A Rate-Compatible Sphere-Packing Analysis of Feedback Coding with Limited Retransmissions

Adam Williamson, Tsung-Yi Chen, and Rick Wesel

UCLA Communication Systems Laboratory

arXiv: 1202.1458

July 6, 2012

Variable-Length Feedback with Termination

Variable-Length Feedback with Termination

- **Variable-length feedback with termination** (VLFT) codes [Polyanskiy et al. 2011]:

  $\Rightarrow$ Transmitter sees all channel outputs and tells the receiver when to terminate through a separate control channel.

Variable-Length Feedback with Termination

- **Variable-length feedback with termination** (VLFT) codes [Polyanskiy et al. 2011]:
  - $\Rightarrow$ Transmitter sees all channel outputs and tells the receiver when to terminate through a separate control channel.
  - $\Rightarrow$ Transmission may terminate after each symbol.

Variable-Length Feedback with Termination

- **Variable-length feedback with termination** (VLFT) codes [Polyanskiy et al. 2011]:
  - $\Rightarrow$ Transmitter sees all channel outputs and tells the receiver when to terminate through a separate control channel.
  - $\Rightarrow$ Transmission may terminate after each symbol.
  - $\Rightarrow$ (Rate-compatible) random coding.

Variable-Length Feedback with Termination

- **Variable-length feedback with termination** (VLFT) codes [Polyanskiy et al. 2011]:

  $\Rightarrow$ Transmitter sees all channel outputs and tells the receiver when to terminate through a separate control channel.

  $\Rightarrow$ Transmission may terminate after each symbol.

  $\Rightarrow$ (Rate-compatible) random coding.

  $\Rightarrow$ General results with numerical examples for BSC and BEC.

This Talk

- **This talk:** Still the basic VLFT framework.

  $\Rightarrow$ Transmitter sees all channel outputs and tells the receiver when to terminate through a separate control channel.

This Talk

- **This talk:** Still the basic VLFT framework.
  - ⇒ Transmitter sees all channel outputs and tells the receiver when to terminate through a separate control channel.
  - ⇒ Transmission may only terminate at the end of a “packet”.

This Talk

- **This talk:** Still the basic VLFT framework.
  - $\Rightarrow$ Transmitter sees all channel outputs and tells the receiver when to terminate through a separate control channel.
  - $\Rightarrow$ Transmission may only terminate at the end of a "packet".
  - $\Rightarrow$ Incremental packet lengths will be optimized.

This Talk

- **This talk:** Still the basic VLFT framework.
  - ⇒ Transmitter sees all channel outputs and tells the receiver when to terminate through a separate control channel.
  - ⇒ Transmission may only terminate at the end of a "packet".
  - ⇒ Incremental packet lengths will be optimized.
  - ⇒ Rate-compatible sphere-packing (RCSP) [Chen et al. 2011].

This Talk

- **This talk:** Still the basic VLFT framework.

  $\Rightarrow$ Transmitter sees all channel outputs and tells the receiver when to terminate through a separate control channel.

  $\Rightarrow$ Transmission may only terminate at the end of a "packet".

  $\Rightarrow$ Incremental packet lengths will be optimized.

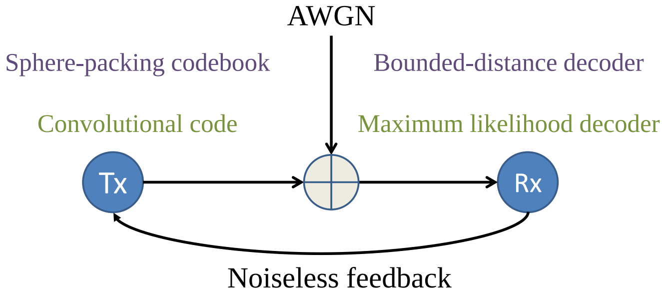
  $\Rightarrow$ Rate-compatible sphere-packing (RCSP) [Chen et al. 2011].

  $\Rightarrow$ Rate-compatible tail-biting convolutional code.

