IEEE Conference on Decision and Control 2013

A Decentralized Algorithm for Stability Analysis of Nonlinear Systems

Reza Kamyar, Matthew M. Peet

Cybernetic Systems and Controls Laboratory (CSCL)

Arizona State University

Dec/12/2013

Solving large-scale problems in control

- We designed decentralized algorithms to decide the stability of the linear systems

$$\dot{x} = A(\alpha)x(t), \quad \alpha \in \Delta \text{ simplex}$$

with 100+ states. We set-up and solved the LMI conditions given by Polya's theorem (TAC2013)

- We designed decentralized algorithms to decide the stability of

$$\dot{x} = A(\alpha)x(t), \quad \alpha \in \Phi \text{ hypercube}$$

We set-up and solved the LMI conditions given by a multi-simplex version of Polya's theorem (CDC2012)

- Our goal to extend our algorithms to decide the local stability of nonlinear systems:

$$\dot{x} = f(x, \alpha) \quad \alpha \in \Phi$$

Nonlinear systems with polynomial vector fields

We examine the local stability of the systems of nonlinear ODEs

$$\dot{x}(t) = f(x) = A(x)x(t), \quad x(t) \in \mathbb{R}^n$$

$$A(x) = \sum_{|\alpha| \le d_a} A_\alpha x^\alpha$$

- $A_\alpha \in \mathbb{R}^{n \times n}$ are the coefficients
- d_a is the degree of $A(x)$
- $x^\alpha = x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ are the monomials of $A(x)$
- $\alpha \in \mathbb{N}^n$ are the exponent vectors, $|\alpha| \equiv \sum_{i=1}^n \alpha_i$.

Our method can be readily used to analyze the stability of

$$\dot{x} = A(x, \gamma)x(t)$$

with parametric uncertainty γ in a compact set

We search for Lyapunov polynomials on hypercubes

Define a Hypercube as

$$\Phi_r^n := \{x \in \mathbb{R}^n : |x_i| \le r_i, i = 1, \cdots, n\}$$

Given r, we search for a $V \in \mathbb{R}[x]$ that satisfies the Lyapunov result:

If there exists a polynomial $V \in \mathbb{R}[x]$ such that

$$V(0) = 0 \quad \text{and} \quad V(x) > 0 \;\; \text{on} \;\; \Phi_r^n \setminus \{0\}$$

and

$$\nabla V(x)^T A(x) x < 0 \quad \text{on} \quad \Phi_r^n \setminus \{0\},$$

then $\dot{x}(t) = A(x)x(t)$ is asymptotically stable on

$$\{x : \{y : V(y) \le V(x)\} \subset \Phi_r^n\}.$$

The optimization problem is NP-hard

Our goal is to estimate the Region of Attraction by solving:

Problem statement:

$$\max_{V,r} \quad \|r\|_1$$
$$\text{s.t.} \quad V(0) = 0, \; V(x) > 0 \; \text{ on } \; \Phi_r^n \setminus \{0\}$$
$$\text{s.t.} \quad \nabla V(x)^T A(x) x < 0 \quad \text{on} \quad \Phi_r^n \setminus \{0\}.$$

Then $\{x : \{y : V^*(y) \leq V^*(x)\} \subset \Phi_{r^*}^n\}$ is an estimate of ROA.

A fact: It is NP-hard to decide the non-emptiness of

$$P = \{x : p_i(x_1, \cdots, x_n) \geq 0, p_i \in \mathbb{R}[x], i = 1, \cdots, m\}.$$

By *quantifier elimination* algorithms, the decision costs $m^{n+1} d^{O(n)}$ flops

We look for a sequence of convex conditions for V to exist

In this work: Instead of directly solving

$$\begin{aligned} \max_{V,r} \quad & \|r\|_1 \\ \text{s.t.} \quad & V(0) = 0,\ V(x) > 0 \ \text{ on } \ \Phi_r^n \setminus \{0\} \\ \text{s.t.} \quad & \nabla V(x)^T A(x) x < 0 \ \text{ on } \ \Phi_r^n \setminus \{0\}, \end{aligned}$$

for every bisection iterate of r, we construct a **sequence** $\{\text{LMI}_k\}$ of increasingly less conservative LMI conditions, *sufficient* for the existence of a V:

$$\begin{aligned} & V(0) = 0,\ V(x) > 0 \ \text{ on } \ \Phi_r^n \setminus \{0\} \\ & \nabla V(x)^T A(x) x < 0 \ \text{ on } \ \Phi_r^n \setminus \{0\}. \end{aligned}$$

The sequence $\{\text{LMI}_k\}$ is a result of **Polya's theorem**.

