ISCAS 中国科学院软件研究所学术年会’2021 暨计算机科学国家重点实验室开放周

学术论文

使用差分凸规划的不变量栅栏函数生成*

Synthesizing Invariant Barrier Certificate via Difference-of-Convex Programming

Qiuye Wang[1], Mingshuai Chen[2], Bai Xue[1], Naijun Zhan[1], Joost-Pieter Katoen[2]

{wangqye, xuebai, znj}@ios.ac.cn; {chenms, katoen}@cs.rwth-Aachen.de

[1] SKLCS, Institute of Software, CAS, Beijing, China
[2] RWTH Aachen University, Aachen, Germany

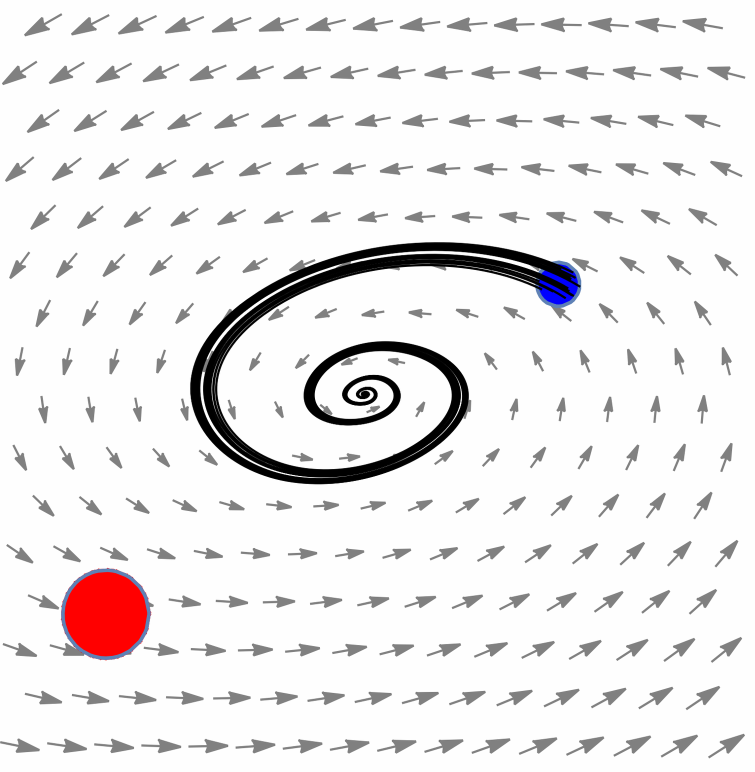
*To appear in CAV 2021

Safety of Dynamic Systems

Ordinary Differential Equations (ODEs):

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}),$$

with unique trajectory $\boldsymbol{\zeta}_{x_0}(t)$.



- Initial set X_0 : blue.
- Unsafe set X_u : red.
- Domain X : can be $\mathbf{R}^n$.
- The Safety Problem: Is the unsafe set reachable from the initial set?

Barrier Certificates (BCs) vs. Invariants

Semantic Barrier Certificate:

