Select Lab

Auton Lab

A Constraint Generation Approach to Learning Stable Linear Dynamical Systems

Sajid M. Siddiqi
Byron Boots
Geoffrey J. Gordon

Carnegie Mellon University

NIPS 2007
poster W22

steam

Application: Dynamic Textures[1]

steam

river

fountain

- Videos of moving scenes that exhibit **stationarity** properties
- Dynamics can be captured by a low-dimensional model
- Learned models can efficiently simulate realistic sequences
- Applications: compression, recognition, synthesis of videos

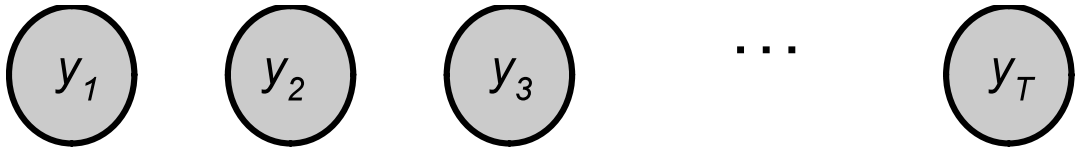
[1] S. Soatto, D. Doretto and Y. Wu. ***Dynamic Textures***. *Proceedings of the ICCV*, 2001

Linear Dynamical Systems

- Input data sequence

$$Y = [y_1 \; y_2 \ldots y_T] \qquad y_t \in \mathcal{R}^m$$

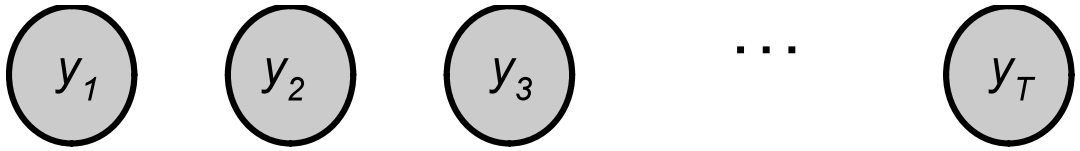
Linear Dynamical Systems



- Input data sequence

$$Y = [y_1 \; y_2 \ldots y_T] \qquad y_t \in \mathcal{R}^m$$

Linear Dynamical Systems



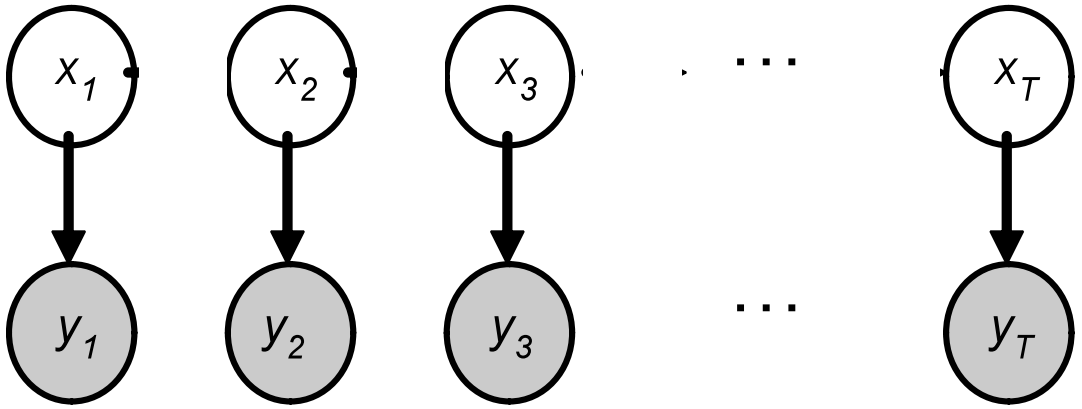
- Input data sequence

$$Y = [y_1 \; y_2 \ldots y_T] \qquad y_t \in \mathcal{R}^m$$

- Assume a **latent variable** that explains the data

$$X = [x_1 \; x_2 \ldots x_T] \qquad x_t \in \mathcal{R}^n$$

Linear Dynamical Systems



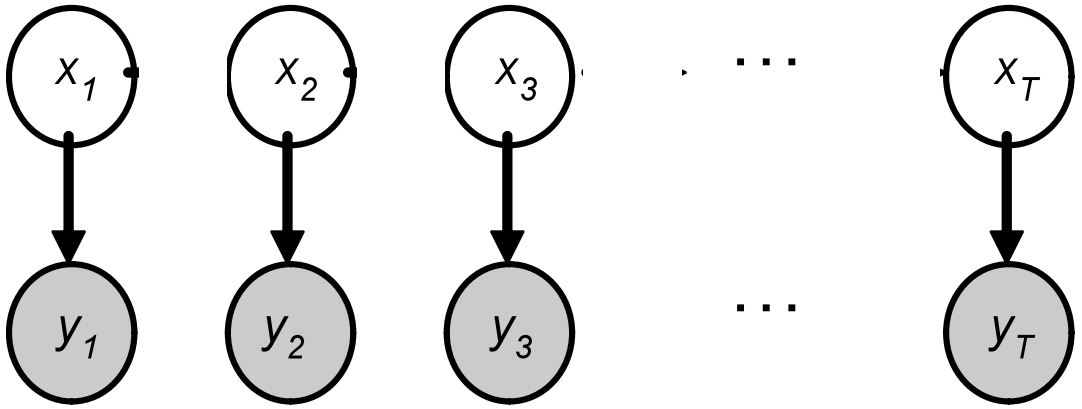
- Input data sequence

$$Y = [y_1 \; y_2 \ldots y_T] \qquad y_t \in \mathcal{R}^m$$

- Assume a **latent variable** that explains the data

$$X = [x_1 \; x_2 \ldots x_T] \qquad x_t \in \mathcal{R}^n$$

Linear Dynamical Systems



- Input data sequence

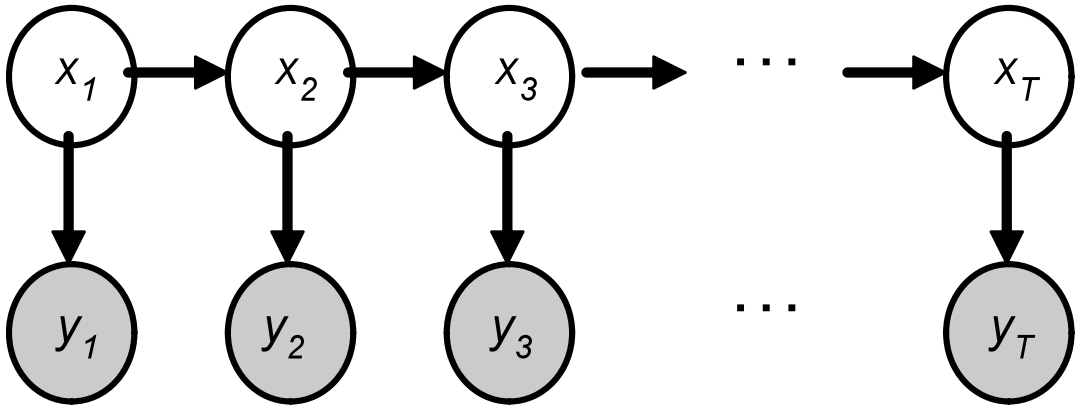
$$Y = [y_1 \; y_2 \ldots y_T] \qquad y_t \in \mathcal{R}^m$$

- Assume a **latent variable** that explains the data

$$X = [x_1 \; x_2 \ldots x_T] \qquad x_t \in \mathcal{R}^n$$

- Assume a **Markov model** on the latent variable

Linear Dynamical Systems



- Input data sequence

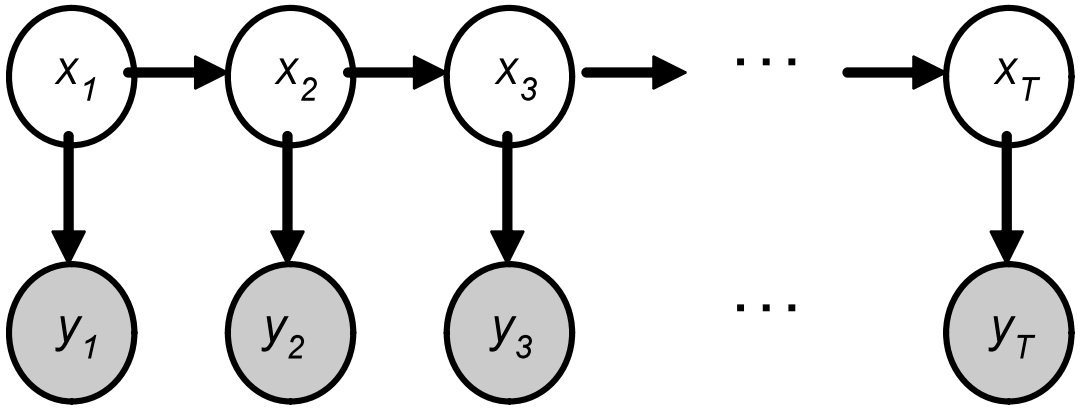
$$Y = [y_1 \; y_2 \ldots y_T] \qquad y_t \in \mathcal{R}^m$$

- Assume a **latent variable** that explains the data

$$X = [x_1 \; x_2 \ldots x_T] \qquad x_t \in \mathcal{R}^n$$

- Assume a **Markov model** on the latent variable

Linear Dynamical Systems[2]



- State and observation models:

$$x_{t+1} = Ax_t + w_t \qquad w_t \sim N(0, Q)$$
$$y_t = Cx_t + v_t \qquad v_t \sim N(0, R)$$

- Dynamics matrix: $A \in \mathcal{R}^{n \times n}$
- Observation matrix: $C \in \mathcal{R}^{m \times n}$

[2] *Kalman, R. E. (1960).* ***A new approach to linear filtering and prediction problems.*** *Trans.ASME-JBE*

Linear Dynamical Systems[2]

This talk:

- Learning LDS parameters from data while ensuring a **stable** dynamics matrix ***A*** more efficiently and accurately than previous methods

- State and observation models:

$$x_{t+1} = Ax_t + w_t \qquad w_t \sim N(0, Q)$$
$$y_t = Cx_t + v_t \qquad v_t \sim N(0, R)$$

- Dynamics matrix: $A \in \mathcal{R}^{n \times n}$
- Observation matrix: $C \in \mathcal{R}^{m \times n}$

[2] *Kalman, R. E. (1960).* ***A new approach to linear filtering and prediction problems.*** *Trans.ASME-JBE*

Learning Linear Dynamical Systems

- Suppose we have an estimated state sequence

$$\hat{X}_{1:\tau} = [\hat{x}_1 \ \hat{x}_2 \ldots \hat{x}_\tau] \in \mathcal{R}^{n \times \tau}$$

Learning Linear Dynamical Systems

- Suppose we have an estimated state sequence

$$\hat{X}_{1:\tau} = [\hat{x}_1 \ \hat{x}_2 \ldots \hat{x}_\tau] \in \mathcal{R}^{n \times \tau}$$

