Stabilization using noisy and noiseless feedback

Anant Sahai
based in part on joint work with student: Hari Palaiyanur

Wireless Foundations
Department of Electrical Engineering and Computer Sciences
University of California at Berkeley

Extended Version of Talk Given at MTNS 2006
Outline

1 Motivation and introduction
   ▶ Problem setup
   ▶ Review of key considerations
2 Main result illustrated
   ▶ What is wrong with random coding
   ▶ The role of noiseless feedback
3 Conclusion
A simple distributed control problem

\[ X_{t+1} = \lambda X_t + U_t + W_t \]

- Unstable $\lambda > 1$, bounded initial condition and disturbance $W$. 

Anant Sahai (UC Berkeley)
A simple distributed control problem

\[ X_{t+1} = \lambda X_t + U_t + W_t \]

- Unstable \( \lambda > 1 \), bounded initial condition and disturbance \( W \).
- Goal: Performance = \( \sup_{t>0} E[\|X_t\|_\eta] \leq K \) for some target \( K < \infty \).
Fortified channels

Noisy forward channel uses

"Fortification" noiseless forward channel uses

Some mix of noisy and noiseless channels
Fortified channels

Noisy forward channel uses

”Fortification” noiseless forward channel uses

- Some mix of noisy and noiseless channels
- Is it all or nothing?
Review: Entirely noiseless channel

Window known to contain $X_t$

will grow by factor of $\lambda > 1$

Sending $R$ bits, cut window by a factor of $2^{-R}$

Encode which control $U_t$ to apply

grows by $\Omega/2$ on each side

giving a new window for $X_{t+1}$

As long as $R > \log_2 \lambda$, we can have $\Delta$ stay bounded forever.
Use entropy and mutual information
Use entropy and mutual information
  ▶ Tatikonda’s insight: directed mutual information captures causality
Review: The separation-principle oriented program

- Use entropy and mutual information
  - Tatikonda’s insight: directed mutual information captures causality
- Write out entropic inequalities
  - Key Inequality: Directed data-processing inequality
Review: The separation-principle oriented program

- Use entropy and mutual information
  - Tatikonda’s insight: directed mutual information captures causality
- Write out entropic inequalities
  - Key Inequality: Directed data-processing inequality
- Set up a mapping between bits and performance
  - You probably don’t care about the entropy of the state.
Use entropy and mutual information
  ▶ Tatikonda’s insight: directed mutual information captures causality

Write out entropic inequalities
  ▶ Key Inequality: Directed data-processing inequality

Set up a mapping between bits and performance
  ▶ You probably don’t care about the entropy of the state.
  ▶ Lower bound performance assuming nested information
  ▶ Equivalent to estimation
Review: The separation-principle oriented program

- Use entropy and mutual information
  - Tatikonda’s insight: directed mutual information captures causality

- Write out entropic inequalities
  - Key Inequality: Directed data-processing inequality

- Set up a mapping between bits and performance
  - You probably don’t care about the entropy of the state.
  - Lower bound performance assuming nested information
  - Equivalent to estimation
  - Rate-distortion theory can be developed
Review: The separation-principle oriented program

- Use entropy and mutual information
  - Tatikonda’s insight: directed mutual information captures causality
- Write out entropic inequalities
  - Key Inequality: Directed data-processing inequality
- Set up a mapping between bits and performance
  - You probably don’t care about the entropy of the state.
  - Lower bound performance assuming nested information
  - Equivalent to estimation
  - Rate-distortion theory can be developed
- Get tight upper bounds and architectures?
Review: The rate-distortion part

Graph showing:
- "Sequential" Rate−distortion (obeys causality)
- Rate−distortion curve (non−causal)
- Stable counterpart (non−causal)

Y-axis: Squared error distortion
X-axis: Rate (in bits)
Consider a system with

- $\lambda = 2$ for the dynamics
- Real packet-drop channel ($C = \infty$)

$$Z_t = \begin{cases} 
Y_t & \text{with Probability } \frac{1}{2} \\
0 & \text{with Probability } \frac{1}{2}
\end{cases}$$
Consider a system with

- $\lambda = 2$ for the dynamics
- Real packet-drop channel ($C = \infty$)

$$Z_t = \begin{cases} 
Y_t & \text{with Probability } \frac{1}{2} \\
0 & \text{with Probability } \frac{1}{2}
\end{cases}$$

No other constraints, so design is obvious: $Y_t = X_t$ and $U_t = -\lambda Z_t$
Review: How bad can entropic bounds be?