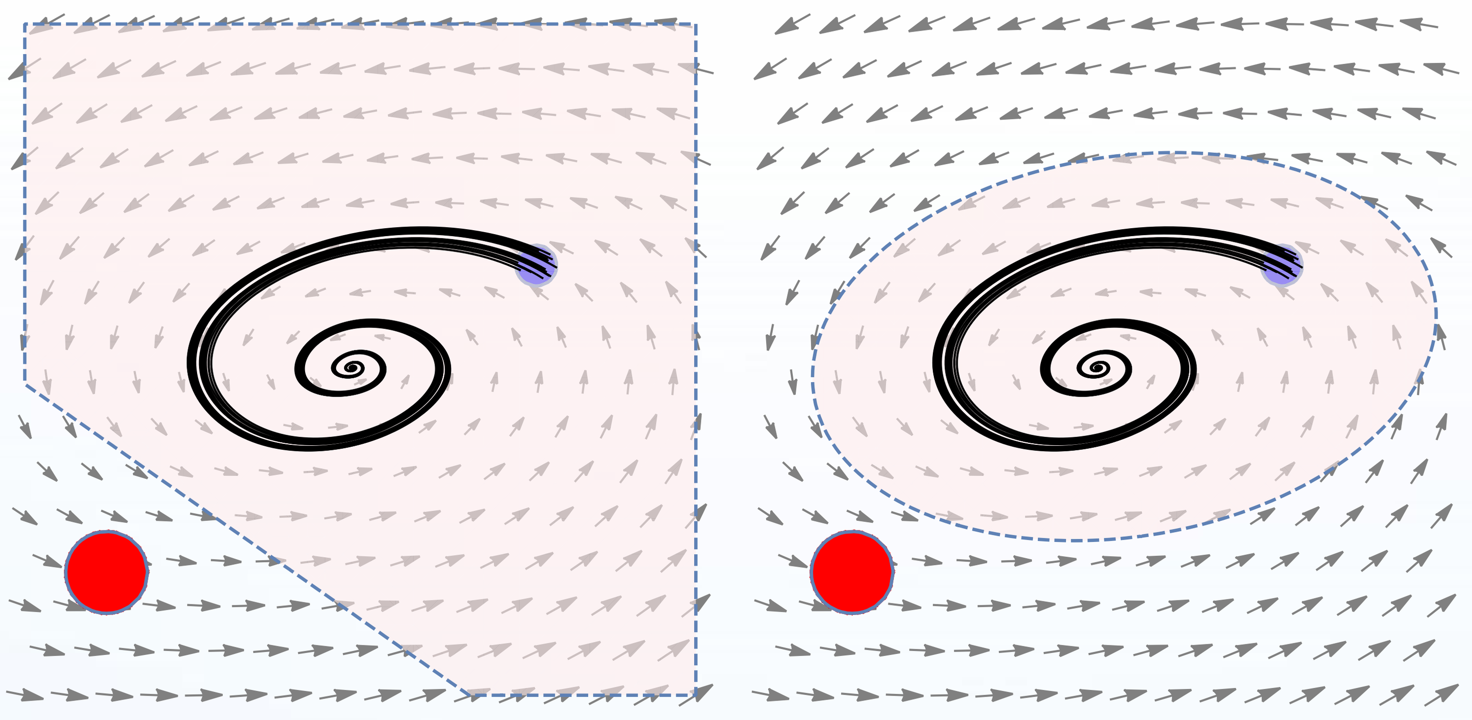
$\forall x_0 \in X_0. \forall t : B(\zeta_{x_0}(t)) \le 0$,

$\forall \boldsymbol{x} \in X_u : B(\boldsymbol{x}) > 0.$

Inductive Invariants:

$\forall x_0 \in \Psi. \forall t : \zeta_{x_0}(t) \in \Psi$,

$\forall \boldsymbol{x} \in X_u : \boldsymbol{x} \notin \Psi.$



Semantically, Inductive Invariance ⇒ BC, but not vice versa.

Practical Barrier Certificate Conditions

Semantic BC condition uses unknown trajectory function $\zeta(t)$, therefore cannot be directly used in synthesis.

Lie derivatives $L_f B(\boldsymbol{x})$: describes the change of function $B(\boldsymbol{x})$ along flow field $\boldsymbol{f}(\boldsymbol{x})$. $L_f^k B(\boldsymbol{x})$ denotes the k-th order Lie derivatives, defined as:

$$L_f^k B(\boldsymbol{x}) \triangleq \begin{cases} B(\boldsymbol{x}), & k = 0, \\ \left\langle \dfrac{\partial}{\partial \boldsymbol{x}} L_f^{k-1} B(\boldsymbol{x}), \boldsymbol{f}(\boldsymbol{x}) \right\rangle, & k > 0. \end{cases}$$

Practical BC condition:

1. $\forall \boldsymbol{x} \in X_0 : B(\boldsymbol{x}) \le 0;$
2. $\forall \boldsymbol{x} \in X_u : B(\boldsymbol{x}) > 0;$

3a. $\forall \boldsymbol{x} \in X : L_f B(\boldsymbol{x}) \le 0.$ (Original, [Prajna et al., 2004])

3b. $\forall \boldsymbol{x} \in X : L_f B(\boldsymbol{x}) = \lambda B(\boldsymbol{x}).$ (Exponential, [Kong et al., 2013])

3c. $\forall \boldsymbol{x} \in X : (B(\boldsymbol{x}) = 0) \Rightarrow (L_f B(\boldsymbol{x}) < 0).$ (Exact, [Yang et al., 2015])

……

all strictly stronger than inductive invariance.