- Define *state reconstruction error* as the objective:

$$J(A) = ||AX_{1:\tau-1} - X_{2:\tau}||_F^2$$

Learning Linear Dynamical Systems

- Suppose we have an estimated state sequence

$$\hat{X}_{1:\tau} = [\hat{x}_1 \ \hat{x}_2 \ldots \hat{x}_\tau] \in \mathcal{R}^{n \times \tau}$$

- Define *state reconstruction error* as the objective:

$$J(A) = ||AX_{1:\tau-1} - X_{2:\tau}||_F^2$$

- We would like to learn $\hat{A}$ such that

$$x_{t+1} \approx \hat{A}x_t$$

i.e. $\hat{A} = min_A J(A)$

Subspace Identification[3]

- Subspace ID uses matrix decomposition to estimate the state sequence

[3] *P. Van Overschee and B. De Moor* ***Subspace Identification for Linear Systems.*** Kluwer, 1996

Subspace Identification[3]

- Subspace ID uses matrix decomposition to estimate the state sequence

- Build a *Hankel matrix* ***D*** of stacked observations

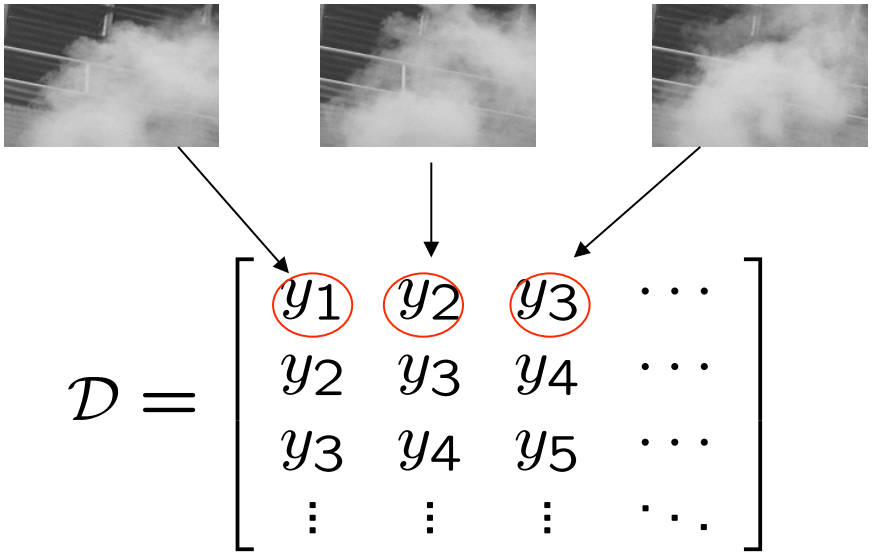
$$\mathcal{D} = \begin{bmatrix} y_1 & y_2 & y_3 & \cdots \\ y_2 & y_3 & y_4 & \cdots \\ y_3 & y_4 & y_5 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

[3] *P. Van Overschee and B. De Moor* ***Subspace Identification for Linear Systems.*** Kluwer, 1996

Subspace Identification[3]

- Subspace ID uses matrix decomposition to estimate the state sequence

- Build a *Hankel matrix* ***D*** of stacked observations



[3] *P. Van Overschee and B. De Moor* ***Subspace Identification for Linear Systems.*** Kluwer, 1996

Subspace Identification[3]

- In expectation, the Hankel matrix is inherently **low-rank!**

[3] *P. Van Overschee and B. De Moor* ***Subspace Identification for Linear Systems.*** Kluwer, 1996

Subspace Identification[3]

- In expectation, the Hankel matrix is inherently **low-rank!**

$$E(\mathcal{D}) = \begin{bmatrix} Cx_1 & Cx_2 & Cx_3 & \cdots \\ Cx_2 & Cx_3 & Cx_4 & \cdots \\ Cx_3 & Cx_4 & Cx_5 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} = \begin{bmatrix} Cx_1 & Cx_2 & Cx_3 & \cdots \\ CAx_1 & CAx_2 & CAx_3 & \cdots \\ CA^2x_1 & CA^2x_2 & CA^2x_3 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

[3] *P. Van Overschee and B. De Moor* ***Subspace Identification for Linear Systems.*** Kluwer, 1996

Subspace Identification[3]

- In expectation, the Hankel matrix is inherently **low-rank!**

$$E(\mathcal{D}) = \begin{bmatrix} Cx_1 & Cx_2 & Cx_3 & \cdots \\ Cx_2 & Cx_3 & Cx_4 & \cdots \\ Cx_3 & Cx_4 & Cx_5 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} = \begin{bmatrix} Cx_1 & Cx_2 & Cx_3 & \cdots \\ CAx_1 & CAx_2 & CAx_3 & \cdots \\ CA^2x_1 & CA^2x_2 & CA^2x_3 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

$$= \begin{bmatrix} C \\ CA \\ CA^2 \\ CA^3 \\ \vdots \end{bmatrix} \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & \cdots \end{bmatrix}$$

[3] *P. Van Overschee and B. De Moor* ***Subspace Identification for Linear Systems.*** Kluwer, 1996

Subspace Identification[3]

- In expectation, the Hankel matrix is inherently **low-rank!**

$$E(\mathcal{D}) = \begin{bmatrix} Cx_1 & Cx_2 & Cx_3 & \cdots \\ Cx_2 & Cx_3 & Cx_4 & \cdots \\ Cx_3 & Cx_4 & Cx_5 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} = \begin{bmatrix} Cx_1 & Cx_2 & Cx_3 & \cdots \\ CAx_1 & CAx_2 & CAx_3 & \cdots \\ CA^2x_1 & CA^2x_2 & CA^2x_3 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

- Can use SVD to obtain the low-dimensional state sequence

$$= \begin{bmatrix} C \\ CA \\ CA^2 \\ CA^3 \\ \vdots \end{bmatrix} \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & \cdots \end{bmatrix}$$

[3] *P. Van Overschee and B. De Moor* ***Subspace Identification for Linear Systems.*** Kluwer, 1996

Subspace Identification[3]

- In expectation, the Hankel matrix is inherently **low-rank!**

$$E(\mathcal{D}) = \begin{bmatrix} Cx_1 & Cx_2 & Cx_3 & \cdots \\ Cx_2 & Cx_3 & Cx_4 & \cdots \\ Cx_3 & Cx_4 & Cx_5 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} = \begin{bmatrix} Cx_1 & Cx_2 & Cx_3 & \cdots \\ CAx_1 & CAx_2 & CAx_3 & \cdots \\ CA^2x_1 & CA^2x_2 & CA^2x_3 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

- Can use SVD to obtain the low-dimensional state sequence

$$= \begin{bmatrix} C \\ CA \\ CA^2 \\ CA^3 \\ \vdots \end{bmatrix} \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & \cdots \end{bmatrix}$$

For *D* with *k* observations per column,

$$\mathcal{D} \approx U\Sigma V^T = \underbrace{\boxed{}}_{U} \underbrace{\boxed{n \times \tau}}_{\Sigma V^T}$$

[3] *P. Van Overschee and B. De Moor* ***Subspace Identification for Linear Systems.*** Kluwer, 1996

Subspace Identification[3]

- In expectation, the Hankel matrix is inherently **low-rank!**

$$E(\mathcal{D}) = \begin{bmatrix} Cx_1 & Cx_2 & Cx_3 & \cdots \\ Cx_2 & Cx_3 & Cx_4 & \cdots \\ Cx_3 & Cx_4 & Cx_5 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} = \begin{bmatrix} Cx_1 & Cx_2 & Cx_3 & \cdots \\ CAx_1 & CAx_2 & CAx_3 & \cdots \\ CA^2x_1 & CA^2x_2 & CA^2x_3 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

$$= \begin{bmatrix} C \\ CA \\ CA^2 \\ CA^3 \\ \vdots \end{bmatrix} \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & \cdots \end{bmatrix}$$

- Can use SVD to obtain the low-dimensional state sequence

For *D* with *k* observations per column,

$$\mathcal{D} \approx U\Sigma V^T =$$

$mk \times n$ $\qquad$ $n \times \tau$

U $\qquad$ ΣV^T

$$\hat{C} = U(1:m,:)$$

$$\hat{X} = \Sigma V^T = [\hat{x}_1 \ \hat{x}_2 \ldots \hat{x}_\tau]$$

[3] *P. Van Overschee and B. De Moor* ***Subspace Identification for Linear Systems.*** Kluwer, 1996

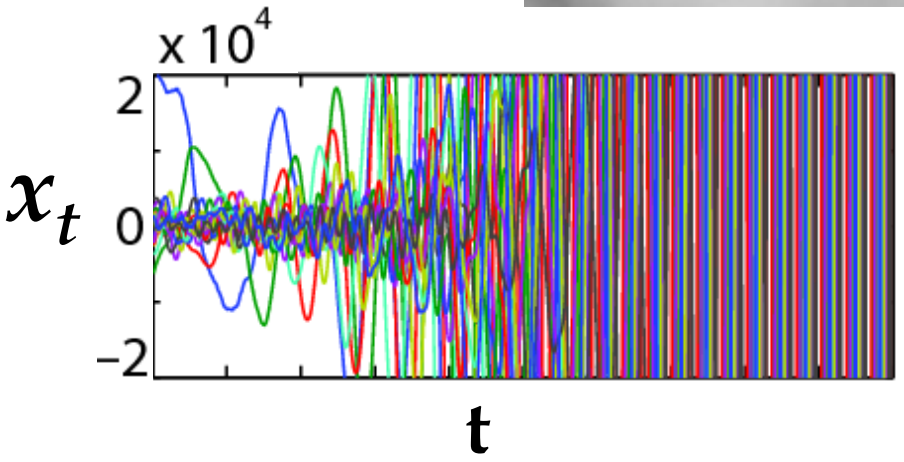
- Lets train an LDS for **steam** textures using this algorithm, and simulate a video from it!

$$Y = [y_1 \; y_2 \ldots y_T]$$

$$= [$$  $$\ldots]$$

$$x_t \in R^{40}$$

Simulating from a learned model



The model is unstable

Notation

- $\lambda_1 , \ldots , \lambda_n$: **eigenvalues of A** $(|\lambda_1| > \ldots > |\lambda_n|)$
- $v_1, \ldots, v_n$: **unit-length eigenvectors of A**
- $\sigma_1, \ldots, \sigma_n$: **singular values of A** $(\sigma_1 > \sigma_2 > \ldots \sigma_n)$

- S_λ : **matrices with** $|\lambda_1| \cdot 1$
- S_σ : **matrices with** $\sigma_1 \cdot 1$

Stability

- a matrix A is **stable if $|\lambda_1| \cdot 1$,** i.e. if $A \in S_\lambda$

Stability

- a matrix A is **stable if $|\lambda_1| \cdot 1$,** i.e. if $A \in S_\lambda$

$$A_1 = \begin{bmatrix} 0.3 & 10 \\ 0 & 0.3 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} 0.3 & 9.9 \\ 0.1 & 0.3 \end{bmatrix}$$