- Consider a system with
  - $\lambda = 2$ for the dynamics
  - Real packet-drop channel ($C = \infty$)

$$Z_t = \begin{cases} 
  Y_t & \text{with Probability } \frac{1}{2} \\
  0 & \text{with Probability } \frac{1}{2}
\end{cases}$$

- No other constraints, so design is obvious: $Y_t = X_t$ and $U_t = -\lambda Z_t$

$$X_{t+1} = \begin{cases} 
  W_t & \text{with Probability } \frac{1}{2} \\
  2X_t + W_t & \text{with Probability } \frac{1}{2}
\end{cases}$$
Consider a system with

- \( \lambda = 2 \) for the dynamics
- Real packet-drop channel \((C = \infty)\)

\[
Z_t = \begin{cases} 
Y_t & \text{with Probability } \frac{1}{2} \\
0 & \text{with Probability } \frac{1}{2}
\end{cases}
\]

No other constraints, so design is obvious: \( Y_t = X_t \) and \( U_t = -\lambda Z_t \)

\[
X_{t+1} = \begin{cases} 
W_t & \text{with Probability } \frac{1}{2} \\
2X_t + W_t & \text{with Probability } \frac{1}{2}
\end{cases}
\]

Under stochastic disturbances, the variance of the state is asymptotically infinite. *(St. Petersburg Lottery Style)*
Review: Delay-universal \textit{(anytime)} communication

\[
\begin{align*}
B_1 & \quad B_2 & \quad B_3 & \quad B_4 & \quad B_5 & \quad B_6 & \quad B_7 & \quad B_8 & \quad B_9 & \quad B_{10} & \quad B_{11} & \quad B_{12} & \quad B_{13} \\
\downarrow & & & & & & & & & & & & & \\
Y_1 & \quad Y_2 & \quad Y_3 & \quad Y_4 & \quad Y_5 & \quad Y_6 & \quad Y_7 & \quad Y_8 & \quad Y_9 & \quad Y_{10} & \quad Y_{11} & \quad Y_{12} & \quad Y_{13} & \quad Y_{14} & \quad Y_{15} & \quad Y_{16} & \quad Y_{17} & \quad Y_{18} & \quad Y_{19} & \quad Y_{20} & \quad Y_{21} & \quad Y_{22} & \quad Y_{23} & \quad Y_{24} & \quad Y_{25} & \quad Y_{26} \\
\downarrow & & & & & & & & & & & & & \\
Z_1 & \quad Z_2 & \quad Z_3 & \quad Z_4 & \quad Z_5 & \quad Z_6 & \quad Z_7 & \quad Z_8 & \quad Z_9 & \quad Z_{10} & \quad Z_{11} & \quad Z_{12} & \quad Z_{13} & \quad Z_{14} & \quad Z_{15} & \quad Z_{16} & \quad Z_{17} & \quad Z_{18} & \quad Z_{19} & \quad Z_{20} & \quad Z_{21} & \quad Z_{22} & \quad Z_{23} & \quad Z_{24} & \quad Z_{25} & \quad Z_{26} \\
\downarrow & & & & & & & & & & & & & \\
\hat{B}_1 & \quad \hat{B}_2 & \quad \hat{B}_3 & \quad \hat{B}_4 & \quad \hat{B}_5 & \quad \hat{B}_6 & \quad \hat{B}_7 & \quad \hat{B}_8 & \quad \hat{B}_9 \\
\downarrow & & & & & & & & & & & & & \\
\text{fixed delay } d = 7
\end{align*}
\]
Fixed-delay reliability $\alpha$ is achievable if there exists a sequence of encoder/decoder pairs with increasing end-to-end delays $d_j$ such that

$$\lim_{j \to \infty} \frac{-1}{d_j} \ln P(B_i \neq \hat{B}_i^j) = \alpha.$$
\( \alpha \) is achievable delay-universally or in an anytime fashion if a single encoder works for all sufficiently large delays \( d \).
The anytime capacity $C_{\text{any}}(\alpha)$ is the supremal rate at which reliability $\alpha$ is achievable in a delay-universal way.
Review: Separation theorem for scalar control

