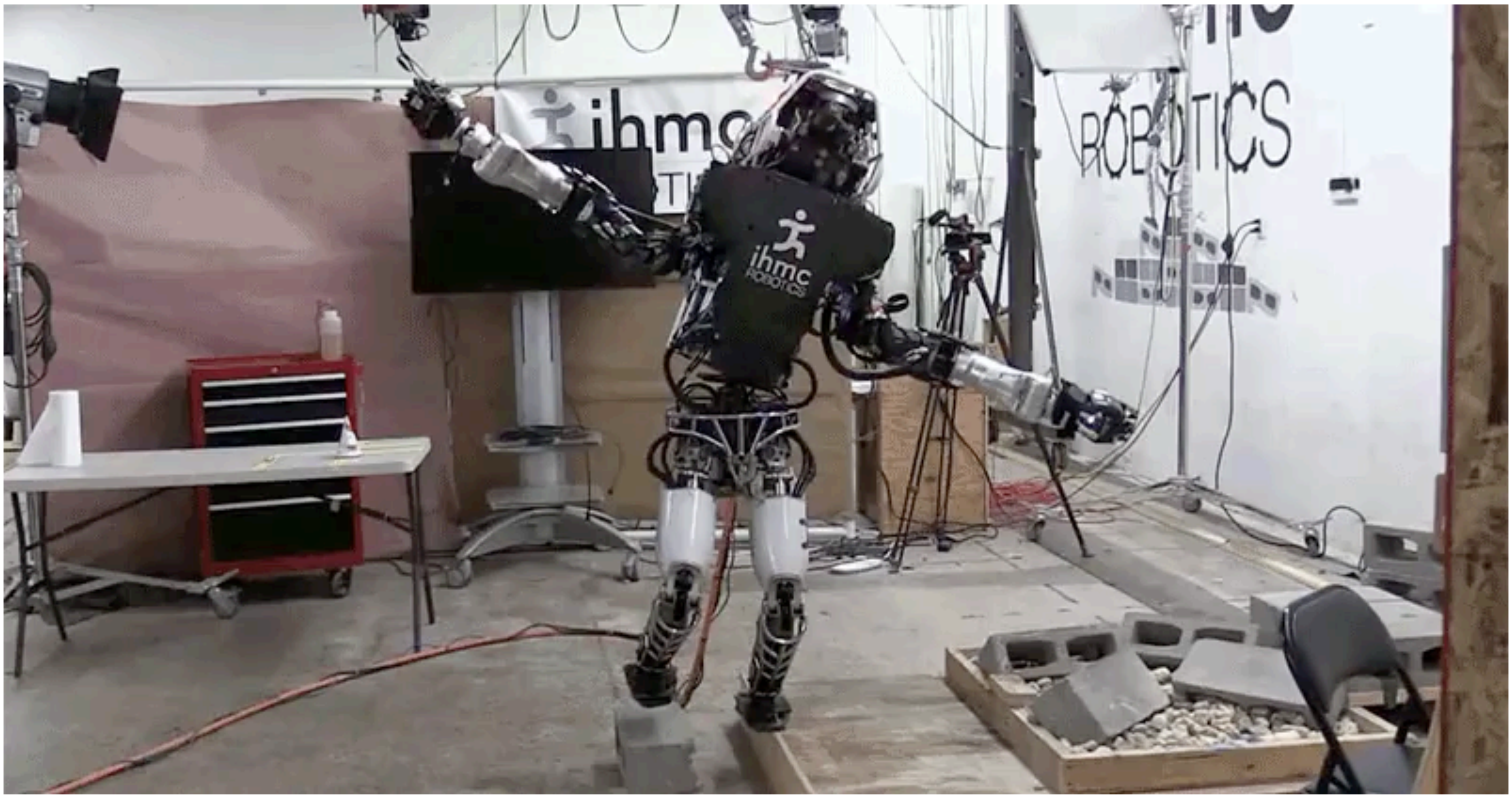
