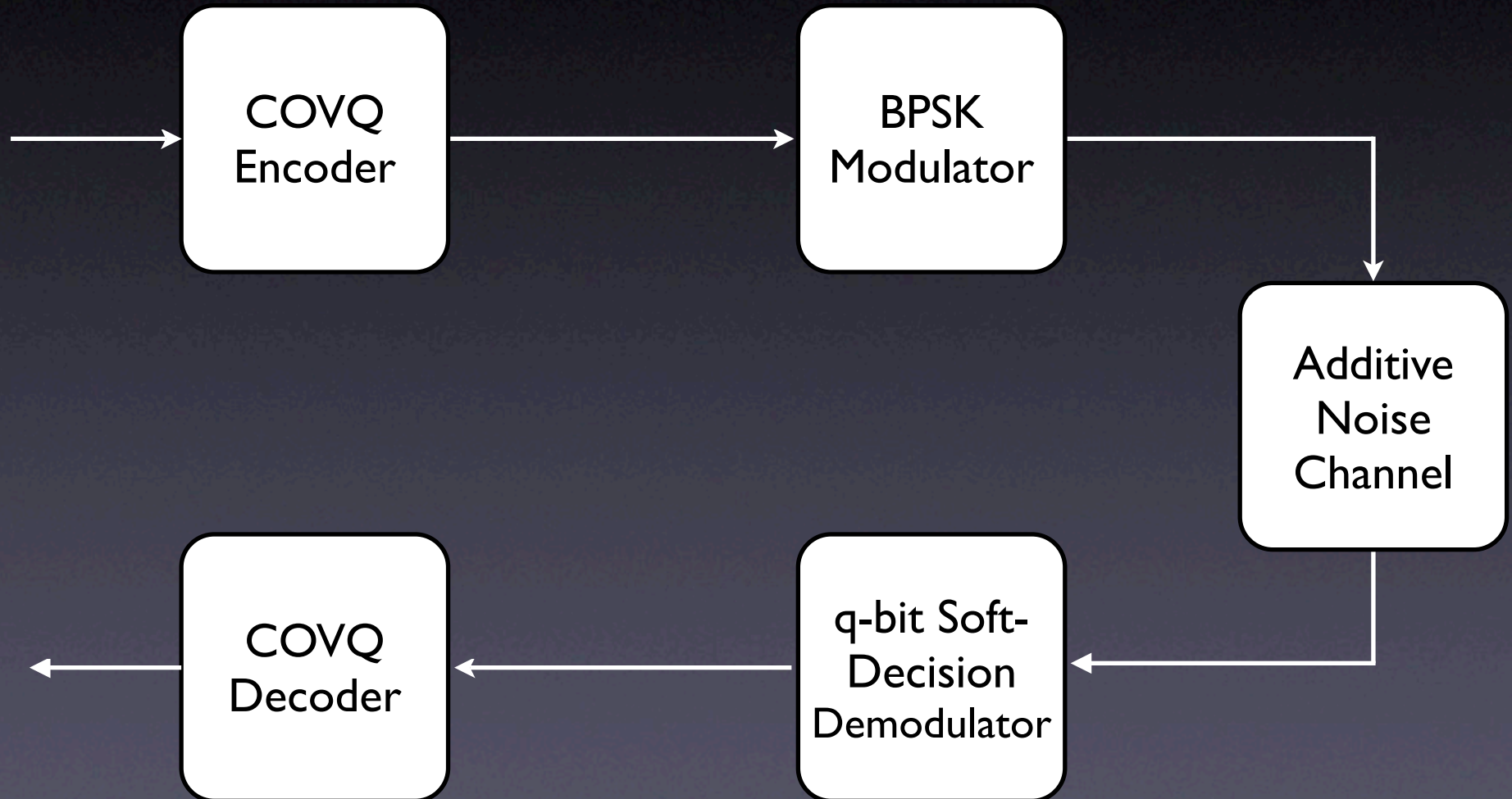
LBG Algorithm



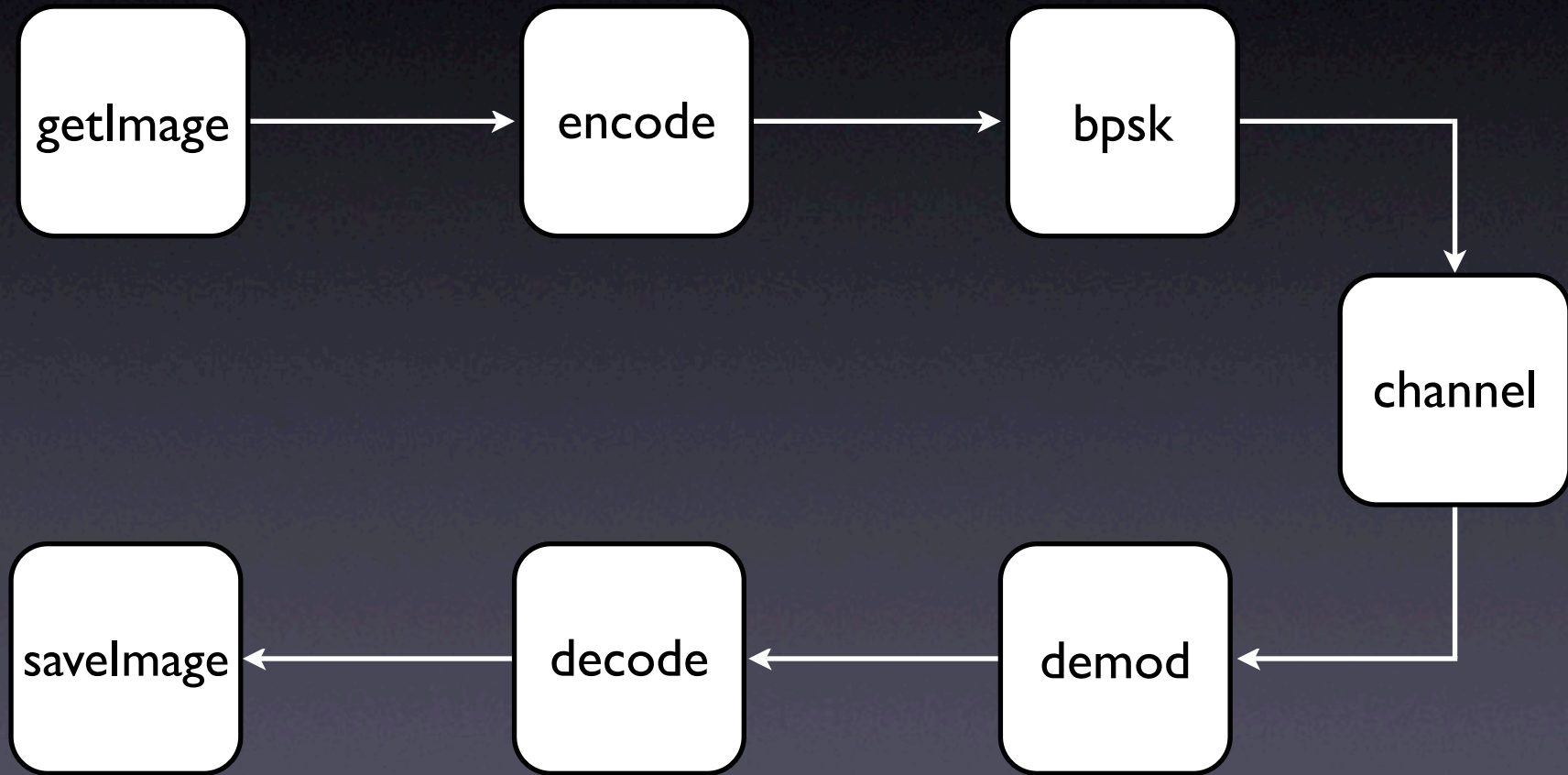
Software Design

- Pipes & Filters Design Pattern
- Simulation Scripts

Pipes and Filters



Pipes and Filters



Pipes and Filters

- To run a simulation we would make the following call on a Unix system (modulo program arguments)

```
getImage | encode | bpsk | channel | demod | decode | saveImage
```

Pipes and Filters

- *Advantages:*
 - Easy to develop, test, and debug since the large program is broken up into smaller and simpler ones
 - It is possible to write each program out of order (like we did!) and put the pieces together at the end

Pipes and Filters

- *Disadvantages:*
 - Slightly slower runtime since you are not manipulating memory between components
 - Many components and parameters to keep track of

#!Scripting

- Had 2 Perl scripts
 - Codebook creation
 - Simulation
- Have many parameters (dimension, rate, q-value, CSNR, orientation,) all of which have multiple possible values.
 - Could have 1000+ codebooks to create!