*Necessity:* If a scalar system with parameter $\lambda > 1$ can be stabilized with finite $\eta$-moment across a noisy channel, then the channel with noiseless feedback must have

$$C_{\text{any}}(\eta \ln \lambda) \geq \ln \lambda$$

In general: If $P(|X| > m) < f(m)$, then $\exists K : P_{\text{error}}(d) < f(K\lambda^d)$
Review: Separation theorem for scalar control

Necessity: If a scalar system with parameter $\lambda > 1$ can be stabilized with finite $\eta$-moment across a noisy channel, then the channel with noiseless feedback must have

$$C_{\text{any}}(\eta \ln \lambda) \geq \ln \lambda$$

In general: If $P(|X| > m) < f(m)$, then $\exists K : P_{\text{error}}(d) < f(K\lambda^d)$

Sufficiency: If there is an $\alpha > \eta \ln \lambda$ for which the channel with noiseless feedback has

$$C_{\text{any}}(\alpha) > \ln \lambda$$

then the scalar system with parameter $\lambda \geq 1$ with a bounded disturbance can be stabilized across the noisy channel with finite $\eta$-moment.
Review: Separation theorem for scalar control

**Necessity:** If a scalar system with parameter $\lambda > 1$ can be stabilized with finite $\eta$-moment across a noisy channel, then the **channel with noiseless feedback** must have

$$C_{\text{any}}(\eta \ln \lambda) \geq \ln \lambda$$

In general: If $P(|X| > m) < f(m)$, then $\exists K : P_{\text{error}}(d) < f(K \lambda^d)$

**Sufficiency:** If there is an $\alpha > \eta \ln \lambda$ for which the **channel with noiseless feedback** has

$$C_{\text{any}}(\alpha) > \ln \lambda$$

then the scalar system with parameter $\lambda \geq 1$ with a bounded disturbance can be stabilized across the noisy channel with finite $\eta$-moment.

Captures stabilization only.
Some easy implications

- If we want $P(|X_t| > m) \leq f(m) = 0$ for some finite $m$, we require zero-error reliability across the channel. Also required (for DMCs) if we want the controller to be finite memory.
Some easy implications

- If we want $P(|X_t| > m) \leq f(m) = 0$ for some finite $m$, we require zero-error reliability across the channel. Also required (for DMCs) if we want the controller to be finite memory.

- For generic DMCs, anytime reliability with feedback is upper-bounded:

  $$f(K\lambda^d) \geq \zeta^d$$
  $$f(m) \geq K'm^{-\frac{\log_2 \frac{1}{\zeta}}{\log_2 \lambda}}$$

A controlled state can have at best a power-law tail.
Some easy implications

- If we want $P(|X_t| > m) \leq f(m) = 0$ for some finite $m$, we require zero-error reliability across the channel. Also required (for DMCs) if we want the controller to be finite memory.

- For generic DMCs, anytime reliability with feedback is upper-bounded:

$$f(K\lambda^d) \geq \zeta^d$$

$$f(m) \geq K'm^{-\frac{\log_2 \frac{1}{\zeta}}{\log_2 \lambda}}$$

A controlled state can have at best a power-law tail.

- If we just want $\lim_{m \to \infty} f(m) = 0$, then just Shannon capacity $> \log_2 \lambda$ is required for DMCs.
Some easy implications

- If we want $P(|X_t| > m) \leq f(m) = 0$ for some finite $m$, we require zero-error reliability across the channel. Also required (for DMCs) if we want the controller to be finite memory.
- For generic DMCs, anytime reliability with feedback is upper-bounded:
  \[
  f(K \lambda^d) \geq \zeta^d
  \]
  \[
  f(m) \geq K'm^{-\frac{1}{\log_2 \lambda}}
  \]

A controlled state can have at best a power-law tail.