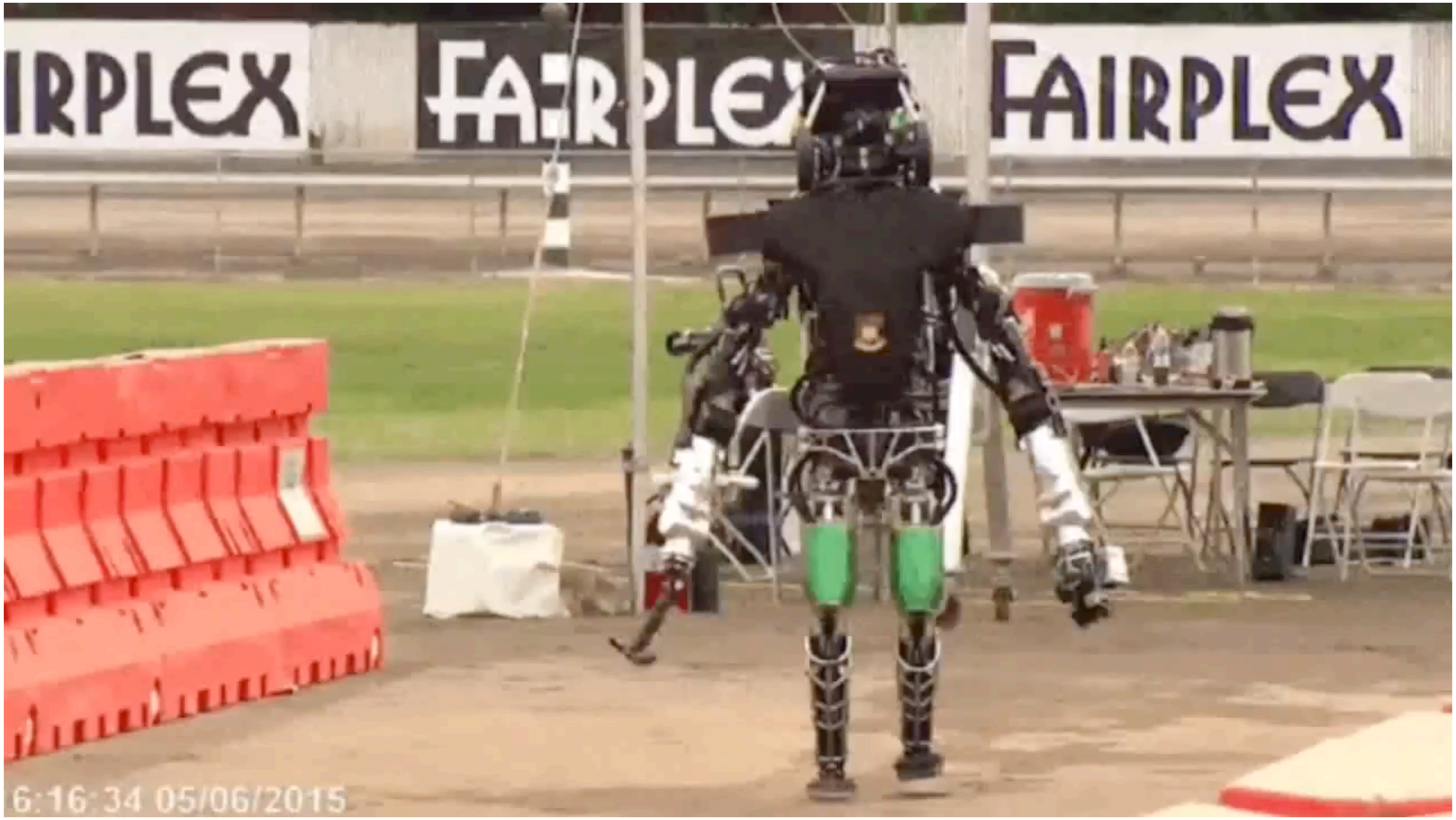
Numerical Proofs in Nonlinear Control