Results

- Image transmission examples
- Some fancy graphs of PSNR vs. CSNR

Image Transmission

- The following slide shows the results of a simulated image transmission for $q = 1,2,3$
- Note this image was NOT one that was used to train the codebook



Original



Q: 1 CSNR: 15.00 (d 2, r 1.00, o 1)



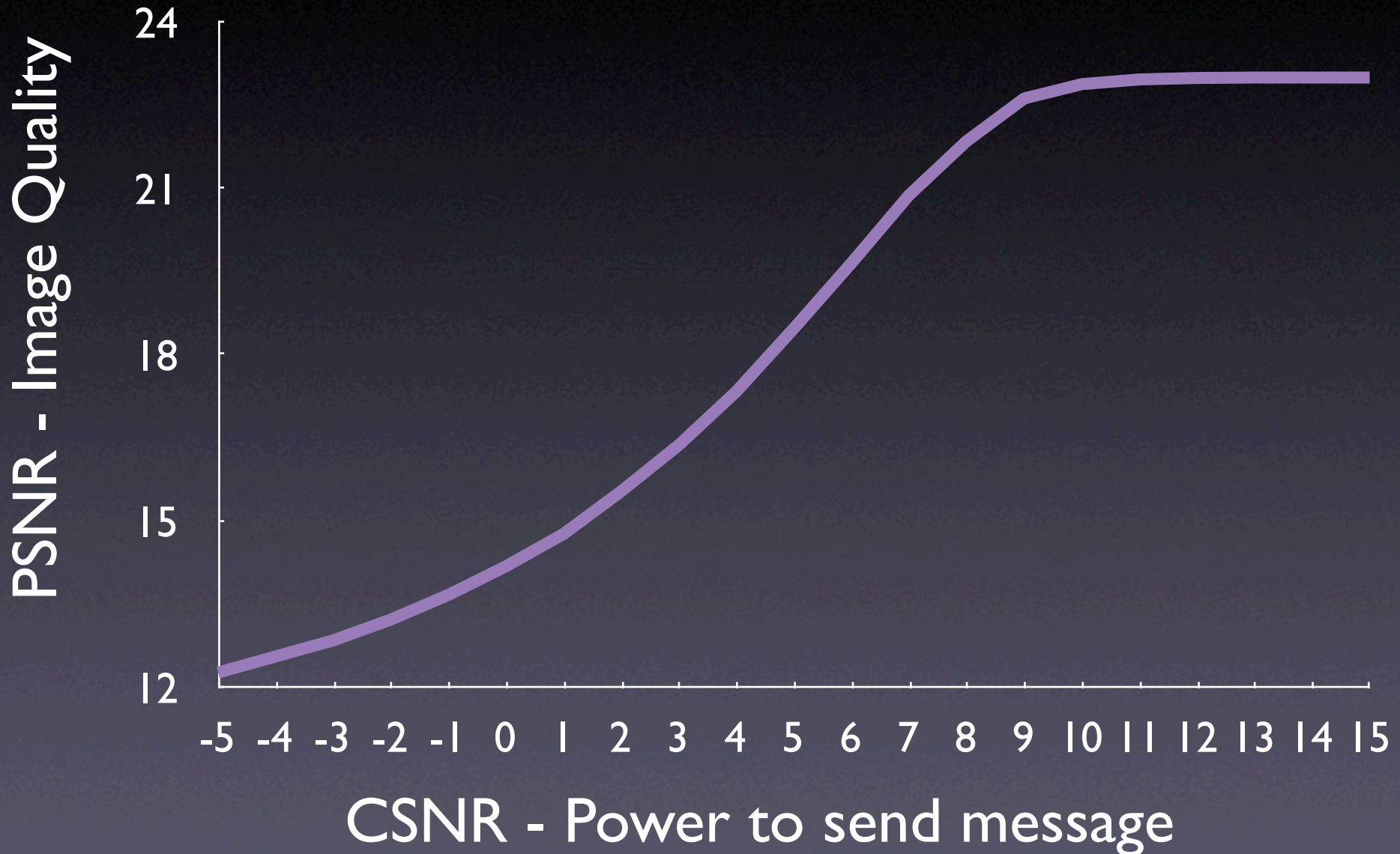
Q: 2 CSNR: 15.00 (d 2, r 1.00, o 1)