Stability

- a matrix A is **stable if $|\lambda_1| \cdot 1$,** i.e. if $A \in S_\lambda$

$$A_1 = \begin{bmatrix} 0.3 & 10 \\ 0 & 0.3 \end{bmatrix}$$

$|\lambda_1| = 0.3$

$$A_2 = \begin{bmatrix} 0.3 & 9.9 \\ 0.1 & 0.3 \end{bmatrix}$$

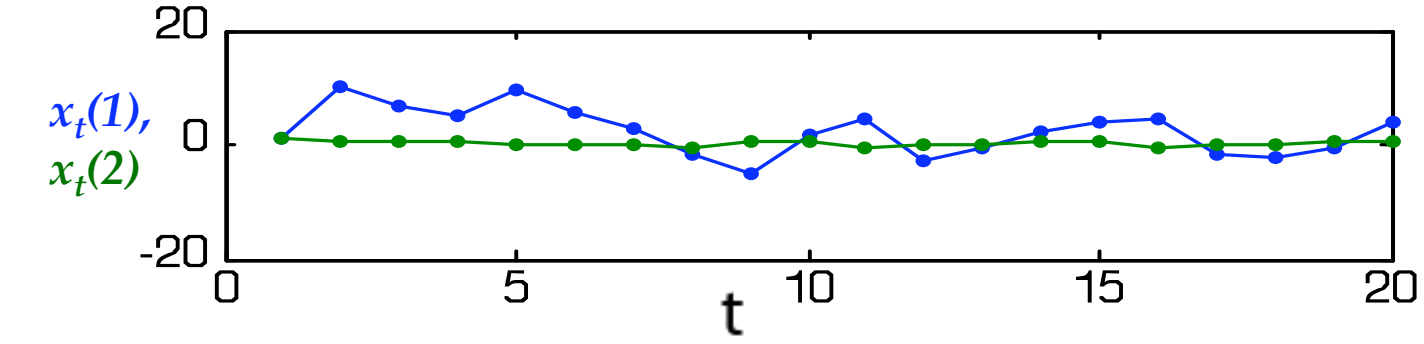
$|\lambda_1| = 1.295$

Stability

- a matrix A is **stable if $|\lambda_1| \cdot 1$,** i.e. if $A \in S_\lambda$

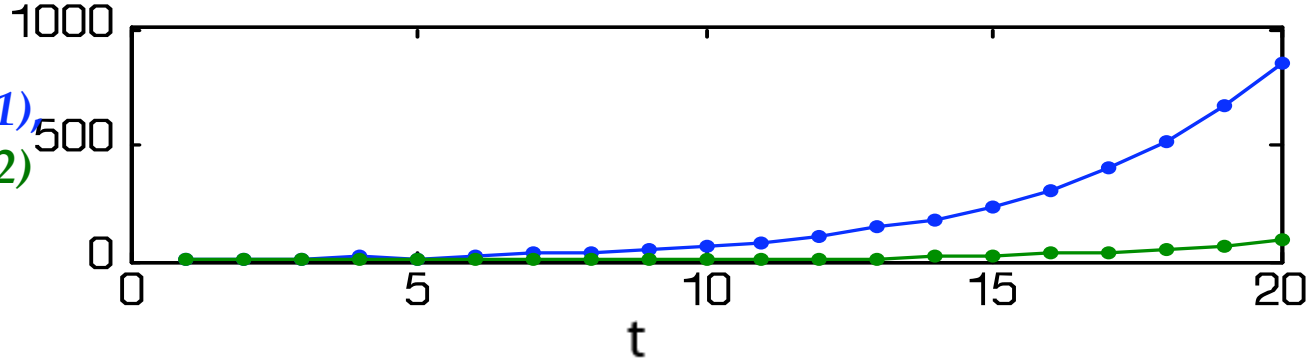
$$A_1 = \begin{bmatrix} 0.3 & 10 \\ 0 & 0.3 \end{bmatrix}$$

$|\lambda_1| = 0.3$



$$A_2 = \begin{bmatrix} 0.3 & 9.9 \\ 0.1 & 0.3 \end{bmatrix}$$

$|\lambda_1| = 1.295$

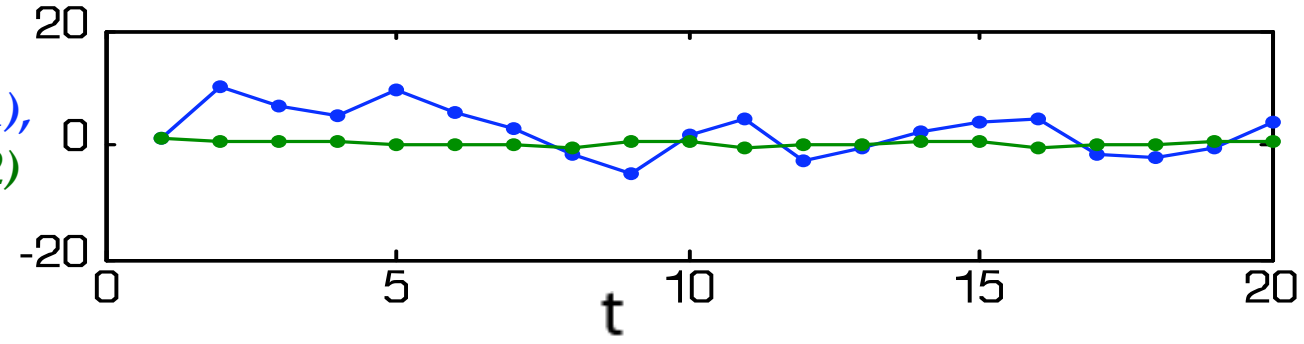


Stability

- a matrix A is **stable if $|\lambda_1| \cdot 1$,** i.e. if $A \in S_\lambda$

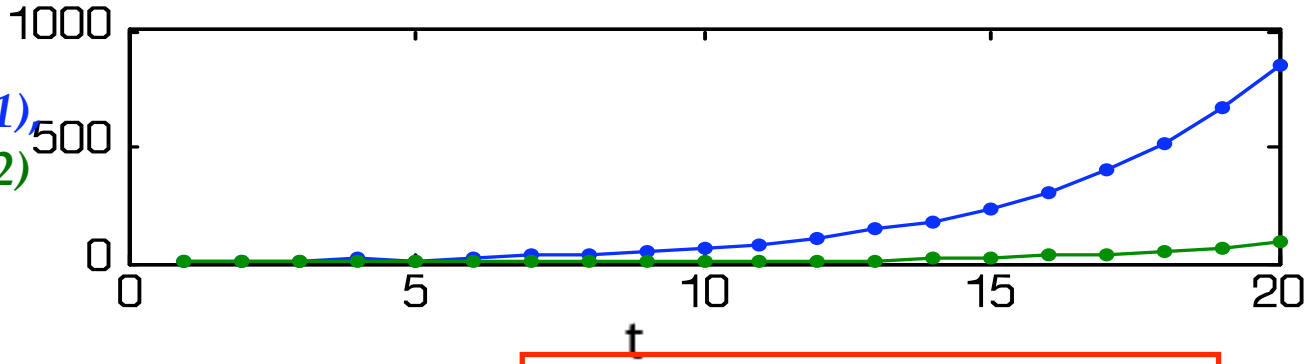
$$A_1 = \begin{bmatrix} 0.3 & 10 \\ 0 & 0.3 \end{bmatrix}$$

$|\lambda_1| = 0.3$



$$A_2 = \begin{bmatrix} 0.3 & 9.9 \\ 0.1 & 0.3 \end{bmatrix}$$

$|\lambda_1| = 1.295$



- We would like to solve

$$\min_A J(A)$$
$$\text{s.t. } A \in S_\lambda$$

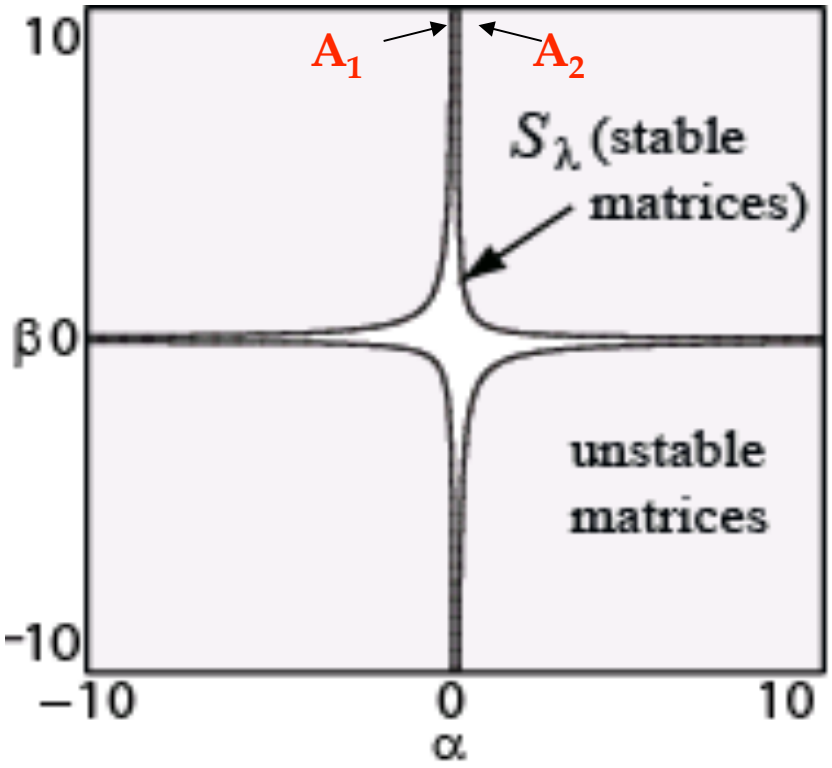
Stability and Convexity

- But S_λ is non-convex!