- If we just want $\lim_{m \to \infty} f(m) = 0$, then just Shannon capacity $> \log_2 \lambda$ is required for DMCs.
- Almost-sure stabilization for $W_t = 0$ follows by simple time-varying transformation.
Outline

1 Motivation and introduction
   ▶ Problem setup
   ▶ Review of key considerations

2 Main result illustrated
   ▶ What is wrong with random coding
   ▶ The role of noiseless feedback

3 Conclusion
Implications for scalar moment stabilization

![Graph showing error exponent vs. rate in nats for noisy and noiseless signals.]

Rate (in nats)

Error Exponent (base e)

Anant Sahai (UC Berkeley)
Implications for scalar moment stabilization

![Graph showing the relationship between moments stabilized and open-loop unstable gain.]

- moments stabilized
- open-loop unstable gain

Anant Sahai (UC Berkeley)

Noisy plus Noiseless

July 25, 2006

17 / 30
Random coding bound is relatively easy to achieve

- Randomly label the uniformly quantized state!
Random coding bound is relatively easy to achieve

- Randomly label the uniformly quantized state!
- Stable system state “renews” itself.

\[ R = \log_2 3 \]
Random coding bound is relatively easy to achieve

- Randomly label the uniformly quantized state!
- Stable system state “renews” itself.
- It diverges locally whenever the channel misbehaves.

\[ R = \log_2 3 \]
Random coding bound is relatively easy to achieve

- Randomly label the uniformly quantized state!
- Stable system state “renews” itself.
- It diverges locally whenever the channel misbehaves.
- Semi-reasonable implementation complexity.

\[
\begin{align*}
R &= \log_2 3 \\
\Delta &= \text{some label here}
\end{align*}
\]
Controller and Computations

- All “false” disjoint paths through the trellis are pairwise independent with the true path.
Controller and Computations

- All “false” disjoint paths through the trellis are pairwise independent with the true path.
- Bound the number of distinct paths by assuming no remerging.
Controller and Computations

- All “false” disjoint paths through the trellis are pairwise independent with the true path.
- Bound the number of distinct paths by assuming no remerging.
- Gallager’s $E_r(R_{branch})$ emerges as the governing exponent.
Controller and Computations

- All “false” disjoint paths through the trellis are pairwise independent with the true path.
- Bound the number of distinct paths by assuming no remerging.
- Gallager’s $E_r(R_{branch})$ emerges as the governing exponent.
- Apply the control based on current ML state.
Controller and Computations

- All “false” disjoint paths through the trellis are pairwise independent with the true path.
- Bound the number of distinct paths by assuming no remerging.
- Gallager’s $E_r(R_{branch})$ emerges as the governing exponent.
- Apply the control based on current ML state.
- **Computational nightmare:** effort grows exponentially with time.
Controller and Computations

- All “false” disjoint paths through the trellis are pairwise independent with the true path.
- Bound the number of distinct paths by assuming no remerging.
- Gallager’s $E_r(R_{\text{branch}})$ emerges as the governing exponent.
- Apply the control based on current ML state.
- **Computational nightmare:** effort grows exponentially with time.
- Use “Stack-based” greedy search algorithm instead.
  - Log likelihoods are additive.
  - The score of a path is a random walk with drift.
  - Bias it so that the true path goes up and false ones down.
Controller and Computations

- All “false” disjoint paths through the trellis are pairwise independent with the true path.
- Bound the number of distinct paths by assuming no remerging.
- Gallager’s $E_r(R_{branch})$ emerges as the governing exponent.
- Apply the control based on current ML state.

**Computational nightmare:** effort grows exponentially with time.

- Use “Stack-based” greedy search algorithm instead.
  - Log likelihoods are additive.
  - The score of a path is a random walk with drift.
  - Bias it so that the true path goes up and false ones down.

- Classical results tell us that with appropriate bias, achieve $E_r(R_{branch})$ for error probability and hence power-law in state.
Controller and Computations

- All “false” disjoint paths through the trellis are pairwise independent with the true path.
- Bound the number of distinct paths by assuming no remerging.
- Gallager’s $E_r(R_{branch})$ emerges as the governing exponent.
- Apply the control based on current ML state.

Computational nightmare: effort grows exponentially with time.