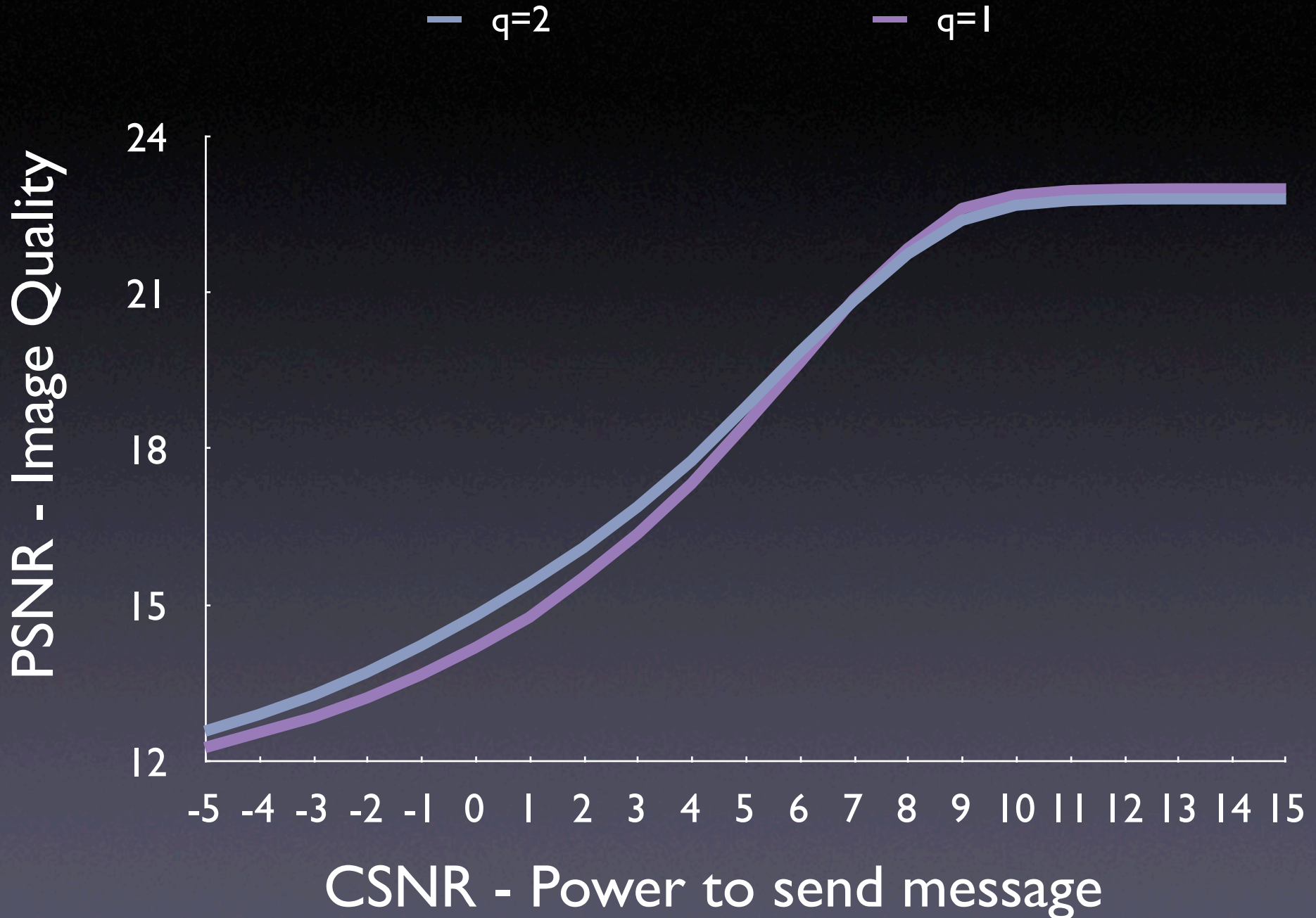
Q: 3 CSNR: 15.00 (d 2, r 1.00, o 1)