Stability and Convexity

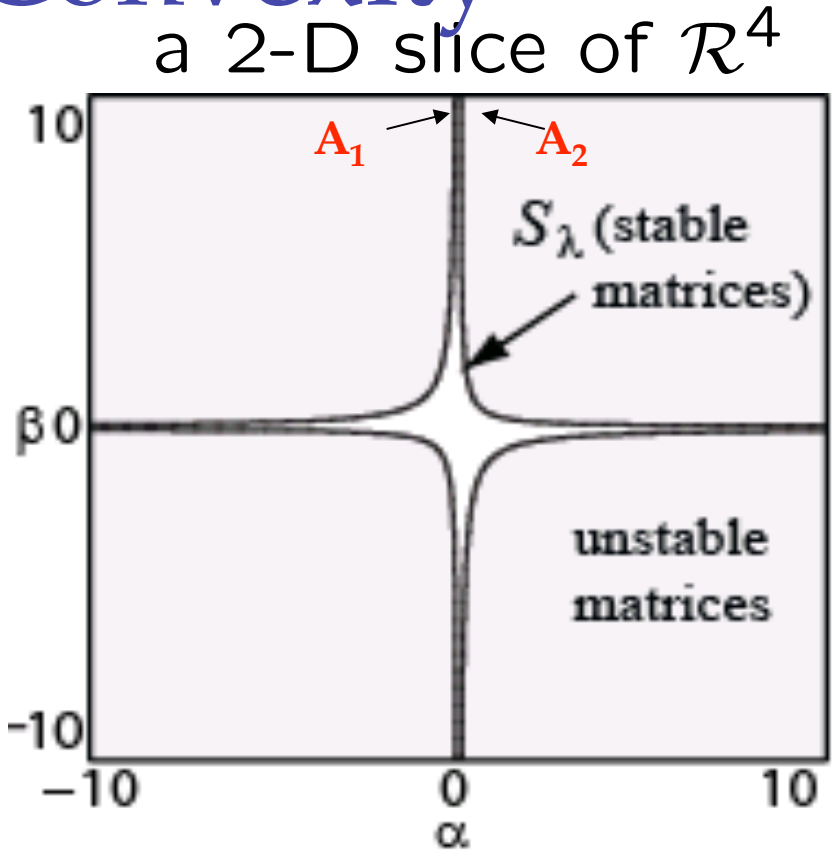
- But S_λ is non-convex!

a 2-D slice of $\mathcal{R}^4$



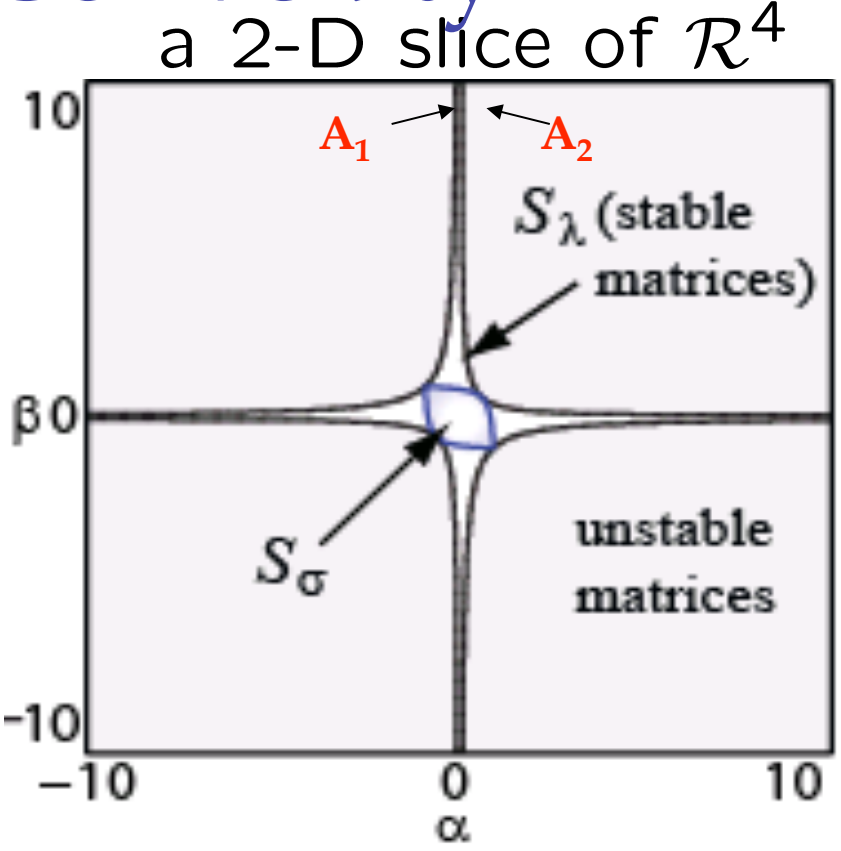
Stability and Convexity

- But S_λ is non-convex!
- Lets look at S_σ instead …
  - S_σ is *convex*
  - $S_\sigma \mu S_\lambda$



Stability and Convexity

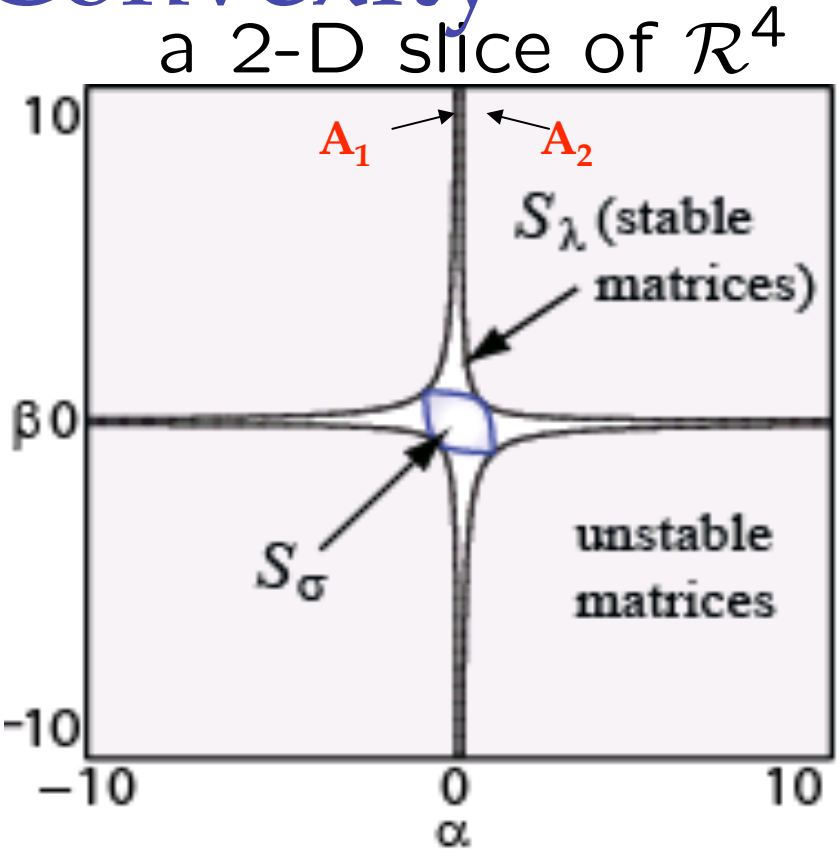
- But S_λ is non-convex!
- Lets look at S_σ instead …
  - S_σ is *convex*
  - $S_\sigma \mu S_\lambda$



Stability and Convexity

- But S_λ is non-convex!

- Lets look at S_σ instead …
  - S_σ is *convex*
  - $S_\sigma \subset S_\lambda$



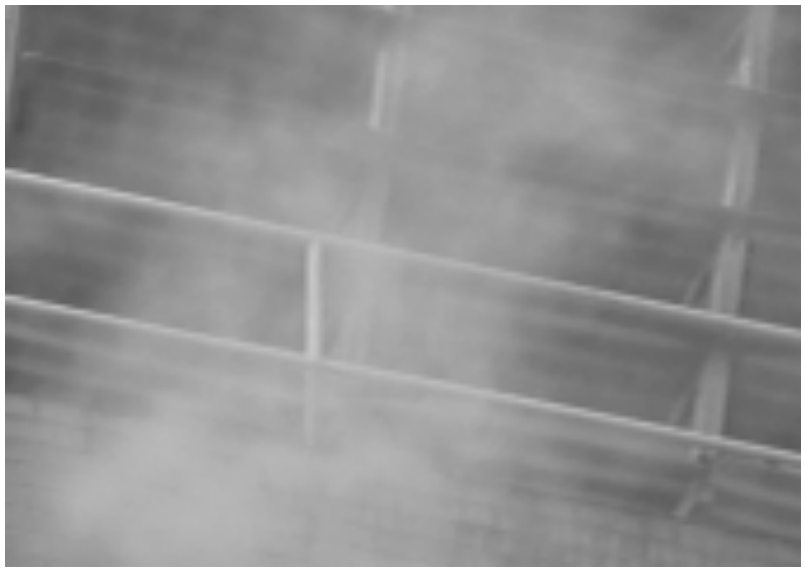
- Previous work[4] exploits these properties to learn a stable A by solving the semi-definite program

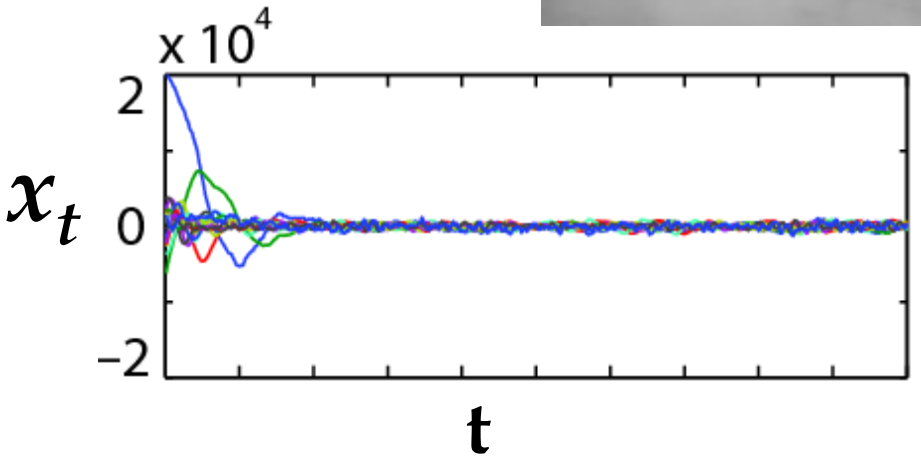
$$\min_A J(A)$$
$$\text{s.t. } A \in S_\sigma$$

[4] S. L. Lacy and D. S. Bernstein. Subspace identification with guaranteed stability using constrained optimization. In *Proc. of the ACC* (2002), *IEEE Trans. Automatic Control* (2003)