Use “Stack-based” greedy search algorithm instead.
  - Log likelihoods are additive.
  - The score of a path is a random walk with drift.
  - Bias it so that the true path goes up and false ones down.

Classical results tell us that with appropriate bias, achieve $E_r(R_{branch})$ for error probability and hence power-law in state.

At the cost of only finite expected computation.
Catch up “all-at-once” phenomenon

Simulation Parameters:
\[ \lambda = 1.1 \]
\[ \varepsilon = 0.05 \]
\[ \Omega = 2.0 \]
\[ \Delta = 5000.0 \]
Bias = 0.55
T = 10
100,000 Blocks
17 seconds to run

Rate = 0.317
Capacity = 0.71
Although we are doing better than exponential growth, we still have power laws on both sides.

What if we needed a finite speed computer in the controller?
Although we are doing better than exponential growth, we still have power laws on both sides.

What if we needed a finite speed computer in the controller?

Bad news:
- Assume 0 control applied if we can not decode yet.
Truth in advertising: computation revisited

- Although we are doing better than exponential growth, we still have power laws on both sides.
- What if we needed a finite speed computer in the controller?
- Bad news:
  - Assume 0 control applied if we can not decode yet.
  - Power law for comp. implies power low for waiting.
Truth in advertising: computation revisited

Although we are doing better than exponential growth, we still have power laws on both sides.

What if we needed a finite speed computer in the controller?

Bad news:
- Assume 0 control applied if we can not decode yet.
- Power law for comp. implies power low for waiting.
- Exponentially rare doubly exponentially bad states!
How to hit the higher bound?
How to hit the higher bound?
Noiseless channel can enable event-based sampling

Noisy forward channel uses

"Fortification" noiseless forward channel uses

Need to allow for gradual progress during bad periods.
Noiseless channel can enable event-based sampling

Noisy forward channel uses

"Fortification" noiseless forward channel uses

- Need to allow for gradual progress during bad periods.
- Use the noiseless channel for supervisory information:
Noiseless channel can enable event-based sampling

Need to allow for gradual progress during bad periods.

Use the noiseless channel for supervisory information:
  - Have the observer do event-based “sampling” of the state.
Noiseless channel can enable event-based sampling

Noisy forward channel uses

”Fortification” noiseless forward channel uses

Need to allow for gradual progress during bad periods.

Use the noiseless channel for supervisory information:

- Have the observer do event-based “sampling” of the state.
- “Quantization net” grows as needed, but has only $e^{nR}$ boxes.
Noiseless channel can enable event-based sampling

Noisy forward channel uses

"Fortification" noiseless forward channel uses

- Need to allow for gradual progress during bad periods.
- Use the noiseless channel for supervisory information:
  - Have the observer do event-based “sampling” of the state.
  - “Quantization net” grows as needed, but has only $e^{nR}$ boxes.
  - Noiseless channel tells controller when it has “resampled.”

Outer net to quantize and encode the state

Inner catchment area to resample the state
Noiseless channel can enable event-based sampling

Need to allow for gradual progress during bad periods.
Use the noiseless channel for supervisory information:
  - Have the observer do event-based “sampling” of the state.
  - “Quantization net” grows as needed, but has only $e^{nR}$ boxes.
  - Noiseless channel tells controller when it has “resampled.”

Use the noisy channel for variable-length block-coding.
Why gradual progress is better: intuition
Why this works: proof strategy

- Lift problem by using large $nR$
Why this works: proof strategy

- Lift problem by using large $nR$
  - Very few noiseless channel uses required
Why this works: proof strategy

Lift problem by using large $nR$

- Very few noiseless channel uses required
- Stopping time for variable-length channel is like $n + \tilde{T}$, where $\tilde{T}$ is geometric $\exp(-E_0(\rho))$. 
Lift problem by using large $nR$

- Very few noiseless channel uses required
- Stopping time for variable-length channel is like $n + \tilde{T}$, where $\tilde{T}$ is geometric $\exp(-E_0(\rho))$.
- Interpret with $\ln \lambda < R = \frac{E_0(\rho)}{\rho} < \frac{E_0(\eta+\epsilon)}{\eta+\epsilon}$
Lift problem by using large $nR$