Polya's original theorem

Polya's theorem defines a systematic way to test the strict positivity of homogeneous polynomials on the positive orthant.

Polya's Theorem: If the homogeneous polynomial $F(x)$ is *strictly* positive on $\{x \in \mathbb{R}^l : x_i \geq 0, \sum_{i=1}^{l} x_i \neq 0\}$, then there exist G and H homogeneous polynomials with only positive coefficients such that

$$H(x)\, F(x) = G(x).$$

For sufficiently large d, one may take $H(x) := (\sum_{i=1}^{l} x_i)^d$.

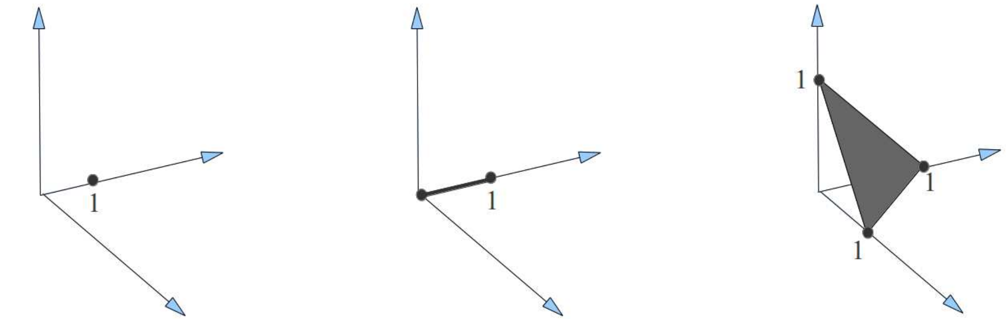
Example: $F(x) = x_1^2 - 1.1 x_1 x_2 + x_2^2 = (x_1 - 0.55x_2)^2 + 0.6975x_2^2$

$$(x_1 + x_2)^1 F(x) = x_1^3 - 0.3x_1^2 x_2 - 0.3x_1 x_2^2 + x_2^3$$
$$(x_1 + x_2)^2 F(x) = x_1^4 + 0.7x_1^3 x_2 - 0.6x_1^2 x_2^2 + 0.7x_1 x_2^3 + x_2^4$$
$$(x_1 + x_2)^3 F(x) = x_1^5 + 1.7x_1^4 x_2 + 0.1x_1^3 x_2^2 + 0.1x_1^2 x_2^3 + 1.7x_1 x_2^4 + x_2^5$$

Polya's theorem (simplex version)

Unit simplex:

$$\Delta^l := \{x \in \mathbb{R}^l : \sum_{i=1}^{l} x_i = 1, x_i \geqslant 0\}$$



Polya's Theorem (simplex version):

If matrix-valued homogeneous polynomial $F(x) \succ 0$ for all $x \in \Delta^l$, then for sufficiently large d,

$$\left(\sum_{i=1}^{l} x_i\right)^d F(x)$$

has all positive definite coefficients.

Our approach to certify positivity on hypercubes

- Polya's theorem is not readily applicable to positivity problems on hypercubes

- In the following slides

  1. For every polynomial defined on hypercube Φ_r^n, we construct an equivalent **multi-homogeneous** polynomial

  $$F(x) = \sum_{|h_1| \leq d_1} \cdots \sum_{|h_N| \leq d_n} P_{\{h_i\}} x_1^{h_1} \cdots x_n^{h_n}$$

  defined on a product of simplices

  $$x \in \Delta^2 \times \cdots \times \Delta^2 \quad \textbf{(multi-simplex)}$$

  2. We use a version of Polya's theorem which certifies positivity on the product of simplices, and as a corollary on hypercubes