- Lets train an LDS for **steam** again, this time constraining A to be in S_σ

Simulating from a Lacy-Bernstein stable texture model





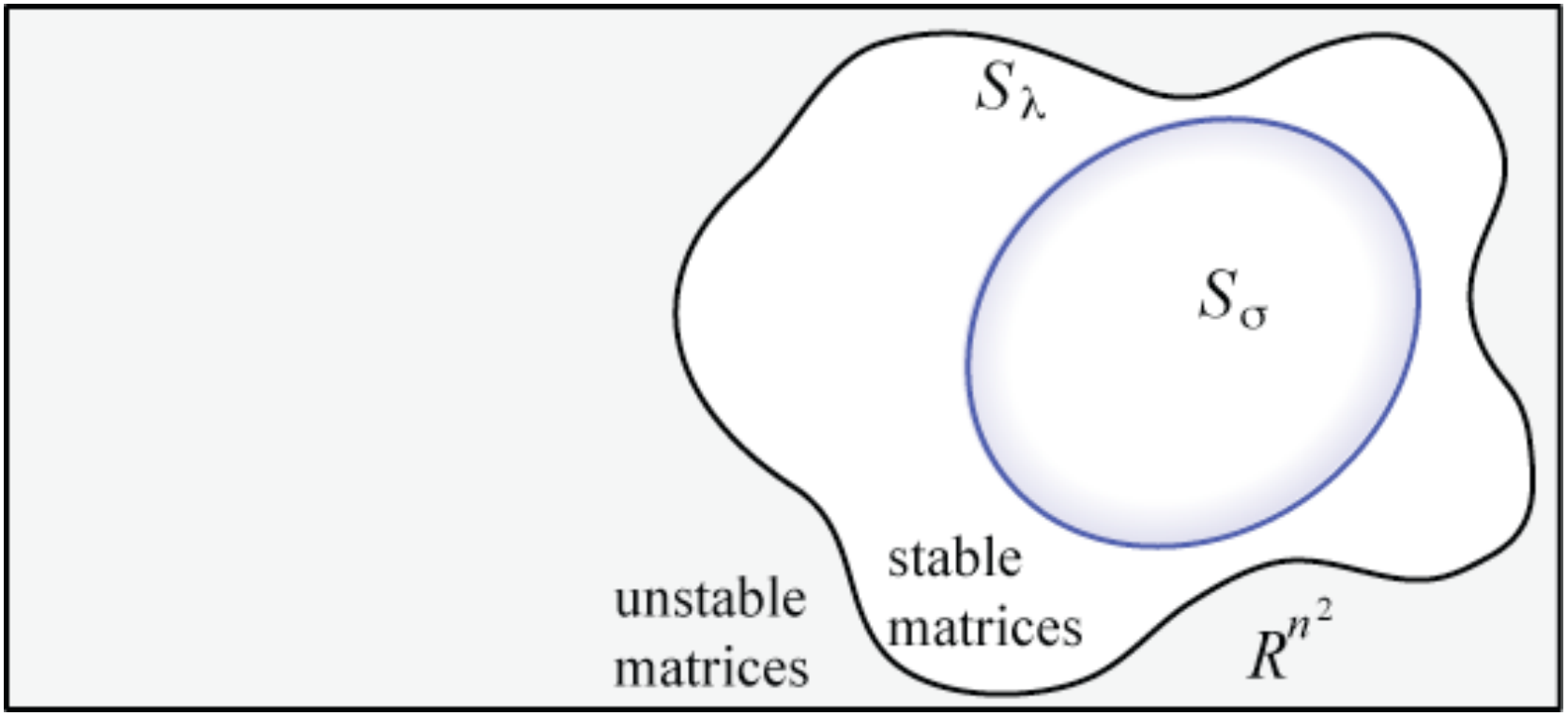
Model is over-constrained.

Can we do better?

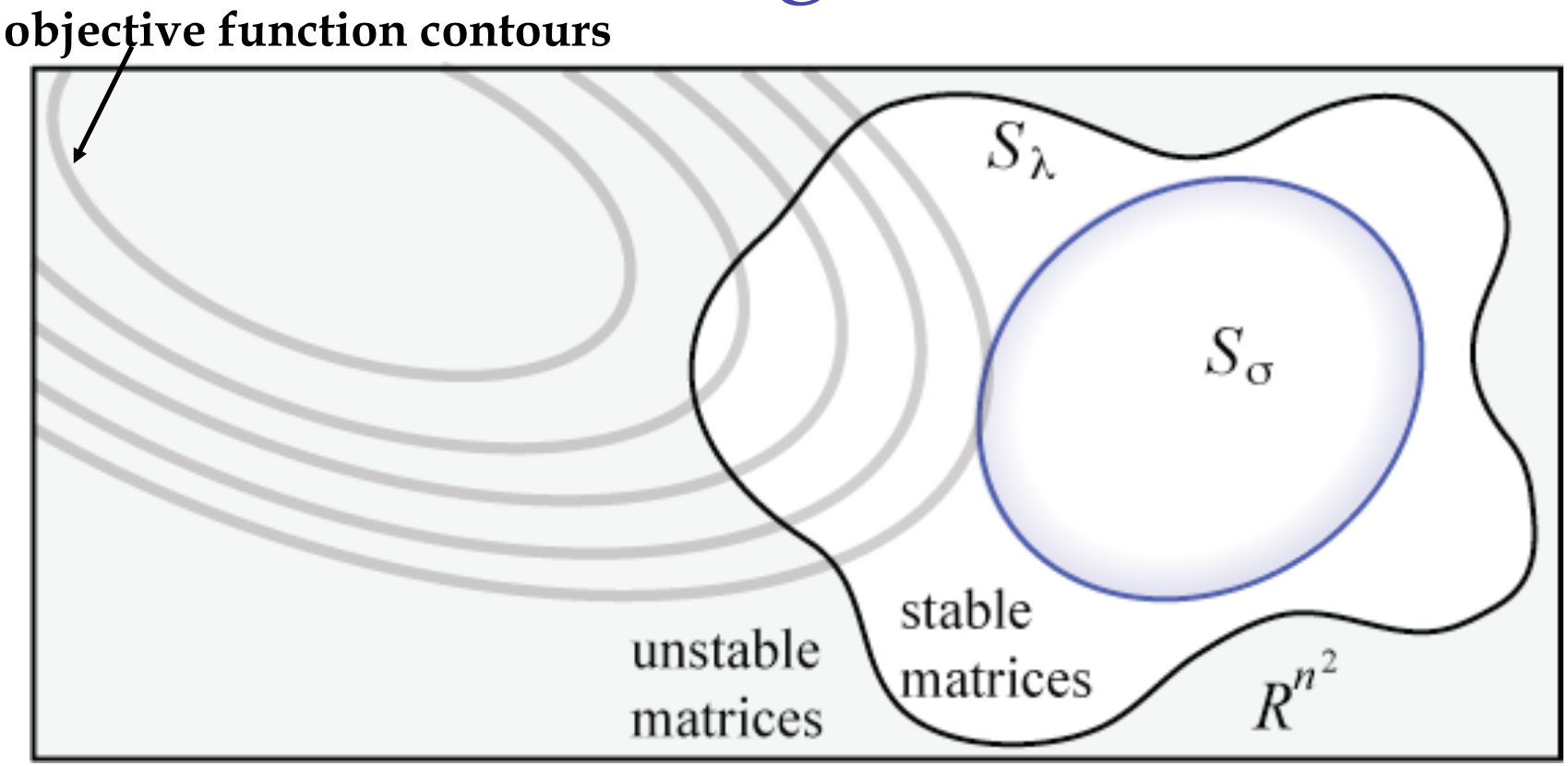
Our Approach

- Formulate the S_σ approximation of the problem as a **semi-definite program (SDP)**
- Start with a **quadratic program (QP) relaxation** of this SDP, and incrementally **add constraints**
- Because the SDP is an inner approximation of the problem, we **reach stability early,** before reaching the feasible set of the SDP
- We **interpolate** the solution to return the best stable matrix possible