This Talk

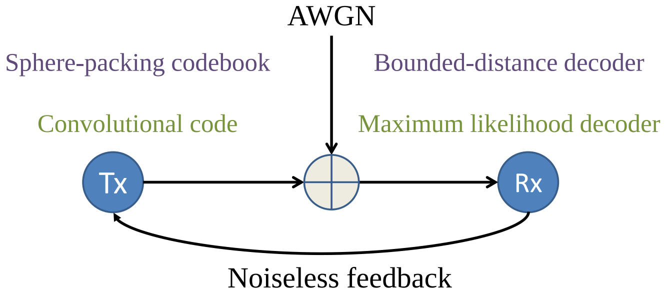
- **This talk:** Still the basic VLFT framework.
  - ⇒ Transmitter sees all channel outputs and tells the receiver when to terminate through a separate control channel.
  - ⇒ Transmission may only terminate at the end of a "packet".
  - ⇒ Incremental packet lengths will be optimized.
  - ⇒ Rate-compatible sphere-packing (RCSP) [Chen et al. 2011].
  - ⇒ Rate-compatible tail-biting convolutional code.
  - ⇒ Focused on the AWGN channel.

Incremental Redundancy Scheme Overview



- Forward channel is **AWGN** with known SNR, η.

Incremental Redundancy Scheme Overview



- Forward channel is **AWGN** with known SNR, η.
- The receiver attempts to decode after each incremental transmission, based on all received symbols.

Transmission Scheme Details (1st transmission)

- $k = \log_2 M$ = information bits.

- 1st transmission:
  - Send I_1 , decode with $N_1 = I_1$.
  - $R_1 = k/N_1$ = initial code rate.

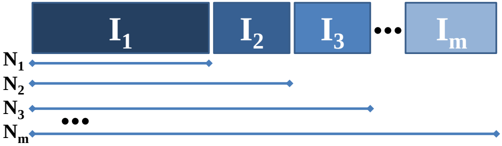
Transmission Scheme Details (2nd transmission)

- $k = \log_2 M =$ information bits.

- 2nd transmission:
  - Send I_2 , decode with $N_2 = I_1 + I_2$.
  - $R_2 = k/N_2$ = code rate.

Transmission Scheme Details (ith transmission)

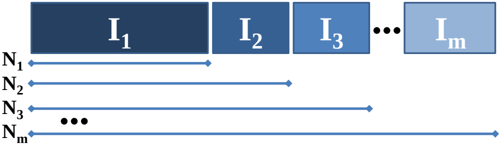
- $k = \log_2 M$ = information bits.



- ith transmission: $(i = 2, \ldots, m)$
  - Send I_i , decode with $N_i = N_{i-1} + I_i$.
  - I_i = incremental step size, N_i = block length at ith transmission.
  - $R_i = k/N_i$ = code rate at ith transmission

Transmission Scheme Details (ith transmission)

- $k = \log_2 M =$ information bits.

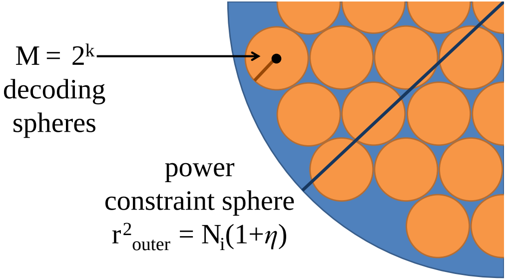


- ith transmission: $(i = 2, \ldots, m)$
  - Send I_i , decode with $N_i = N_{i-1} + I_i$.
  - I_i = incremental step size, $N_i =$ block length at ith transmission.
  - $R_i = k/N_i =$ code rate at ith transmission
- m = maximum number of transmissions (before repetition).

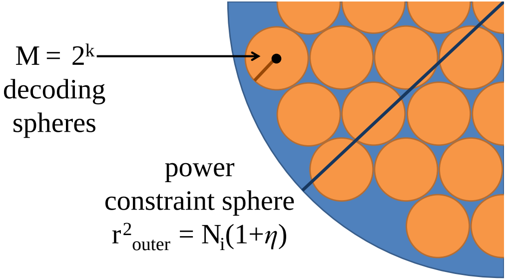
Start Over if Failure after m Transmissions

- If decoding is unsuccessful after m transmissions, start over by sending I_1 bits, then I_2 bits, etc. (similar to ARQ).
- This is a practical limitation.
- Simplifies analysis.

Decoding Error Probability for Sphere-Packing



Decoding Error Probability for Sphere-Packing



- P[error with block length N_i]

$$= \mathrm{P}(\zeta_i) = \mathrm{P}\left(\sum_{\ell=1}^{N_i} z_\ell^2 > r_i^2\right) = 1 - F_{\chi^2_{N_i}}(r_i^2),$$

  - $r_i^2 = \frac{N_i(1+\eta)}{2^{2k/N_i}}$ is the sphere-packing radius (squared),
  - $z_\ell \sim \mathcal{N}(0,1)$ are the noise samples.