Sicun Gao, UCSD

Nonlinear control working

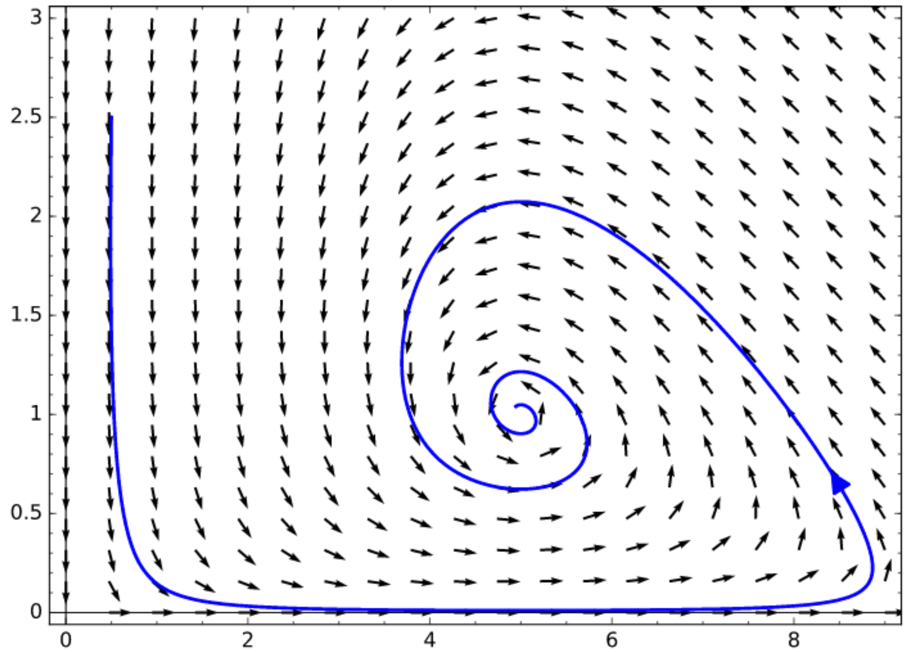


Nonlinear control not working



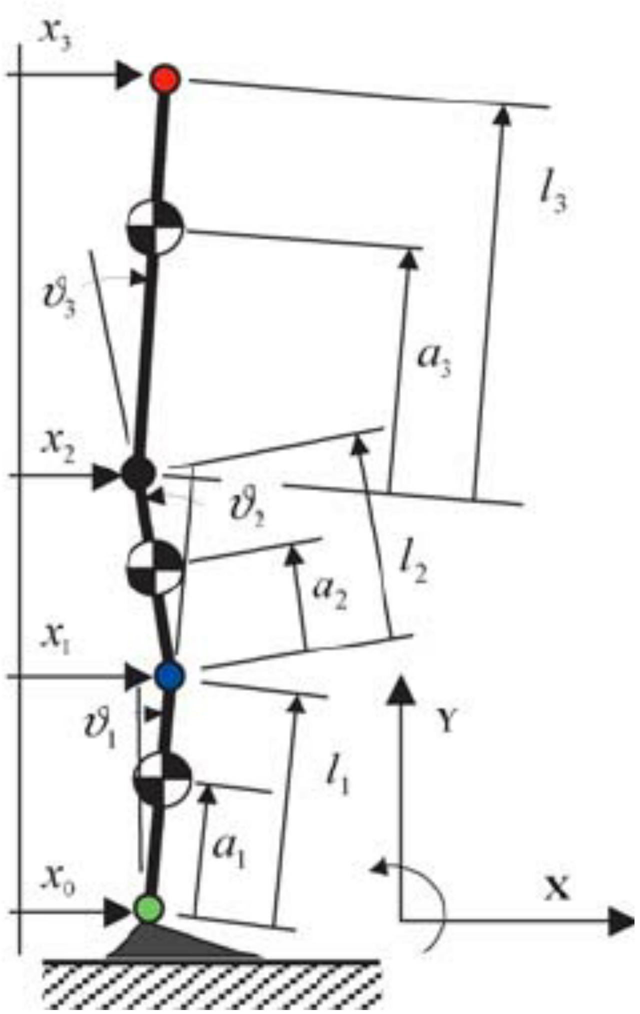
Dynamical systems are simple loops

$$x(t) = x(0) + \int_0^t f(x, u(x))\mathrm{d}s$$



$x = x_0$
$t = 0$
while true **do**
  $x = f(x, u(x)) \cdot \mathrm{d}t + x$
  $t = t + \mathrm{d}t$
end while

Dynamical systems are simple loops



$$M\left(\theta\right)\ddot{\theta} + C(\theta,\dot{\theta})\dot{\theta} + \tau(\theta) = Bu,$$

$$\theta = \left[\theta_1, \theta_2, \ldots, \theta_n\right]^{\mathrm{T}} \in \mathbb{R}^n, u \in \mathbb{R}^n$$

$$M(\theta) = \left[a_{ij}\cos\left(\theta_j - \theta_i\right)\right], M\left(\theta\right) \in \mathbb{R}^{n\times n}$$

$$C(\theta,\dot{\theta}) = \left[-a_{ij}\dot{\theta}_j\sin\left(\theta_j - \theta_i\right)\right], C(\theta,\dot{\theta}) \in \mathbb{R}^{n\times n},$$

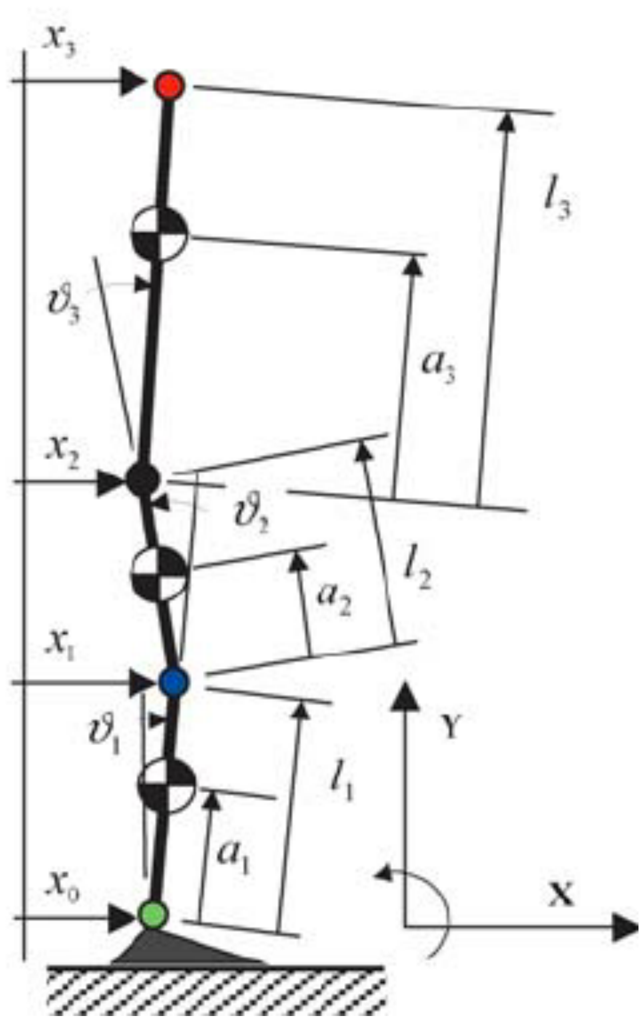
$$\tau(\theta) = \left[-b_i\sin\theta_i\right], G(\theta) \in \mathbb{R}^n,$$