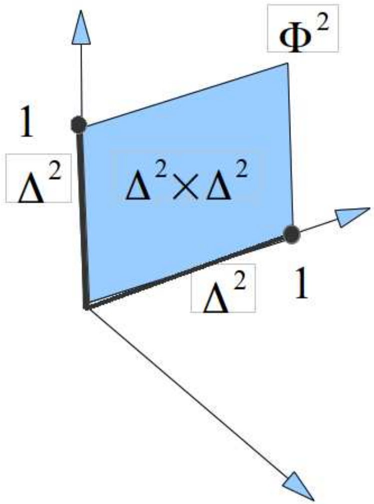
Constructing hypercubes with simplices

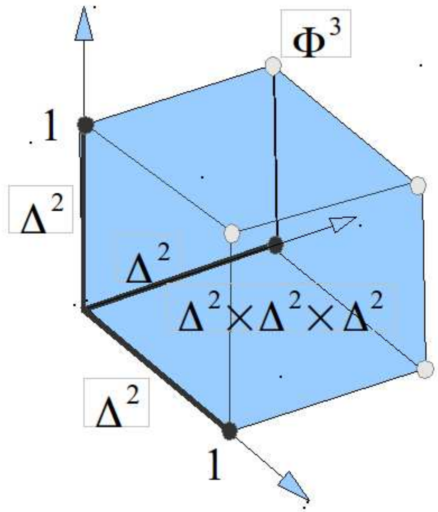
By defining the multi-homogeneous representation

$$F(x) = \sum_{|h_1| \leq d_1} \cdots \sum_{|h_N| \leq d_n} P_{\{h_i\}} x_1^{h_1} \cdots x_n^{h_n}$$

on a multi-simplex, each variable can vary independently inside $[0 \quad 1]$.

$$x \in \Delta^2 \times \cdots \times \Delta^2 \quad \textbf{(multi-simplex)}$$





Multi-homogeneous representation of $A(x), x \in \Phi_r^l$

Claim:

For every $A(x), x \in \Phi_r^l$, there exists a multi-homogeneous

$$A_H(y), y \in \underbrace{\Delta^2 \times \cdots \times \Delta^2}_{l}$$

such that

$$\{A(x) : x \in \Phi_r^l\} = \{A_H(y) : y \in \underbrace{\Delta^2 \times \cdots \times \Delta^2}_{l}\}$$

Sketch of proof:

To construct $A_H(y)$:

1. **Scaling:** Define $y_{i,1} = \frac{x_i - r_i}{2r_i} \in [0 \quad 1]$ for $i = 1, \cdots, l$
2. Define $y_{i,2} = 1 - y_{i,1}$ for $i = 1, \cdots, l$, so that $(y_{i,1}, y_{i,2}) \in \Delta^2$
3. **Shifting:** Define $B(y)$ by substituting y for x in $A(x)$
4. **Homogenizing:** Make $B(y)$ multi-homogeneous by multiplying its monomials by $(y_{i,1} + y_{i,2})^b$ with suitable b

Multi-homogeneous representation of $A(x), x \in \Phi_r^l$

Example: $A(x) = 2x_1x_2^3 - 5x_1^2x_2$

$$x \in \Phi^2 := \{x \in \mathbb{R}^2 \ : \ |x_1| \leq 2 \text{ and } |x_2| \leq 0.5\},$$

find multi-homogeneous $A_H(y)$:

$$\{A(x) : x \in \Phi^2\} = \{A_H(y) : y \in \Delta^2 \times \Delta^2\}$$

Procedure:

1. **Scaling:** Define $y_{1,1} := \frac{x_1 - 2}{4}$ and $y_{2,1} := x_2 - 0.5$
2. Define new variables $y_{1,2} := 1 - y_{1,1}$ and $y_{2,2} := 1 - y_{2,1}$
3. **Shifting:** Substitute $y_{1,1}$ and $y_{2,1}$ for x_1 and x_2 in $A(x)$:

$$B(y) := 2(4y_{1,1} + 2)(y_{2,1} + 0.5)^3 - 5(4y_{1,1} + 2)^2(y_{2,1} + 0.5)$$

4. **Homogenizing:** Make $B(y)$ multi-homogeneous:

$$A_H(y) := 2(4y_{1,1} + 2)(y_{1,1} + y_{1,2})(y_{2,1} + 0.5)^3 - 5(4y_{1,1} + 2)^2(y_{2,1} + 0.5)(y_{2,1} + y_{2,2})^2$$

Polya's theorem (multi-simplex version)