- Very few noiseless channel uses required
- Stopping time for variable-length channel is like $n + \tilde{T}$, where $\tilde{T}$ is geometric $\exp(-E_0(\rho))$.
- Interpret with $\ln \lambda < R = \frac{E_0(\rho)}{\rho} < \frac{E_0(\eta+\epsilon)}{\eta+\epsilon}$

Behaves like a “virtual” packet-erase channel.
Why this works: proof strategy

- Lift problem by using large $nR$
  - Very few noiseless channel uses required
  - Stopping time for variable-length channel is like $n + \tilde{T}$, where $\tilde{T}$ is geometric $\exp(-E_0(\rho))$.
  - Interpret with $\ln \lambda < R = \frac{E_0(\rho)}{\rho} < \frac{E_0(\eta + \epsilon)}{\eta + \epsilon}$

- Behaves like a “virtual” packet-erasure channel.
  - Each packet carries $n(R - \ln \lambda)$ nats.
Lift problem by using large $nR$
  
  - Very few noiseless channel uses required
  - Stopping time for variable-length channel is like $n + \tilde{T}$, where $\tilde{T}$ is geometric $\exp(-E_0(\rho))$.
  - Interpret with $\ln \lambda < R = \frac{E_0(\rho)}{\rho} < \frac{E_0(\eta+\epsilon)}{\eta+\epsilon}$

Behaves like a “virtual” packet-erasure channel.
  
  - Each packet carries $n(R - \ln \lambda)$ nats.
  - Disturbances grow by factor $O(\lambda^n)$
Why this works: proof strategy

- Lift problem by using large $nR$
  - Very few noiseless channel uses required
  - Stopping time for variable-length channel is like $n + \tilde{T}$, where $\tilde{T}$ is geometric $\exp(-E_0(\rho))$.
  - Interpret with $\ln \lambda < R = \frac{E_0(\rho)}{\rho} < \frac{E_0(\eta+\epsilon)}{\eta+\epsilon}$

- Behaves like a “virtual” packet-erasure channel.
  - Each packet carries $n(R - \ln \lambda)$ nats.
  - Disturbances grow by factor $O(\lambda^n)$
  - Erasure probability $\exp(-E_0(\rho))$
Outline

1 Motivation and introduction
   - Problem setup
   - Review of key considerations

2 Main result illustrated
   - What is wrong with random coding
   - The role of noiseless feedback

3 Conclusion
Straight random coding is a bad idea in control applications.
Conclusion

- Straight random coding is a bad idea in control applications.
- Better to use a few noiseless bits for supervising.
Conclusion

- Straight random coding is a bad idea in control applications.
- Better to use a few noiseless bits for supervising.
- Event-based sampling and variable-length communication is very helpful in the large deviations sense.
Conclusion

- Straight random coding is a bad idea in control applications.
- Better to use a few noiseless bits for supervising.
- Event-based sampling and variable-length communication is very helpful in the large deviations sense.
- Implementation complexity also realizable.
Conclusion

- Straight random coding is a bad idea in control applications.
- Better to use a few noiseless bits for supervising.
- Event-based sampling and variable-length communication is very helpful in the large deviations sense.
- Implementation complexity also realizable.
- Can merge the two ideas — no need to use the big net every time:
  - Most of the time: use straight random coding
Conclusion

- Straight random coding is a bad idea in control applications.
- Better to use a few noiseless bits for supervising.
- Event-based sampling and variable-length communication is very helpful in the large deviations sense.
- Implementation complexity also realizable.
- Can merge the two ideas — no need to use the big net every time:
  - Most of the time: use straight random coding
  - Use noisefree bits to switch to nonuniform emergency mode.
Conclusion

- Straight random coding is a bad idea in control applications.
- Better to use a few noiseless bits for supervising.
- Event-based sampling and variable-length communication is very helpful in the large deviations sense.
- Implementation complexity also realizable.
- Can merge the two ideas — no need to use the big net every time:
  - Most of the time: use straight random coding
  - Use noise-free bits to switch to nonuniform emergency mode.
  - Switch back once uncertainty has been contained.
Synthesizing “noisefree” channel uses
Synthesizing “noisefree” channel uses

Anant Sahai (UC Berkeley)

July 25, 2006