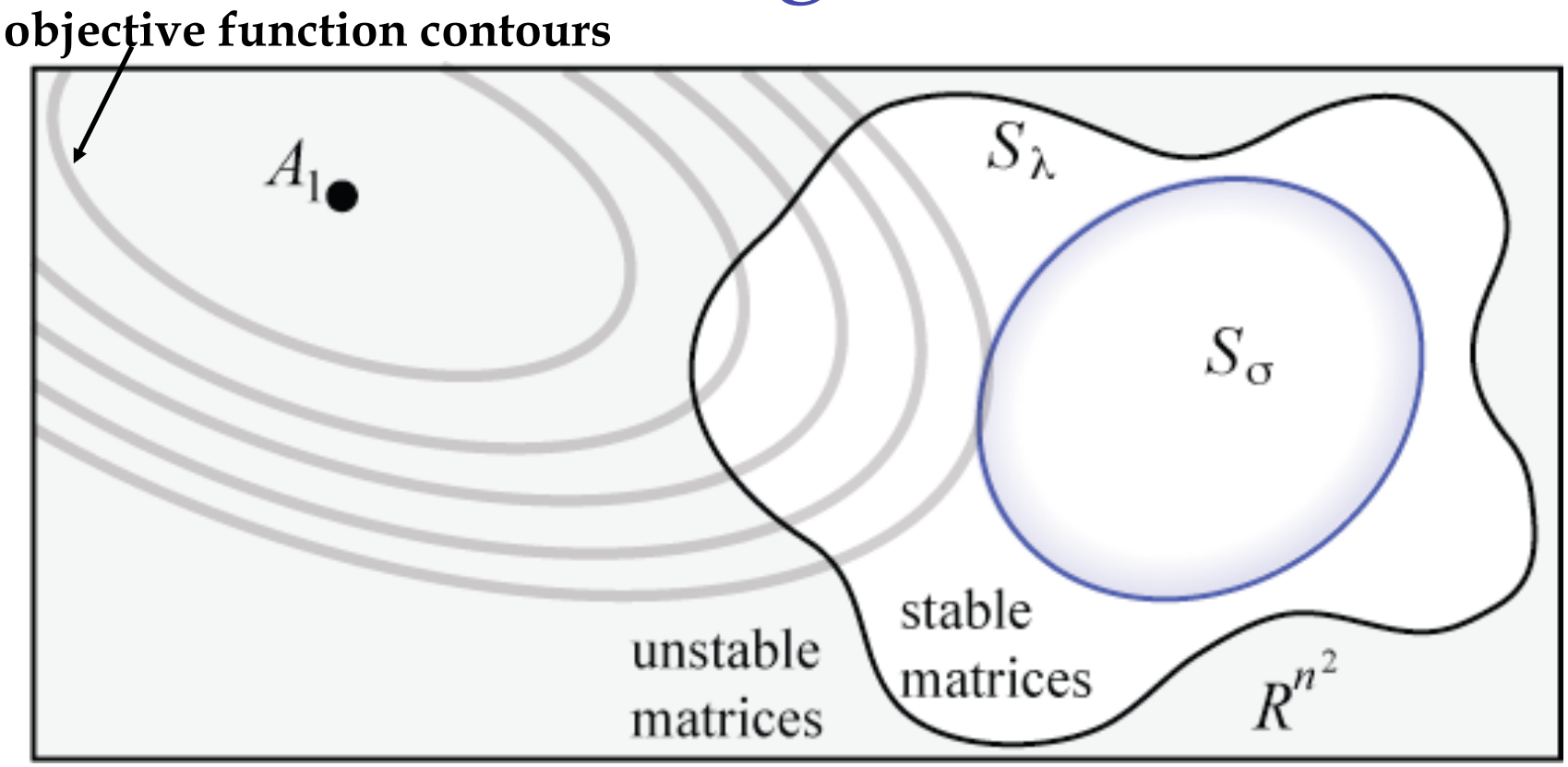
The Algorithm



The Algorithm

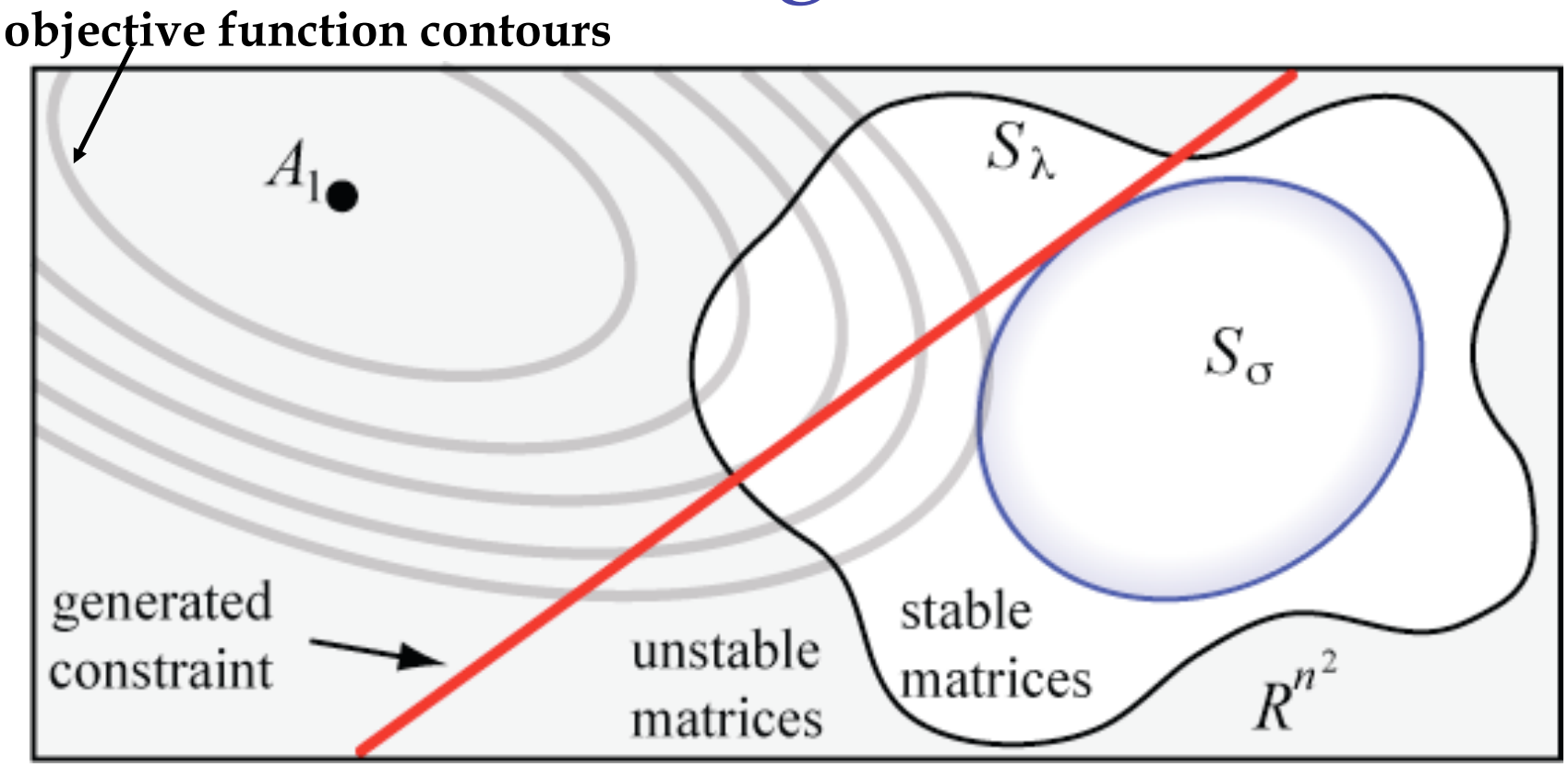


The Algorithm



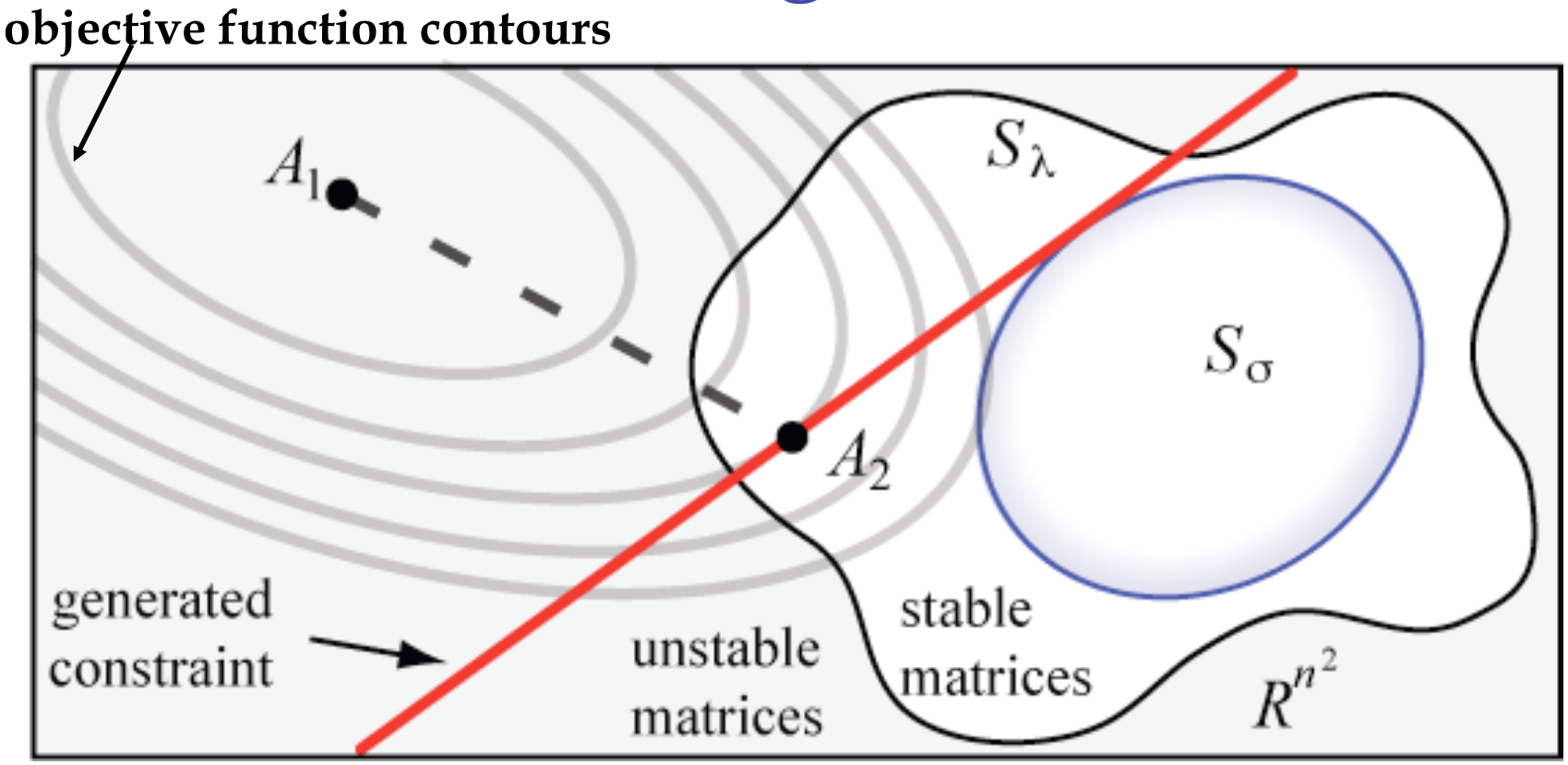
- A_1: unconstrained QP solution (least squares estimate)

The Algorithm



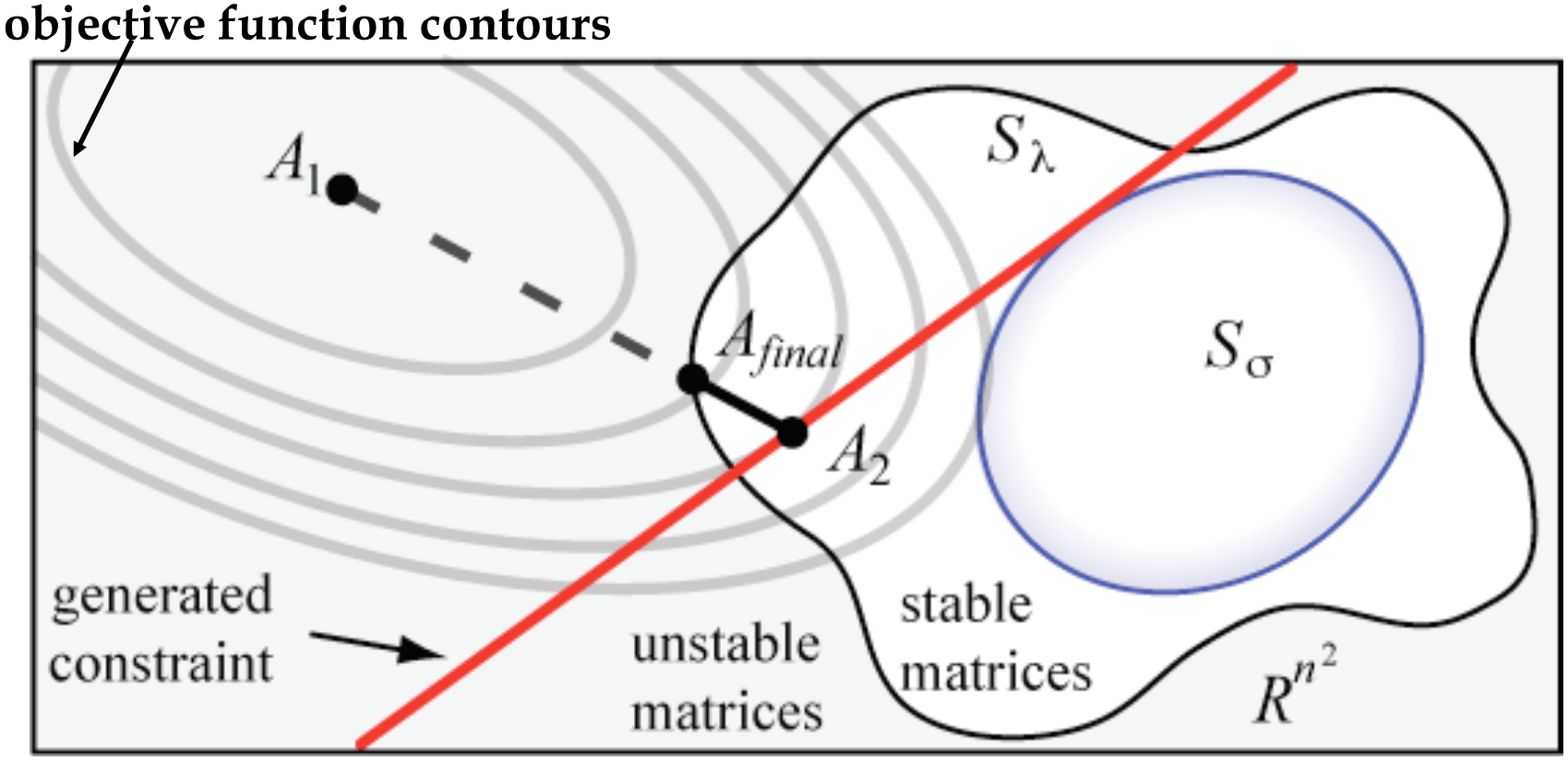
- A_1: unconstrained QP solution (least squares estimate)