Polya's Theorem (multi-simplex version):

If matrix-valued multi-homogeneous polynomial $F(x) \succ 0$ for all $x \in \Delta^{l_1} \times \cdots \times \Delta^{l_N}$, then for some $d \in \mathbb{N}$, G has only PD coeffs.

$$G := \prod_{i=1}^{N} \left(\sum_{j=1}^{l_i} x_{i,j} \right)^d F(x)$$

Example:

$((x_1, x_2), (y_1, y_2)) \in \Delta^2 \times \Delta^2$

Iteration 1:

$F(x, y) = x_1^2 y_1 + x_1^2 y_2 - x_1 x_2 y_1 - x_1 x_2 y_2 + x_2^2 y_1 + x_2^2 y_2$

Iteration 2:

$(x_1 + x_2)(y_1 + y_2) F(x, y) =$
$x_1^3 y_1^2 + x_2^3 y_1^2 + 2x_1^3 y_1 y_2 + 2x_2^3 y_1 y_2 + x_1^3 y_2^2 + x_2^3 y_2^2$

For $d = 1$, all of the resulting coefficients are positive
$\Rightarrow F > 0$ on $\Delta^2 \times \Delta^2$

The Lyapunov inequalities

We search for a $P \in \mathbb{R}[x]$:

$$V(x) = x^T P(x) x > 0, \; x \in \Phi_r^n$$

$$\Longleftrightarrow$$

$$\boxed{P(x) \succ 0, \; x \in \Phi_r^n}$$

and

$$\dot{V}(x) = \nabla V(x)^T A(x) x < 0, \; x \in \Phi_r^n$$

$$\Longleftrightarrow$$

$$\boxed{\begin{gathered} A^T(x)P(x)+P(x)A(x)+\frac{1}{2}\left(A^T(x)\begin{bmatrix} x^T \frac{\partial P(x)}{\partial x_1} \\ \vdots \\ x^T \frac{\partial P(x)}{\partial x_n} \end{bmatrix} + \begin{bmatrix} x^T \frac{\partial P(x)}{\partial x_1} \\ \vdots \\ x^T \frac{\partial P(x)}{\partial x_n} \end{bmatrix}^T A(x) \right) \prec 0 \\ x \in \Phi_r^n \end{gathered}}$$

Applying Polya's theorem to Lyap. inequalities

Step 1 (Multi-homogenizing): Represent $P(x), x \in \Phi_r^n$ in multi-homogeneous form

$$P_H(y) = \sum_{|h_1| \leq d_1} \cdots \sum_{|h_n| \leq d_n} P_{\{h_i\}} y_1^{h_1} \cdots y_n^{h_n}, \quad y \in \Delta^2 \times \cdots \times \Delta^2$$

with degree vector $[d_1, \cdots, d_n]$.

Step 2 (Degree elevation): Multiply by $\prod_{i=1}^n (y_{i,1} + y_{i,2})^\lambda$

$$\prod_{i=1}^n (y_{i,1} + y_{i,2})^\lambda \; P_H(y) =$$

$$\sum_{|\gamma_1| \leq d_1 + \lambda} \cdots \sum_{|\gamma_n| \leq d_n + \lambda} \underbrace{\left(\sum_{|h_1| \leq d_1} \cdots \sum_{|h_n| \leq d_n} \beta_{\{h_i, \gamma_i\}} P_{\{h_i\}} \right)}_{Coefficients} y_1^{\gamma_1} \cdots y_n^{\gamma_n}$$

Applying Polya's theorem to Lyap. inequalities

Step 3 (constraints for $V > 0$): Define $\text{LMI}_\lambda^{(1)}$ with variables $P_{\{h_i\}} \in \mathbb{R}^{n\times n}$ as

$$\begin{pmatrix} \sum_{|h_1|\le d_1}\cdots\sum_{|h_n|\le d_n}\beta_{\{h_i,\gamma_{i,1}\}}P_{\{h_i\}} & & \\ & \ddots & \\ & & \sum_{|h_1|\le d_1}\cdots\sum_{|h_n|\le d_n}\beta_{\{h_i,\gamma_{i,L}\}}P_{\{h_i\}} \end{pmatrix} \succ 0$$

Step 4 (constraints for $\dot{V} < 0$): Similarly, define $\text{LMI}_\lambda^{(2)}$ by applying Polya's theorem to $\dot{V} < 0$.