Sphere-Packing: Myth or Reality?

- An ideal sphere-packing codebook is mythical.
  - $\Rightarrow$ Upper bound on packing density ϕ in n dimensions: $\phi \;\leq\; (n/e)\, 2^{-n/2}$.

Sphere-Packing: Myth or Reality?

- An ideal sphere-packing codebook is mythical.
  $\Rightarrow$ Upper bound on packing density ϕ in n dimensions:
  $\phi \;\leq\; (n/e)\, 2^{-n/2}$.

- ... but we will see that a convolutional code can achieve sphere-packing performance.

Marginal vs. Joint Decoding Error Probability

- $\mathrm{P}[\text{error with block length } N_i] = \mathrm{P}(\zeta_i)$ **(marginal)**

$$= \mathrm{P}\left(\sum_{\ell=1}^{N_i} z_\ell^2 > r_i^2\right) = 1 - F_{\chi^2_{N_i}}(r_i^2),$$

Marginal vs. Joint Decoding Error Probability

- P[error with block length N_i] $= \mathrm{P}(\zeta_i)$ **(marginal)**

$$= \mathrm{P}\left(\sum_{\ell=1}^{N_i} z_\ell^2 > r_i^2\right) = 1 - F_{\chi^2_{N_i}}(r_i^2),$$

- P[error after j transmissions] $= \mathrm{P}(\zeta_1, \zeta_2, \ldots, \zeta_j)$ **(joint)**

$$= \mathrm{P}\left(\sum_{\ell=1}^{N_1} z_\ell^2 > r_1^2, \sum_{\ell=1}^{N_2} z_\ell^2 > r_2^2, \ldots, \sum_{\ell=1}^{N_j} z_\ell^2 > r_j^2\right)$$

$$= \int_{r_1^2}^{\infty} \int_{r_2^2 - t_1}^{\infty} \cdots \int_{r_{j-1}^2 - \sum_{i=1}^{j-2} t_i}^{\infty} f_{\chi^2_{I_1}}(t_1) \ldots f_{\chi^2_{I_{j-1}}}(t_{j-1}) \times \left(1 - F_{\chi^2_{I_j}}\left(r_j^2 - \sum_{i=1}^{j-1} t_i\right)\right) dt_{j-1} \ldots dt_1.$$

Latency and Throughput (for $m = 1$, the ARQ Case)

- λ = **latency** = expected number of forward channel uses.

Latency and Throughput (for $m = 1$, the ARQ Case)

- λ = **latency** = expected number of forward channel uses.

$$\lambda = I_1 \left(1 + \mathrm{P}(\zeta_1) + \mathrm{P}(\zeta_1)^2 + \mathrm{P}(\zeta_1)^3 + \dots\right)$$
$$= \frac{I_1}{1 - P\left(\zeta_1\right)}$$
$$= \frac{I_1}{F_{\chi^2_{N_1}}\left(r_1^2\right)}$$

Latency and Throughput (for $m = 1$, the ARQ Case)

- λ = **latency** = expected number of forward channel uses.

$$\begin{aligned}\lambda &= I_1 \left(1 + \mathrm{P}(\zeta_1) + \mathrm{P}(\zeta_1)^2 + \mathrm{P}(\zeta_1)^3 + \dots\right) \\ &= \frac{I_1}{1 - P\left(\zeta_1\right)} \\ &= \frac{I_1}{F_{\chi^2_{N_1}}\left(r_1^2\right)}\end{aligned}$$

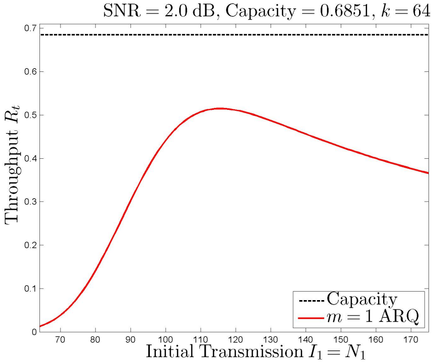
- R_t = **throughput** = k/λ.
- Select I_1 to maximize R_t.