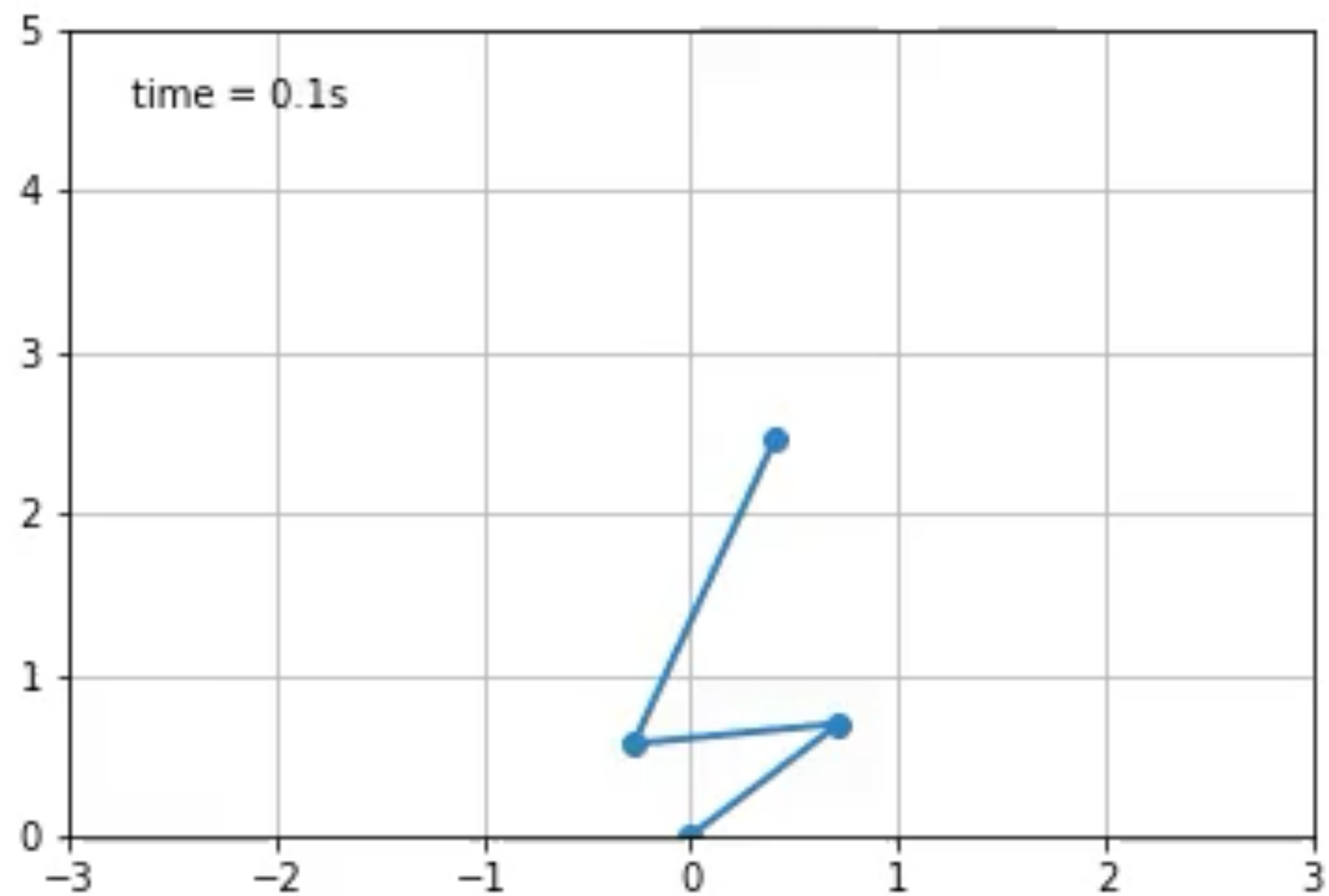
$$B = \left[1, 1, \ldots, 1\right]^{\mathrm{T}}$$

$$\begin{cases} a_{ii} = I_i + m_i\ell_{ci}^2 + \ell_i^2\sum_{k=i+1}^{n} m_k, 1 \le i \le n \\ a_{ij} = a_{ji} = m_j\ell_i\ell_{cj} + \ell_i\ell_j\sum_{k=j+1}^{n} m_k, 1 \le i < j \le n \end{cases}$$

$$b_i = \left(m_i\ell_{ci} + \ell_i\sum_{k=i+1}^{n} m_k\right)g, 1 \le i \le n,$$