The Algorithm



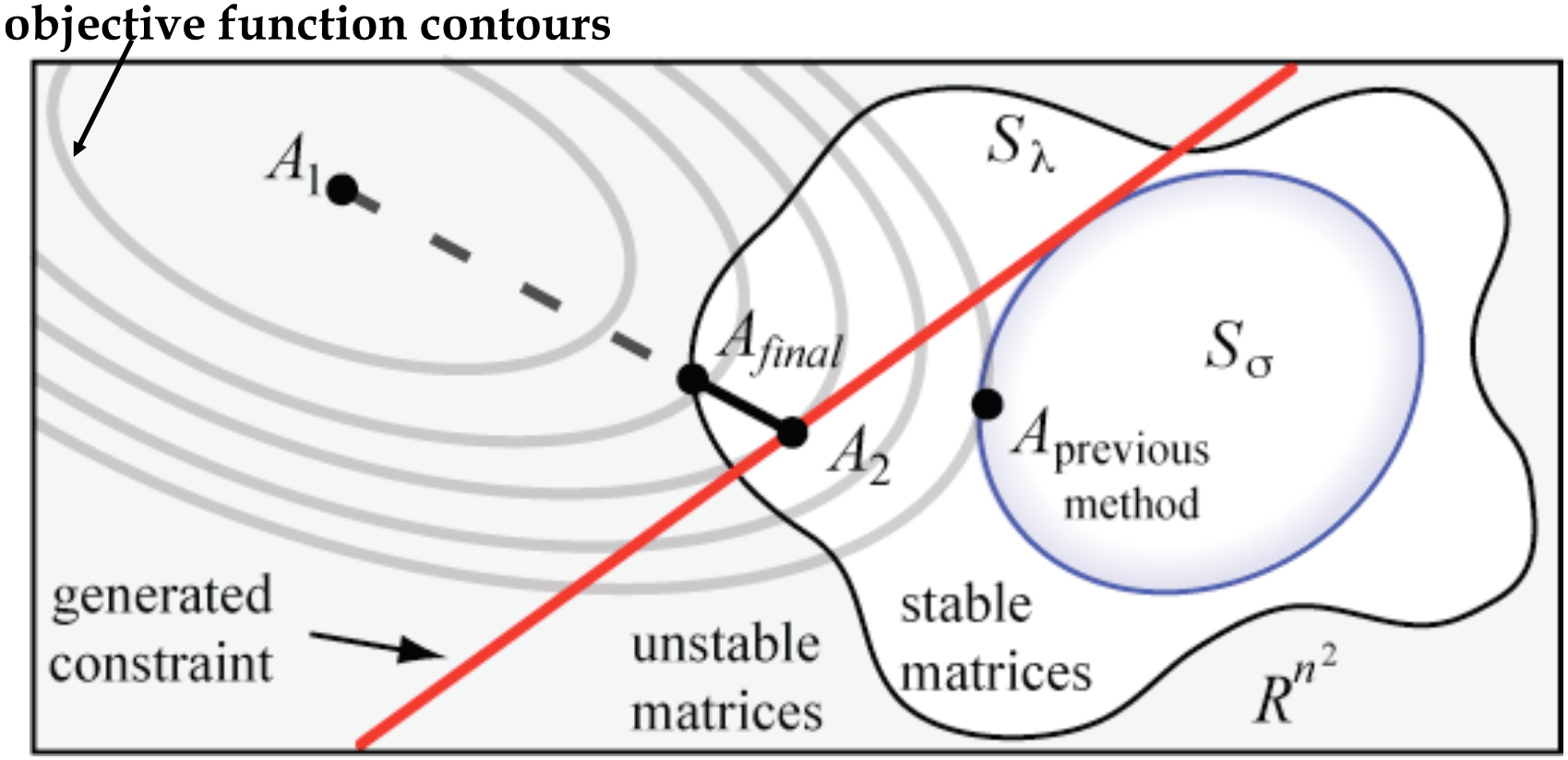
- A_1: unconstrained QP solution (least squares estimate)
- A_2: QP solution after 1 constraint (happens to be stable)

The Algorithm



- A_1: unconstrained QP solution (least squares estimate)
- A_2: QP solution after 1 constraint (happens to be stable)
- A_{final}: Interpolation of stable solution with the last one

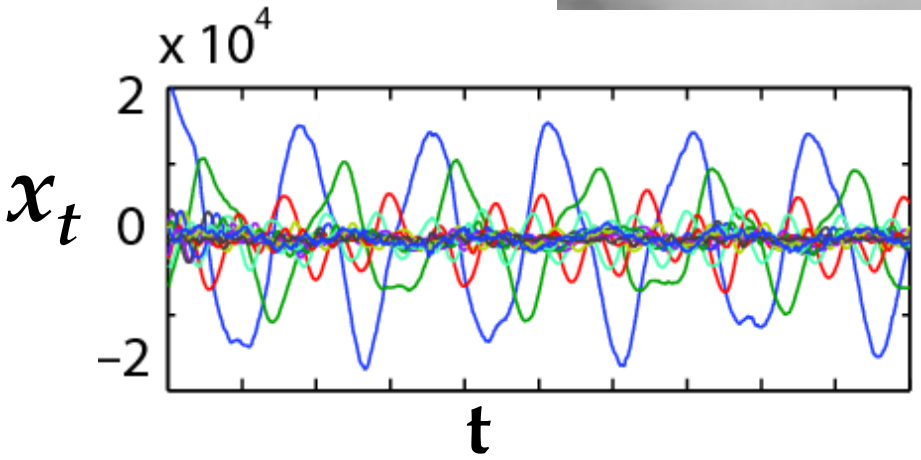
The Algorithm



- A_1: unconstrained QP solution (least squares estimate)
- A_2: QP solution after 1 constraint (happens to be stable)
- A_{final}: Interpolation of stable solution with the last one
- $A_{previous\ method}$: Lacy Bernstein (2002)

- Lets train an LDS for **steam** using constraint generation, and simulate …

Simulating from a Constraint Generation stable texture model



Model captures more dynamics and is still stable

- Least Squares

- Constraint Generation

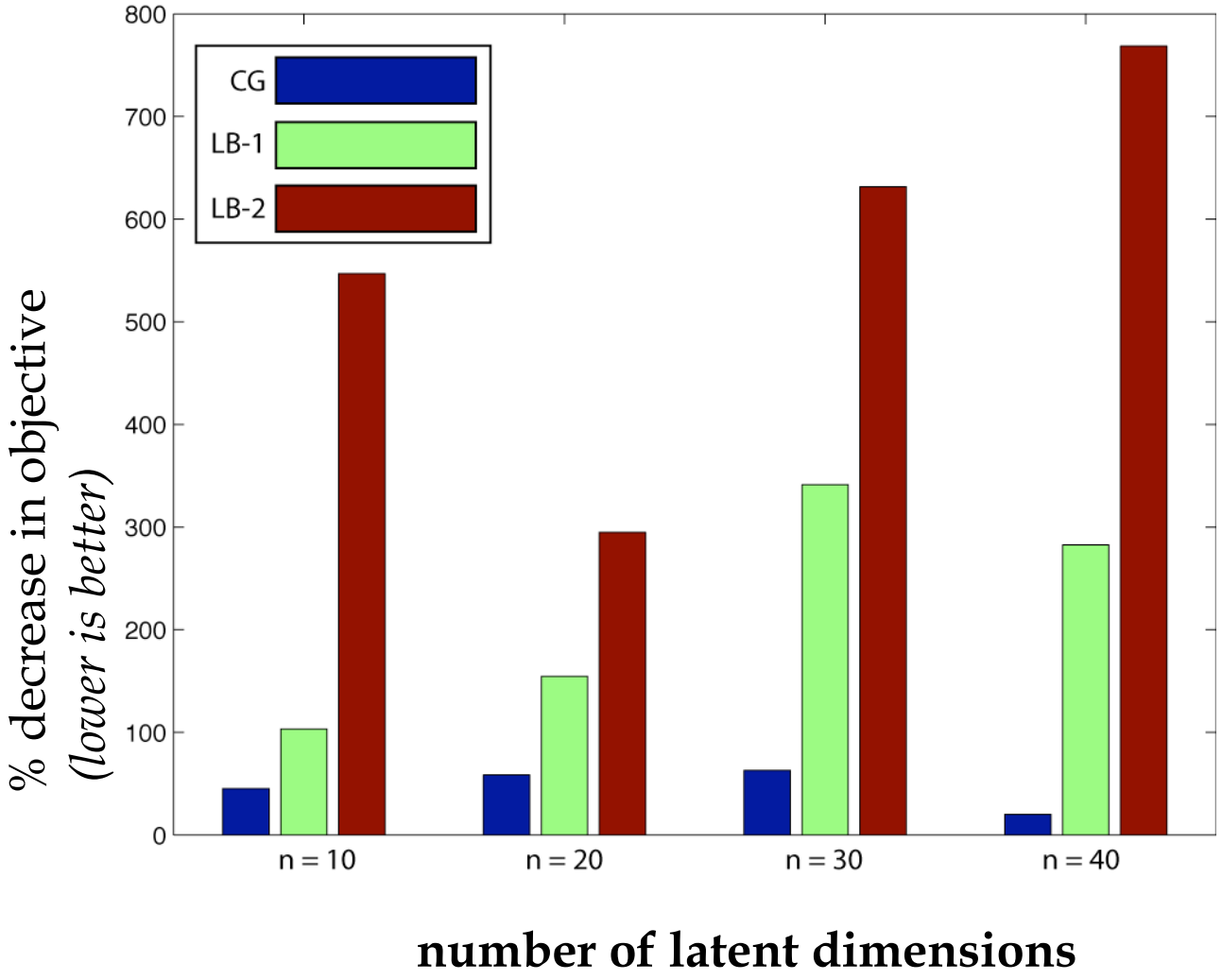
Empirical Evaluation

- Algorithms:
  - Constraint Generation – **CG** (our method)
  - Lacy and Bernstein (2002) –**LB-1**
    - finds a $\sigma_1 \cdot 1$ solution
  - Lacy and Bernstein (2003)–**LB-2**
    - solves a similar problem in a transformed space

- Data sets
  - Dynamic textures
  - Biosurveillance baseline models (see paper)