Latency and Throughput (for $m = 1$, the ARQ Case)

- λ = **latency** = expected number of forward channel uses.

$$\lambda = I_1 \left(1 + \mathrm{P}(\zeta_1) + \mathrm{P}(\zeta_1)^2 + \mathrm{P}(\zeta_1)^3 + \dots\right)$$
$$= \frac{I_1}{1 - P(\zeta_1)}$$
$$= \frac{I_1}{F_{\chi^2_{N_1}}(r_1^2)}$$

- R_t = **throughput** = k/λ.
- Select I_1 to maximize R_t.



What About $m > 1$?

- **Latency**

$$\lambda = \frac{I_1 + \sum_{i=2}^{m} I_i P\left(\bigcap_{j=1}^{i-1} \zeta_j\right)}{1 - P\left(\bigcap_{j=1}^{m} \zeta_j\right)}$$

What About $m > 1$?

- **Latency**

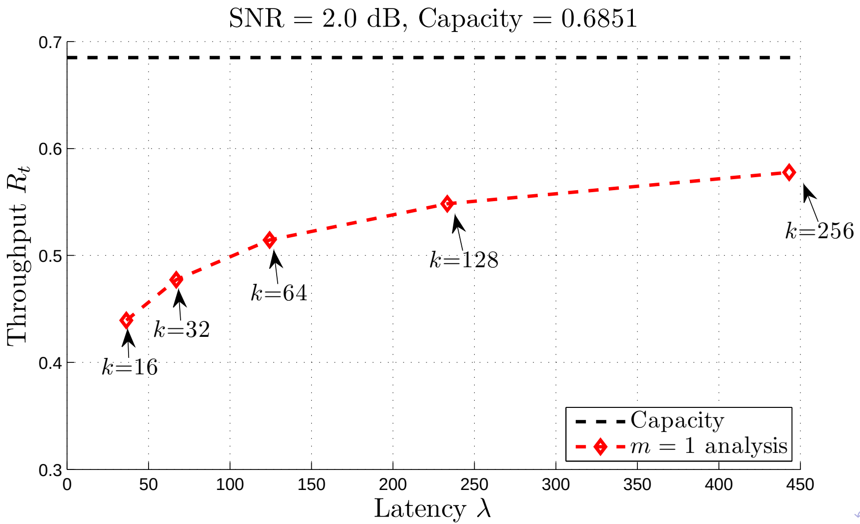
$$\lambda = \frac{I_1 + \sum_{i=2}^{m} I_i P\left(\bigcap_{j=1}^{i-1} \zeta_j\right)}{1 - P\left(\bigcap_{j=1}^{m} \zeta_j\right)}$$

- **Throughput**

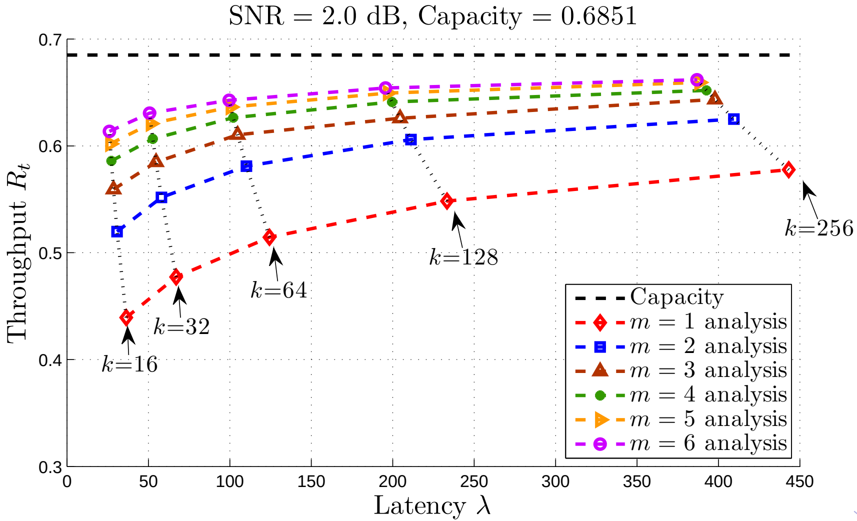
$$R_t = k/\lambda$$

- Select $\{I_1, I_2, \ldots, I_m\}$ to maximize R_t.