Dynamical systems are simple loops



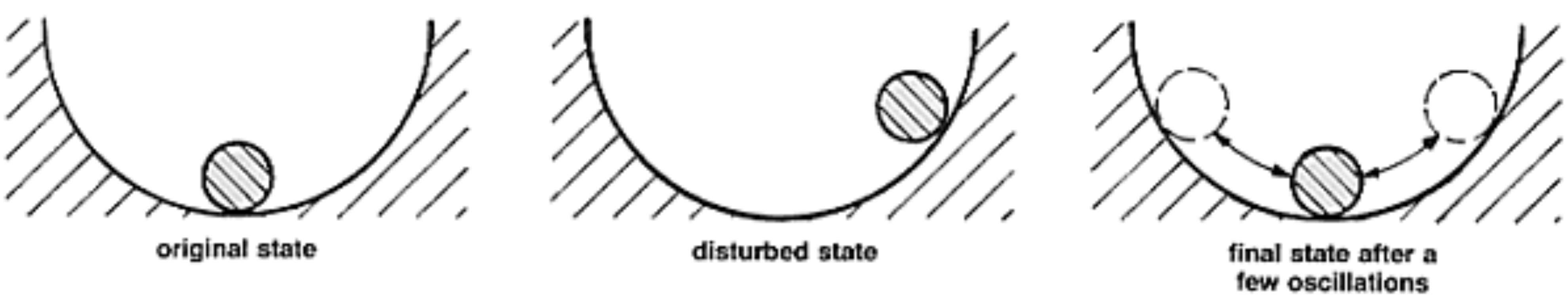


Properties we care about

- Safety: do not reach bad states

$$\forall x_0 \forall t \forall x_t \Big(x_t = F_u(x_0, t) \rightarrow \mathrm{safe}(x_t) \Big)$$

- Stability (Liveness): eventually reach good states



Properties we care about

- Safety: do not reach bad states

$$\forall x_0 \forall t \forall x_t \Big(x_t = F_u(x_0, t) \to \mathrm{safe}(x_t) \Big)$$

- Stability (Liveness-ish): eventually reach good states

$$\forall \varepsilon \exists \delta \forall x_0 \forall t \forall x_t \Big(\|x_0\| < \delta \wedge x_t = F_u(x_0, t)$$
$$\to (\|x_t\| < \varepsilon \wedge \lim_{t \to \infty} x_t = 0) \Big)$$

Recall: invariants for programs

For a discrete loop of the transition relation $T(x, x')$

- **Safety** (core part)

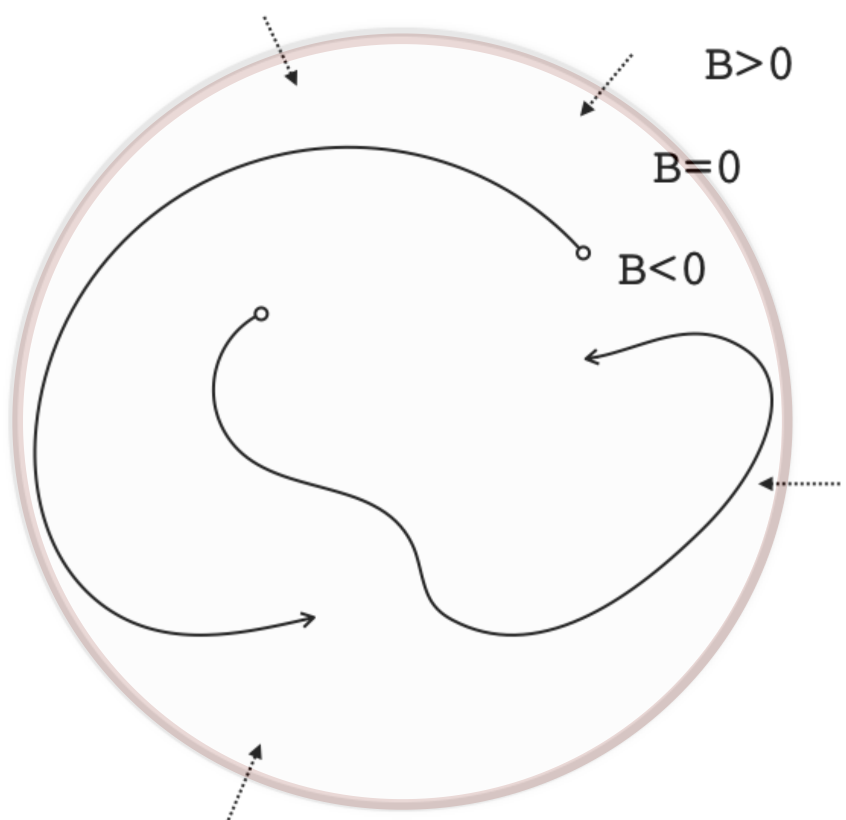
$$\Big(\mathrm{Inv}(x) \wedge \mathrm{T}(x, x')\Big) \rightarrow \mathrm{Inv}(x')$$