Step 5 (Solving for $P_{\{h_i\}}$): If for some λ there exist $P_{\{h_i\}}$ such that

$$\begin{bmatrix} \text{LMI}_\lambda^{(1)} & 0 \\ 0 & -\text{LMI}_\lambda^{(2)} \end{bmatrix} \succ 0,$$

then $V = x^T P(x) x$ is a Lyapunov function.

Decentralization scheme: Calculating $\dot{V}$

To calculate a part of $\dot{V}$, i.e., $P(x)A(x)$:

$$P(x) = \underbrace{P_1 x_1}_{\text{CPU \#1}} + \underbrace{P_2 x_1 x_2}_{\text{CPU \#2}} + \underbrace{P_3}_{\text{CPU \#3}} \qquad A(x) = \underbrace{A_1 x_1}_{\text{CPU \#1}} + \underbrace{A_2}_{\text{CPU \#2}}$$

1. Each processor calculates its $P(x)A(x)$:

$$\begin{aligned} \text{CPU \#1} &: P_1 A_1 x_1^2 + P_1 A_2 x_1 \\ \text{CPU \#2} &: P_2 A_1 x_1^2 x_2 + P_2 A_2 x_1 x_2 \\ \text{CPU \#3} &: P_3 A_1 x_1 + P_3 A_2 \end{aligned}$$

2. Each processor multi-homogenizes its $P(x)A(x)$:

$$\begin{aligned} \text{CPU \#1} &: (P_1 A_1 + P_1 A_2) x_1^2 x_2 + (P_1 A_1 + P_1 A_2) x_1^2 \overline{x}_2 + P_1 A_2 x_1 \overline{x}_1 x_2 \\ &\quad + P_1 A_2 x_1 \overline{x}_1 \overline{x}_2 \qquad \text{(4 monomials)} \\ \text{CPU \#2} &: P_2 A_1 x_1^2 x_2 + P_2 A_2 x_1^2 x_2 + P_2 A_2 x_1 \overline{x}_1 x_2 \qquad \text{(3 monomials)} \\ \text{CPU \#3} &: (P_3 A_1 + P_3 A_2) x_1^2 x_2 + (P_3 A_1 + P_3 A_2) x_1^2 \overline{x}_2 + (P_3 A_1 + 2 P_3 A_2) x_1 \overline{x}_1 x_2 \\ &\quad + (P_3 + A_1 2 P_3 A_2) x_1 \overline{x}_1 \overline{x}_2 + P_3 A_2 \overline{x}_1^2 x_2 + P_3 A_2 \overline{x}_1^2 \overline{x_2} \qquad \text{(5 monomials)} \end{aligned}$$

Decentralization scheme: Calculating $\dot{V}$

3. Redistribute the monomials to maintain the load balance:

CPU #1 : $(P_1A_1 + P_1A_2)x_1^2x_2 + (P_1A_1 + P_1A_2)x_1^2\overline{x}_2 + P_1A_2x_1\overline{x}_1x_2 + P_1A_2x_1\overline{x}_1\overline{x}_2$ (4 monomials)

CPU #2 : $P_2A_1x_1^2x_2 + P_2A_2x_1^2x_2 + P_2A_2x_1\overline{x}_1x_2(P_3A_1 + P_3A_2)x_1^2x_2 + (P_3A_1 + P_3A_2)x_1^2\overline{x}_2$ (4 monomials)

CPU #3 : $(P_3A_1 + 2P_3A_2)x_1\overline{x}_1x_2 + (P_3 + A_12P_3A_2)x_1\overline{x}_1\overline{x}_2 + P_3A_2\overline{x}_1^2x_2 + P_3A_2\overline{x}_1^2\overline{x_2}$ (4 monomials)

Senders \ Recievers	CPU 1	CPU 2	CPU 3
CPU 1	$\emptyset$	A_1	A_1
CPU 2	A_2	$\emptyset$	A_2
CPU 3	$\emptyset$	P_3	$\emptyset$

Remarks on our decentralization

- Using Similar schemes, we perform **Scaling**, **shifting**, **degree elevation** and **differentiation** on the monomials of V and $\dot{V}$
- For arbitrary number of monomials and processors, we have designed a set of **communication rules** (refer to paper).
- *Per core* communication complexity for setting up the LMIs scales **polynomially** (n^2) in SS dimension and **exponentially** (2^d) in degrees of P and A
- We use our parallel SDP solver (Kamyar, Peet, TAC 2013) to solve the LMIs.