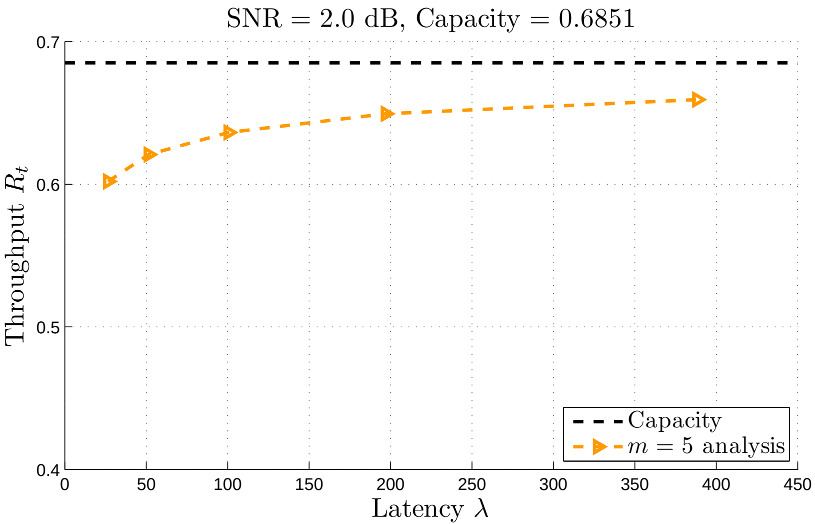
RCSP: Latency vs. Throughput for $m = 1$ (ARQ) Using Optimal Step Size I_1



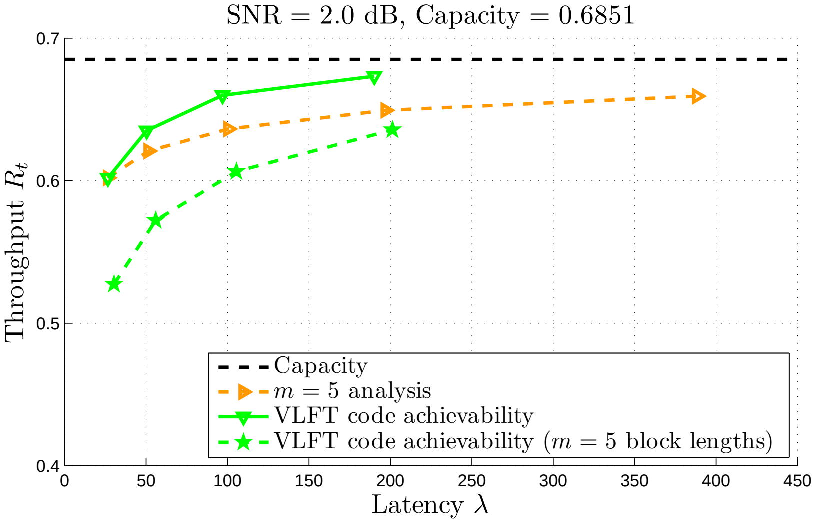
RCSP: Latency vs. Throughput for $m = 1$ to $m = 6$, Using Optimal Step Sizes I_i



RCSP: Latency vs. Throughput for $m = 5$, Using Optimal Step Sizes I_i



Comparison with [Polyanskiy et al. 2011]



Convolutional Code Simulations for $m = 5$

- Mother codes are rate $1/3$, 64-state and 1024-state convolutional codes from [Lin and Costello 2004].

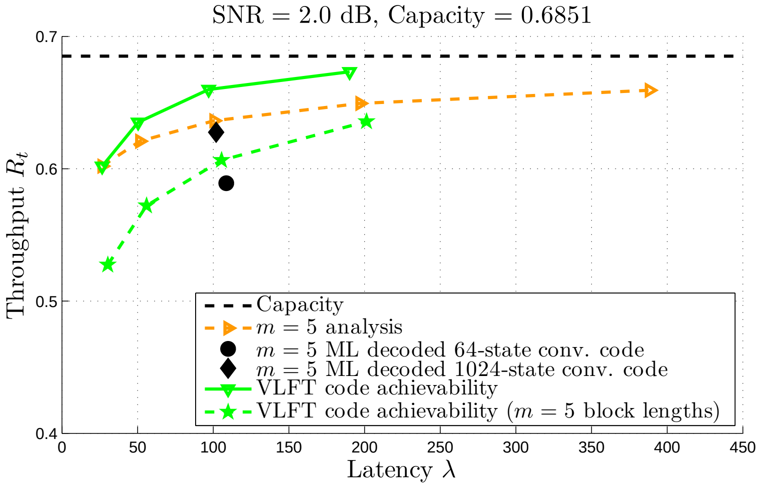
Convolutional Code Simulations for $m = 5$

- Mother codes are rate $1/3$, 64-state and 1024-state convolutional codes from [Lin and Costello 2004].
- Use transmission lengths $\{I_1^m\}$ identified in RCSP optimization for $m = 5$.
- High-rate codes obtained by pseudo-random puncturing of mother codes.