- **Termination** (core part)

$$\mathrm{T}(x, x') \rightarrow \Big(\mathrm{Rank}(x) > \mathrm{Rank}(x')\Big)$$

Inductive proofs over R^n

- Safety: barrier functions, differential invariants



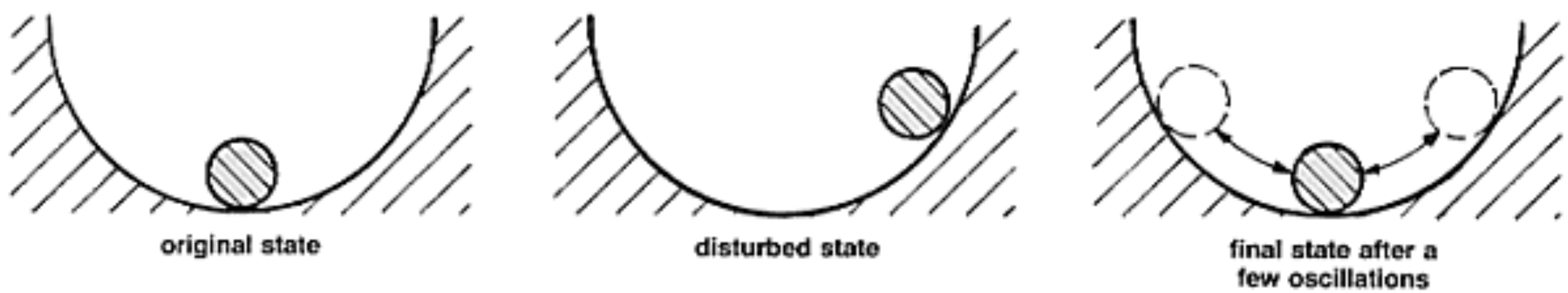
$$B(x) = 0 \rightarrow \nabla_f B(x) < 0$$

- Lie Derivative

$$\nabla_f V(x) = \sum_i \frac{\partial V}{\partial x_i}\frac{\mathrm{d}x}{\mathrm{d}t} = \sum_i \frac{\partial V}{\partial x_i} f_i(x)$$

Inductive proofs over R^n

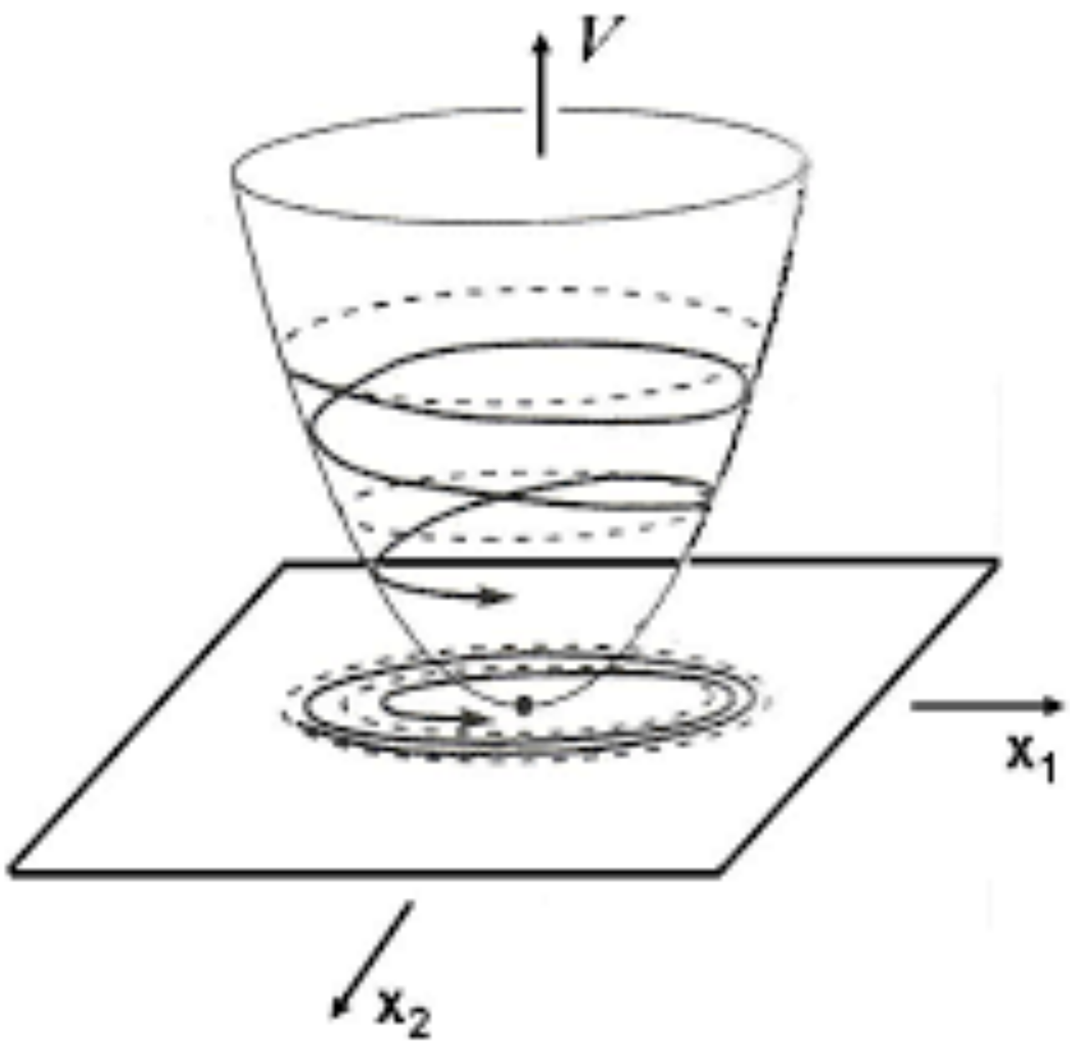
- Stability: Lyapunov functions



Find an "energy" landscape that forces stabilization (same as ranking function for termination)