An open problem: find a BC condition that is equivalent to inductive invariance, while still admitting efficient synthesis.

Invariant BC Condition

1. $\forall \boldsymbol{x} \in X_0 : B(\boldsymbol{x}) \le 0;$
2. $\forall \boldsymbol{x} \in X_u : B(\boldsymbol{x}) > 0;$
3. $\forall \boldsymbol{x} \in X : \bigwedge_{i=0}^{N_{B,f}} \left(\left(\bigwedge_{j=0}^{i-1} L_f^j B(\boldsymbol{x}) = 0 \right) \Rightarrow L_f^i B(\boldsymbol{x}) \le 0 \right).$

This condition is exactly equivalent to inductive invariance.

Apply sum-of-squares (SOS) transformations on 3rd condition:

$$- L_f^i B(\boldsymbol{x}) + \sum_{j=0}^{i-1} \underbrace{v_{i,j}(\boldsymbol{x})}_{unknown} \cdot \underbrace{L_f^j B(\boldsymbol{x})}_{unknown} \text{ is a SOS.}$$

Bilinearity arises!

Main difficulty: how to deal with the resulted, non-convex bilinear matrix inequality (BMI) problem?

Difference-of-Convex Programming for BMI

$$\underset{\boldsymbol{z} = (\boldsymbol{x},\ \boldsymbol{y})}{maximize}\ g(\boldsymbol{z})$$

$$s.t.\ \ \mathcal{B}(\boldsymbol{x},\ \boldsymbol{y}) \triangleq \sum_{i=1}^{m} \sum_{j=1}^{n} x_i y_j F_{i,j} + \sum_{i=1}^{m} x_i H_i + \sum_{j=1}^{n} y_j G_j + F \preccurlyeq 0$$

Using Kronecker product $\otimes$, $\mathcal{B}(\boldsymbol{x},\ \boldsymbol{y})$ can be rewritten as:

$$\mathcal{B}(\boldsymbol{x},\ \boldsymbol{y}) = (\boldsymbol{z} \otimes I)^{\mathrm{T}} M\, (\boldsymbol{z} \otimes I) + \Omega(\boldsymbol{z} \otimes I) + F,$$

where matrices M and Ω are obtained from H_i, G_j and $F_{i,j}$.

It can be proved that: $\mathcal{B}(\boldsymbol{z})$ is convex ⇔ $M \succcurlyeq 0$.

A difference-of-convex programming (DCP) procedure is given as follows:

$$\underbrace{\mathcal{B}(\boldsymbol{x},\ \boldsymbol{y})}_{non-convex} = \underbrace{\mathcal{B}^{+}(\boldsymbol{x},\ \boldsymbol{y})}_{convex} - \underbrace{\mathcal{B}^{-}(\boldsymbol{x},\ \boldsymbol{y})}_{convex} \qquad \text{(Decompose } M\text{)}$$

$$\underbrace{\mathcal{B}^{+}(\boldsymbol{z}) - \mathcal{B}^{-}(\boldsymbol{z}^k) - \mathcal{D}\mathcal{B}^{-}(\boldsymbol{z}^k)(\boldsymbol{z} - \boldsymbol{z}^k) \preccurlyeq 0}_{convex} \quad \text{(Linearize } -\mathcal{B}^{-}(\boldsymbol{x},\ \boldsymbol{y})\text{)}$$

Solvable via SDP solver!

The optimal solution is used as the next linearizing point $\boldsymbol{z}^{k+1}$.

To summary, DCP deals with the original non-convex problem via solving a series of convex programs.

Experiment Results

Our Mathematica prototype implement *SIBC* uses CSDP as the backend SDP solver.

Experiments are done in a benchmark of 24 examples, against *PENLAB* (a BMI solver using augmented Langrage method) and *SOSTOOLS* (solving LMIs with original BC condition [Prajna et al.]) , with the results organized in the following table:

	Number of accepted cases	Rate of acceptance	Average time spent on accepted cases
SIBC	20	83.3%	1.218s
PENLAB	9	37.5%	6.533s
SOSTOOLS	11	45.8%	0.215s