Convolutional Code Simulations for $m = 5$

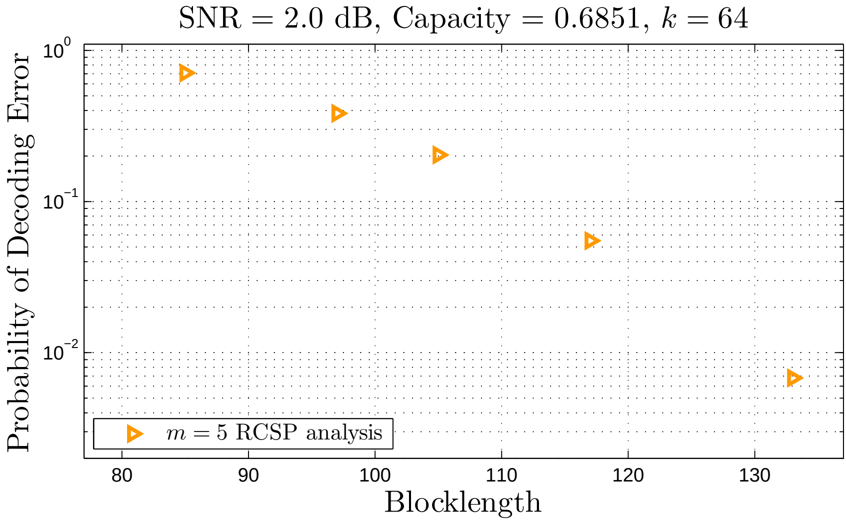
- Mother codes are rate $1/3$, 64-state and 1024-state convolutional codes from [Lin and Costello 2004].
- Use transmission lengths $\{I_1^m\}$ identified in RCSP optimization for $m = 5$.
- High-rate codes obtained by pseudo-random puncturing of mother codes.
- **Maximum likelihood (ML)** decoding.
  - ML decoding regions completely fill the power constraint sphere.
- Tail-biting implementations used for throughput efficiency.

Convolutional Code Achievability, $m = 5$

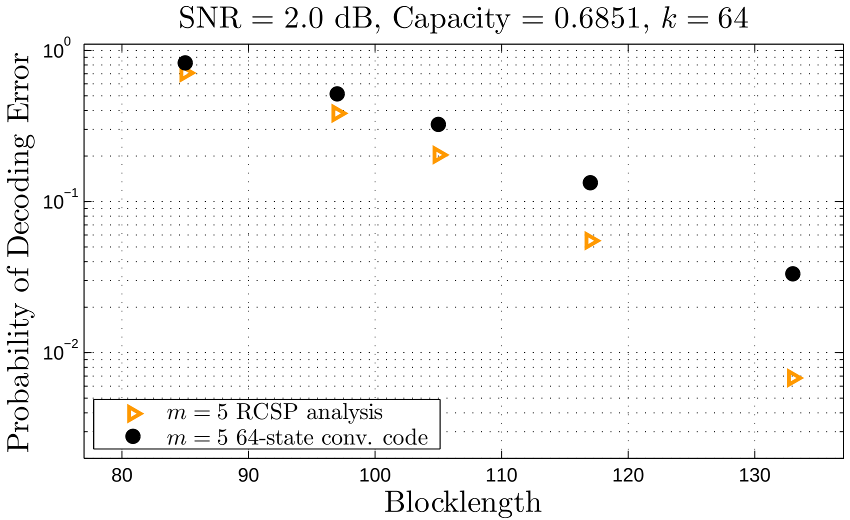


- 90% of AWGN capacity in ~100 symbols.