Inductive proofs over R^n

- Stability (Lyapunov functions)



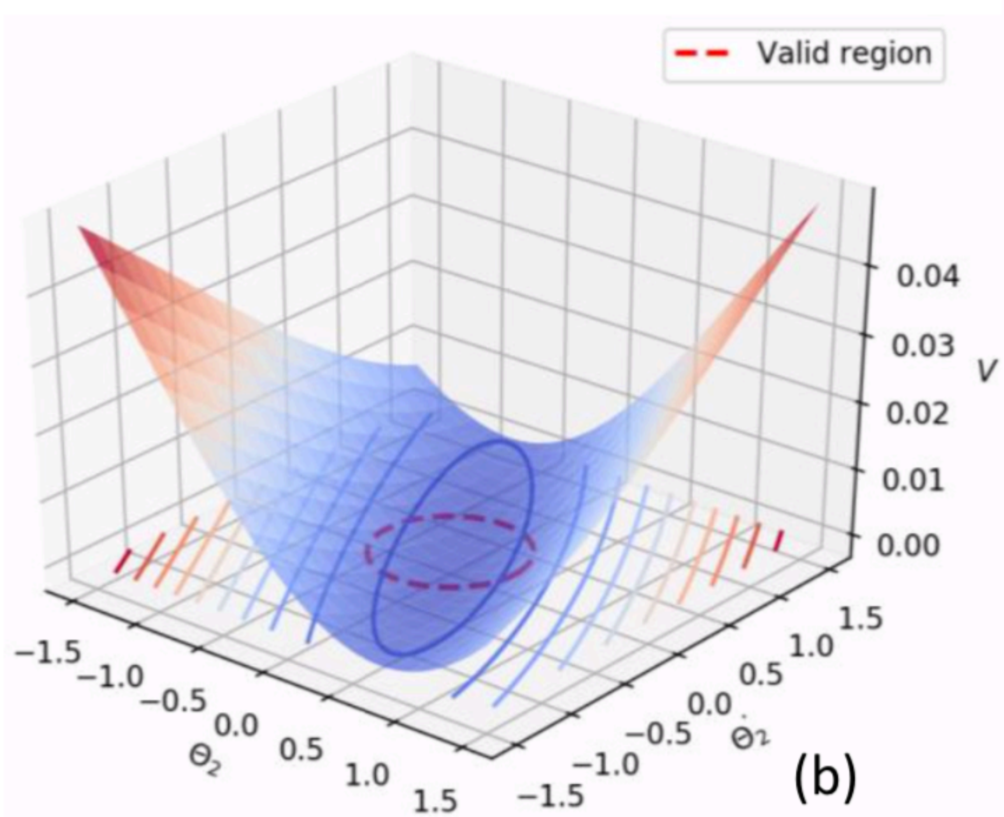
$$V(0) = 0,\ \dot{V}(0) = 0$$

$$V(x) > 0, \forall x \in D \setminus \{0\}$$

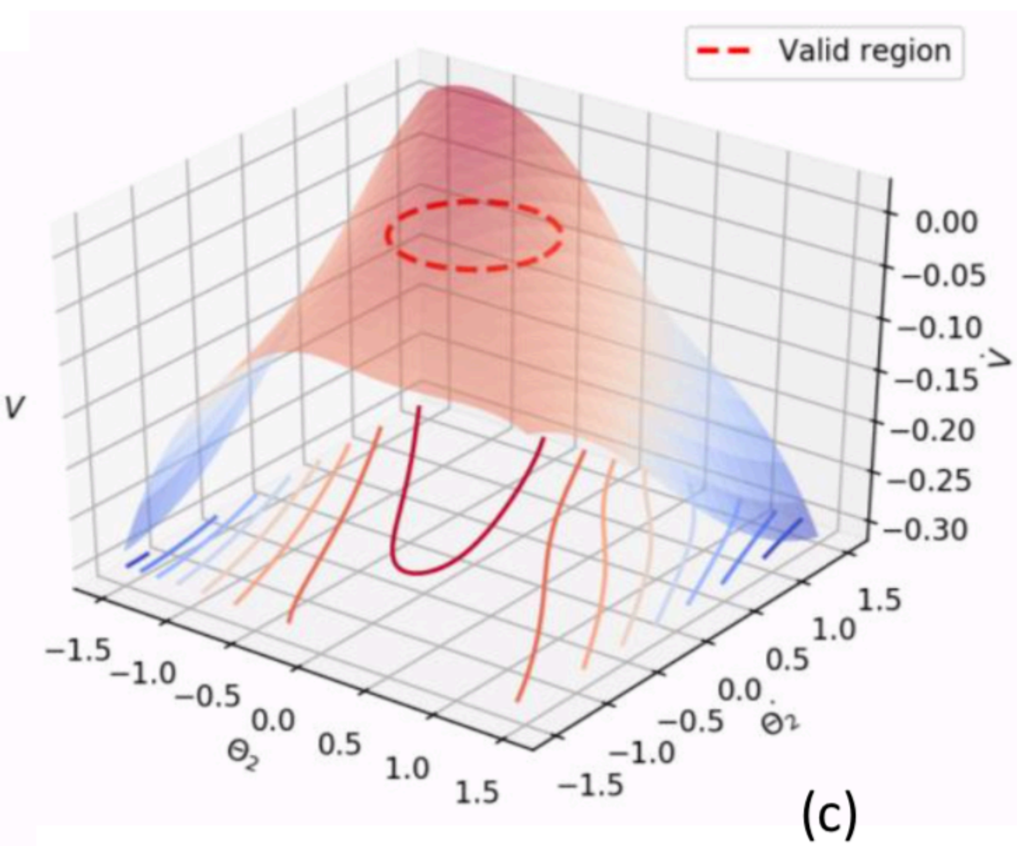
$$\nabla_f V(x) < 0, \forall x \in D \setminus \{0\}$$