Example: Accuracy increases with d_p

Consider Van der Pol oscillator in reverse time

$$\dot{x}_1(t) = -x_2(t), \;\; \dot{x}_2(t) = x_1(t) + x_2(t)\left(x_1^2(t) - 1\right).$$

For hypercubes of radii

$$r_1 = [1, 1],\; r_2 = [1.5, 1.5],\; r_3 = [1.7, 1.8],\; r_4 = [1.9, 2.4]$$

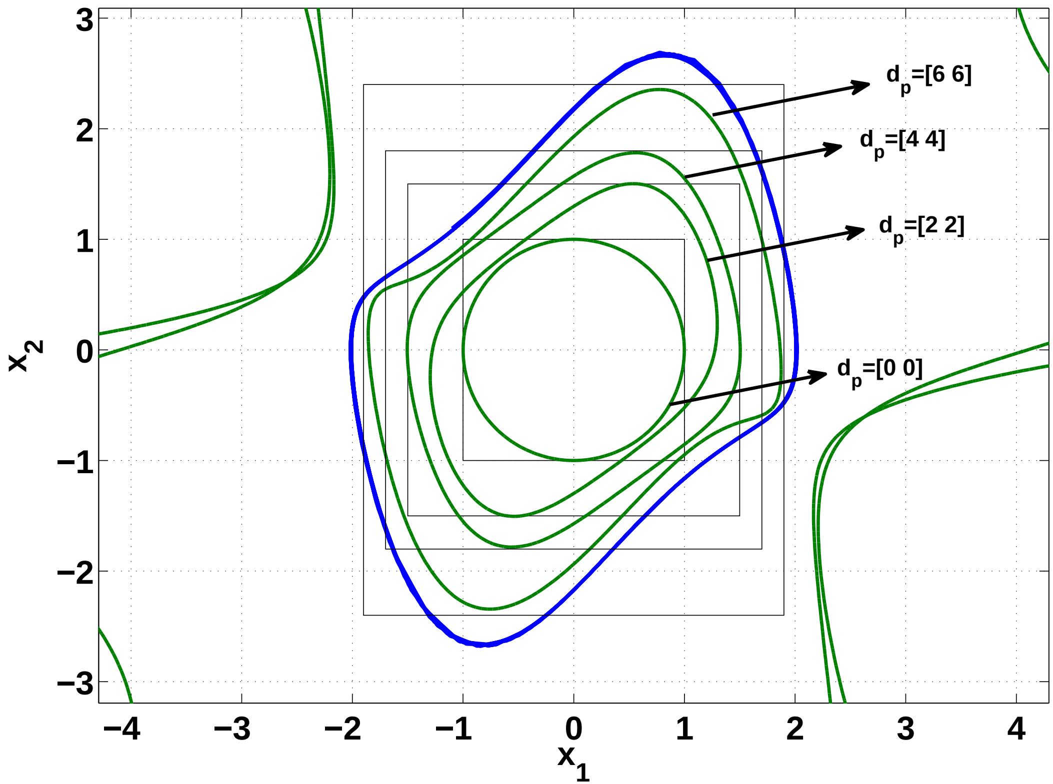
we want to solve

$\min_{P,d} \; d$ subject to

$V(x) = x^T P(x) x > 0, x \in \Phi_{r_i}$ and with $d_p = [d, d]$

$\dot{V} < 0, \quad x \in \Phi_{2,r_i}$

Example: ROA estimations for different degrees of P



Ongoing works

- Optimizing the set-up algorithm to improve its **speed-up** and its **scalability** with SS dimension and degree of V

- Extension to the analysis on convex **polyhedral** regions using affine basis functions

- **Decentralized computation** for H_2 and H_∞ controllers for uncertain systems:
  We will set-up the general form of LMIs

$$\sum_i A_i(\alpha)X(\alpha)B_i(\alpha) + B_i(\alpha)^T X(\alpha)A_i(\alpha)^T + Q_i(\alpha) \succ 0$$

- GPU implementation of set-up and solver algorithms