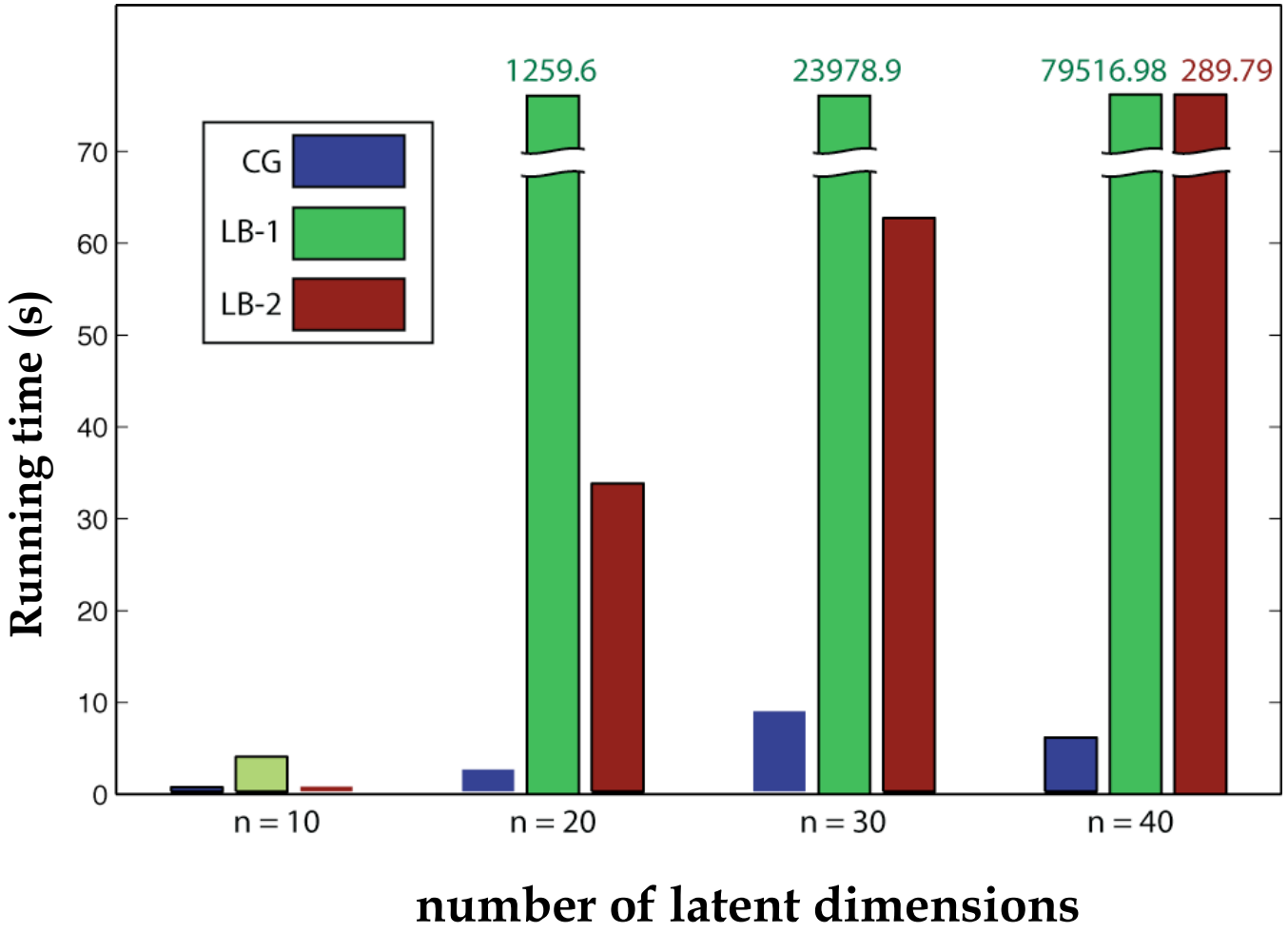
Reconstruction error

- *Steam* video



Running time

- *Steam* video



Conclusion

- A novel **constraint generation algorithm** for learning **stable linear dynamical systems**

- SDP relaxation enables us to optimize over a **larger set of matrices** while being **more efficient**

Conclusion

- A novel **constraint generation algorithm** for learning **stable linear dynamical systems**

- SDP relaxation enables us to optimize over a **larger set of matrices** while being **more efficient**

- Future work:
  - Adding stability constraints to EM
  - Stable models for more structured dynamic textures

- Thank you!

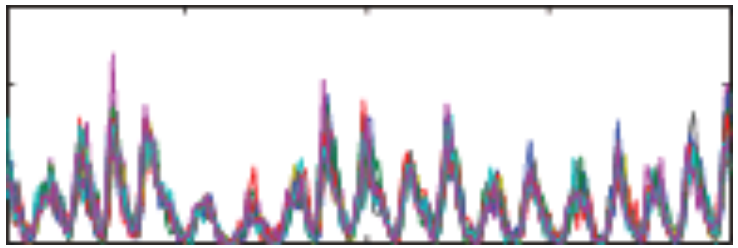
Subspace ID with Hankel matrices

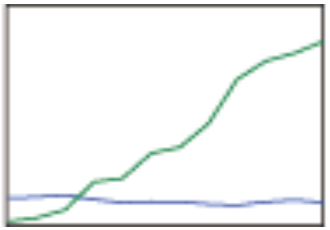
Stacking multiple observations in D forces latent states to model the future

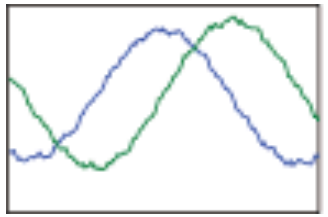
$$\mathcal{D} = \begin{bmatrix} y_1 & y_2 & y_3 & \cdots & y_\tau \\ y_2 & y_3 & y_4 & \cdots & y_{\tau+1} \\ y_3 & y_4 & y_5 & \cdots & y_{\tau+2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ y_k & y_{k+1} & y_{k+2} & \cdots & y_{\tau+k} \end{bmatrix}_{mk \times \tau} = \underbrace{\begin{matrix} \\ \\ \\ \\ \\ \end{matrix}}_{U}{}_{mk \times n} \; \underbrace{\begin{matrix} \\ \\ \\ \\ \\ \end{matrix}}_{\Sigma V^T}{}_{n \times \tau}$$

$$\hat{C} = U(1:m, :)$$

e.g. annual sunspots data with 12-year cycles



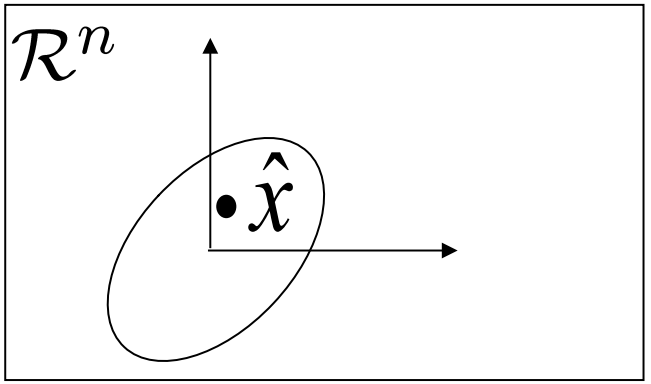




First 2 columns of U

Stability and Convexity

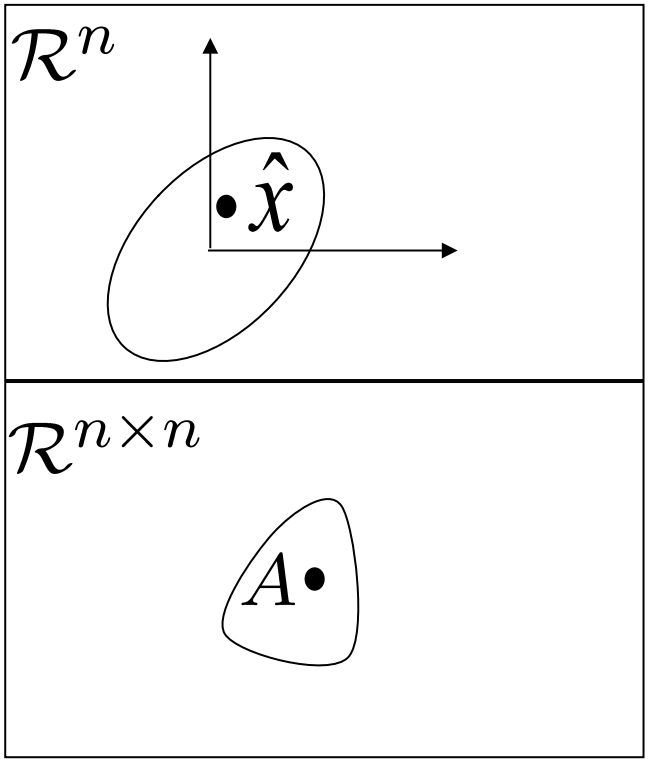
- The state space of a stable LDS lies inside some ellipse



Stability and Convexity

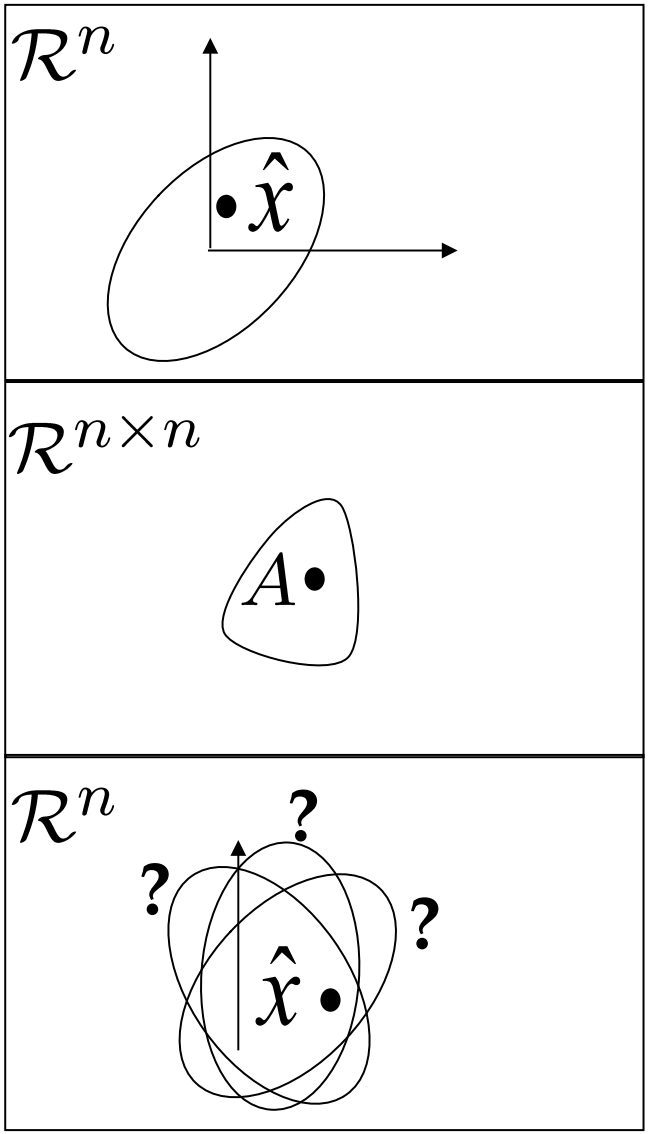
- The state space of a stable LDS lies inside some ellipse

- The set of matrices that map a particular ellipse into itself (and hence are stable) is convex



Stability and Convexity

- The state space of a stable LDS lies inside some ellipse
- The set of matrices that map a particular ellipse into itself (and hence are stable) is convex
- If we knew in advance which ellipse contains our state space, finding A would be a convex problem.
  But we don't



Stability and Convexity

- The state space of a stable LDS lies inside some ellipse
- The set of matrices that map a particular ellipse into itself (and hence are stable) is convex
- If we knew in advance which ellipse contains our state space, finding A would be a convex problem.
  But we don't
- …and the set of all stable matrices is non-convex