PSNR vs. CSNR for glasses.pgm

— $q=1$

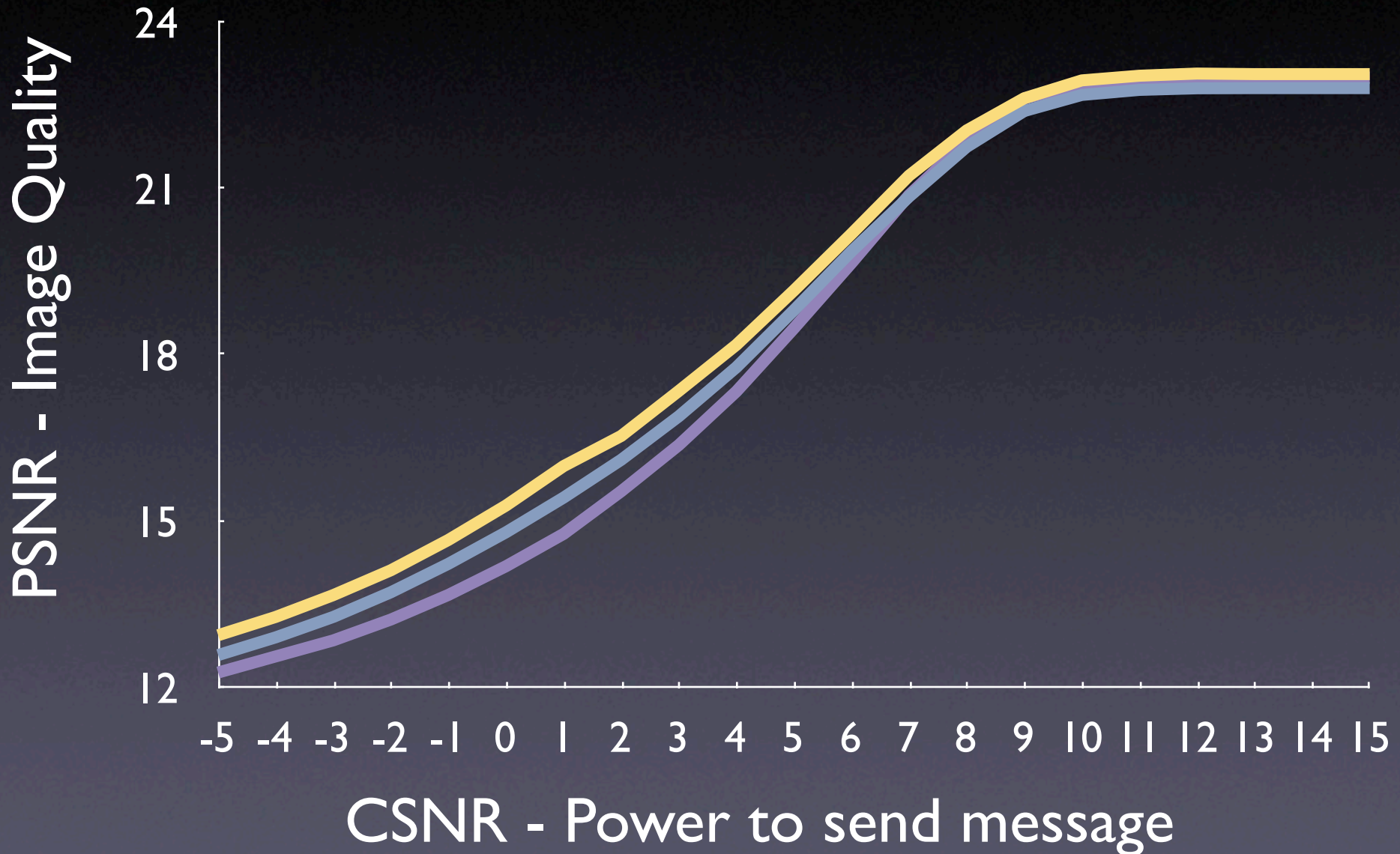


PSNR vs. CSNR for glasses.pgm



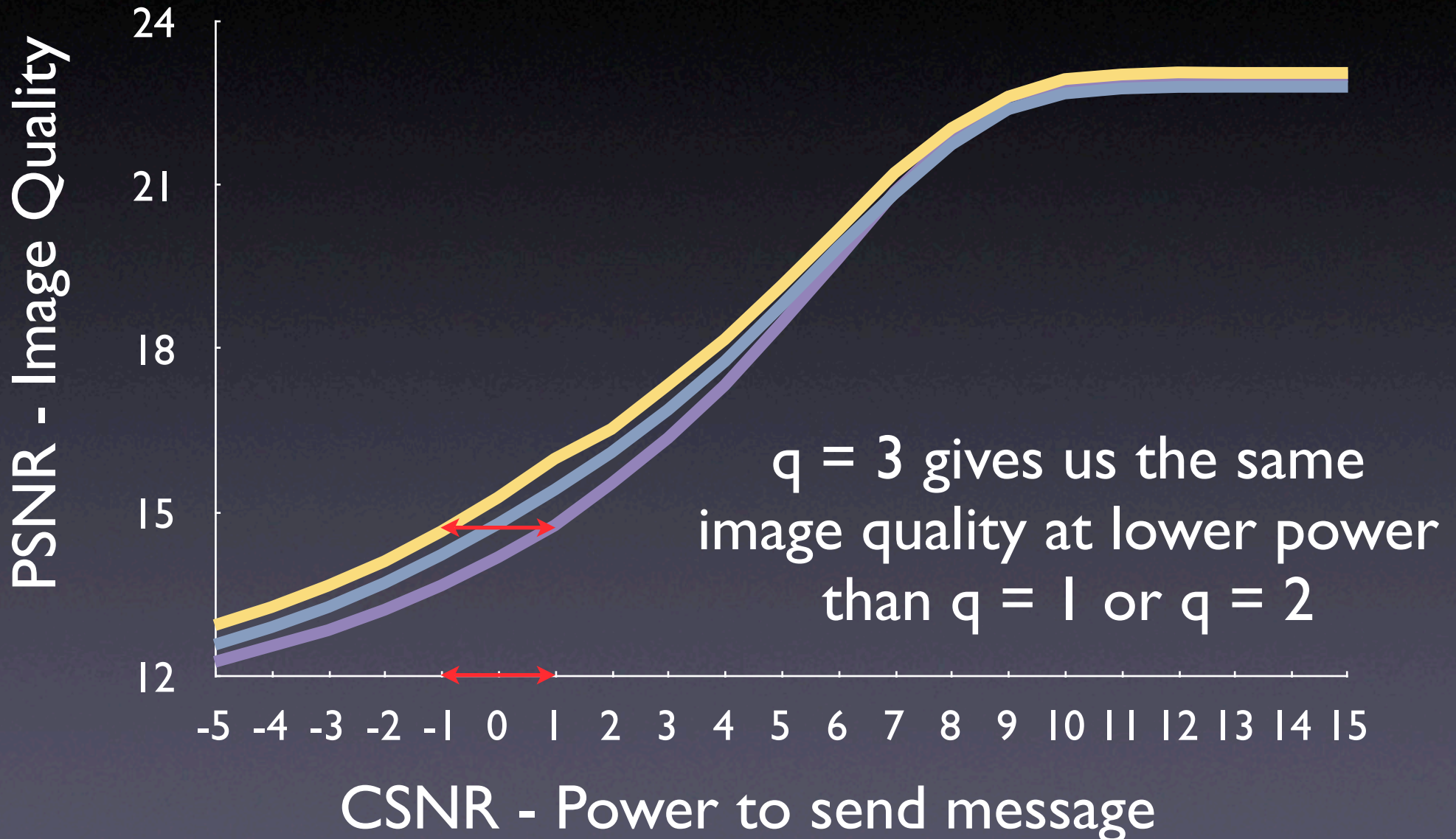
PSNR vs. CSNR for glasses.pgm

— $q=3$ — $q=2$ — $q=1$



PSNR vs. CSNR for glasses.pgm

— $q=3$ — $q=2$ — $q=1$



Conclusion

- With little exception soft decision improved image quality
- For noisy channels, the improvement was visually substantial

Advice for the 06's

- Move to different modulation schemes to achieve higher bit rates
- Try different channels that we didn't have time to try
- Add a comparison of performance versus conventional tandem coding systems

Thanks

- Dr. Firouz Behnamfar
- Dr. Fady Alajaji



Questions?