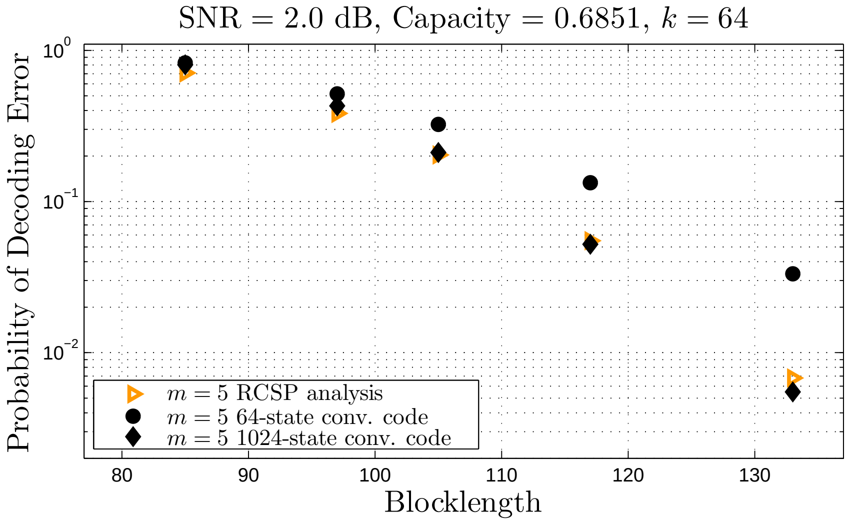
Decoding Error Trajectory



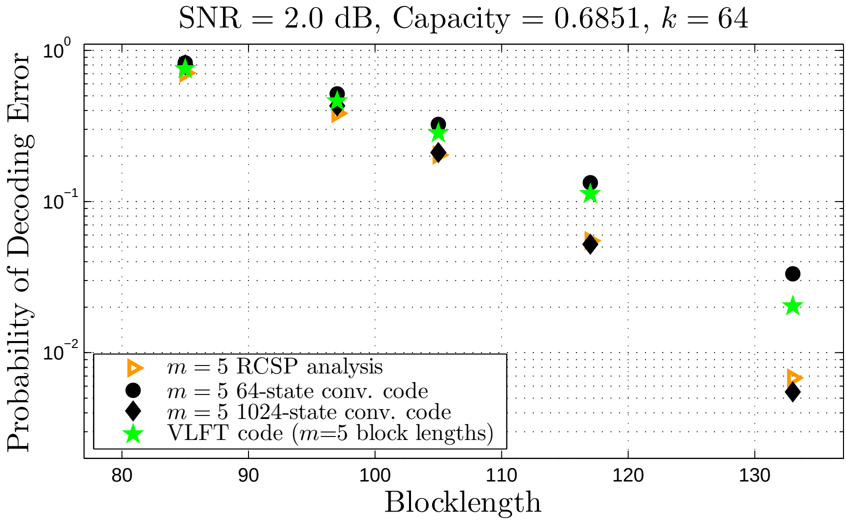
Decoding Error Trajectory



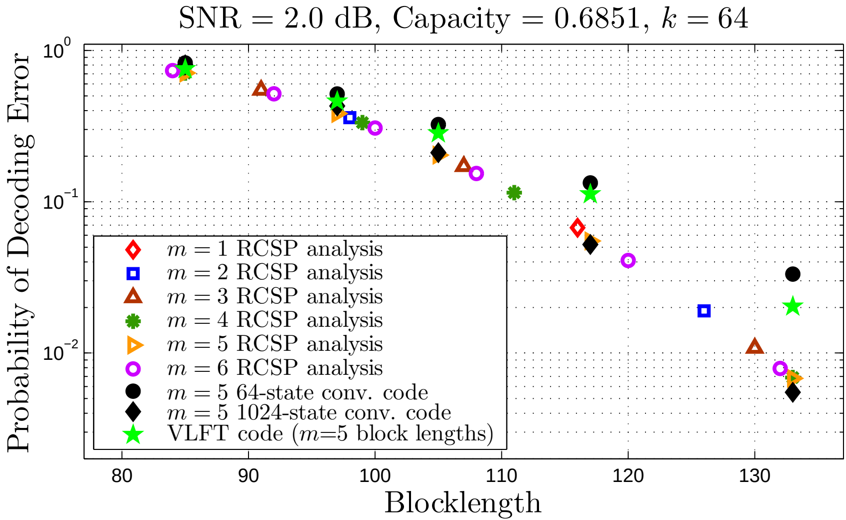
Decoding Error Trajectory



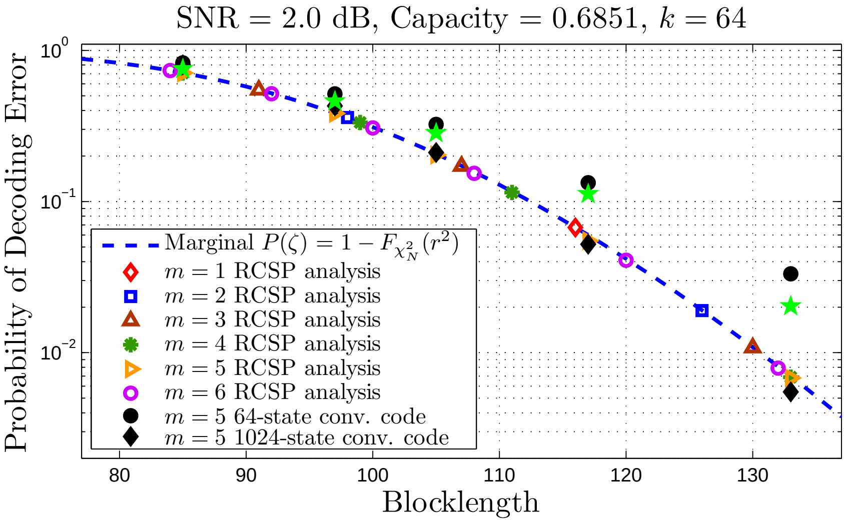
Decoding Error Trajectory



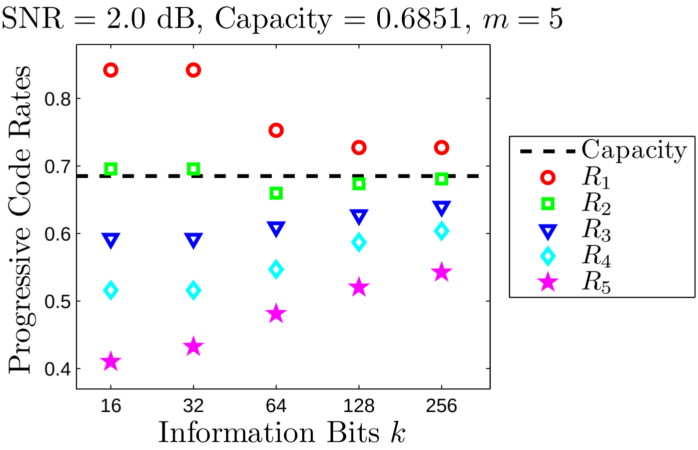
Decoding Error Trajectory



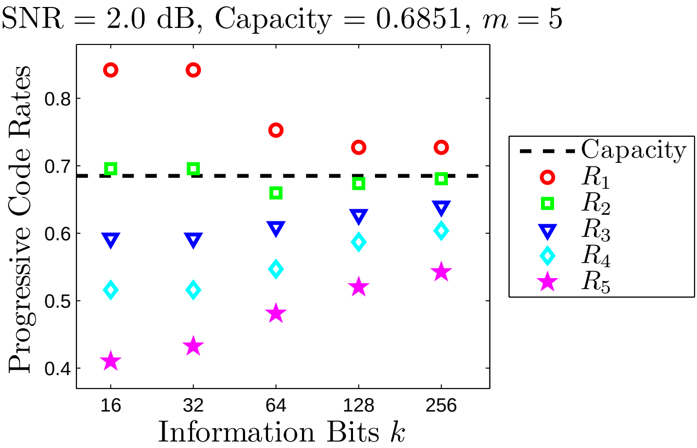
Marginal Chi-Square: A Design Objective



Optimal Rates



Optimal Rates



- $R_1 > C$

Concluding Thoughts

- Feedback improves achievable rate for finite block lengths.

Concluding Thoughts

- Feedback improves achievable rate for finite block lengths.
    - Feedback after every bit is best.

Concluding Thoughts

- Feedback improves achievable rate for finite block lengths.
  - Feedback after every bit is best.
  - When transmissions must be grouped, pick the sizes wisely.

Concluding Thoughts

- Feedback improves achievable rate for finite block lengths.
  - Feedback after every bit is best.
  - When transmissions must be grouped, pick the sizes wisely.
- Find good codes by matching RCSP error trajectories.

Concluding Thoughts

- Feedback improves achievable rate for finite block lengths.
  - Feedback after every bit is best.
  - When transmissions must be grouped, pick the sizes wisely.
- Find good codes by matching RCSP error trajectories.
- Questions?