Inductive proofs over R^n

- Stability: Lyapunov functions



V



$\nabla_f V$

Difficulty due to nonlinearity

- For discrete programs, finding invariants is always hard, but checking them is easy

$$\Big(\mathrm{Inv}(x) \wedge \mathrm{T}(x, x')\Big) \rightarrow \mathrm{Inv}(x')$$

$$\mathrm{T}(x, x') \rightarrow \Big(\mathrm{Rank}(x) > \mathrm{Rank}(x')\Big)$$

- Just encode the negations of these as SMT and hope for an unsat answer

Difficulty due to nonlinearity

- In the continuous case, even checking the inductive conditions is very hard
  - First-order theory over nonlinear real arithmetic

$$\nabla_f V(x) \le 0, \quad \forall x \in D \subseteq \mathbb{R}^n$$

$\mathrm{Th}\Big(\langle \mathbb{R}, \le, \{+, \times\}\rangle\Big)$ is decidable but doubly-exponential

$\mathrm{Th}_{\Sigma_1}\Big(\langle \mathbb{R}, \le, \{\sin, +, \times\}\rangle\Big)$ is undecidable

Delta-decisions

- FOL over reals is not that scary if we can allow some numerical errors in the decisions
  - Delta-decisions over reals [Gao-Avigad-Clarke, LICS'12]
- Can deal with any formula in $\langle \mathbb{R}, \leq, \mathscr{F} \rangle$ where $\mathscr{F}$ is the set of all Type 2 computable functions

Type 2 Computability

- Manipulate real numbers through natural encodings as functions over the integers (e.g. Cauchy sequences)
- A real function is Type 2 computable if an algorithm can approximate it up to arbitrary finite precisions (effective continuity)
- $\mathscr{F}$ contains polynomials, sin, cos, exp, ODEs, etc.

(pretty much all the functions we need in engineering)

Delta-decisions

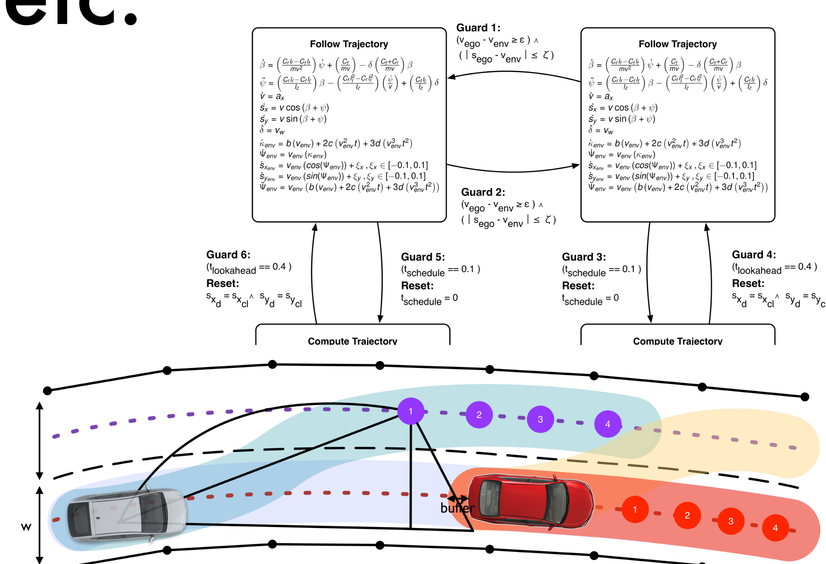
- Delta-weakening: put a formula in a positive normal form and relax all $f(x) \geq 0$ to $f(x) \geq -\delta$ where $\delta \in \mathbb{Q}^+$
  - Example: $\exists x(x = 0)$ is relaxed to $\exists x(|x| \leq \delta)$.
- We say a formula is delta-satisfiable if its delta-weakening is satisfiable. The delta-decision problem asks if a formula is unsat or delta-sat.

Delta-decisions

- Theorem: $\mathscr{L}_{\mathbb{R},\mathscr{F}}$ formulas are delta-decidable over any compact domain.
- Theorem: The complexity of delta-deciding these formulas is the same as their Boolean counterparts.
  - Complexity results for free: e.g., global multi-objective disjunctive nonlinear optimization is Σ_2^P-complete ($\mathrm{NP}^{\mathrm{NP}}$).

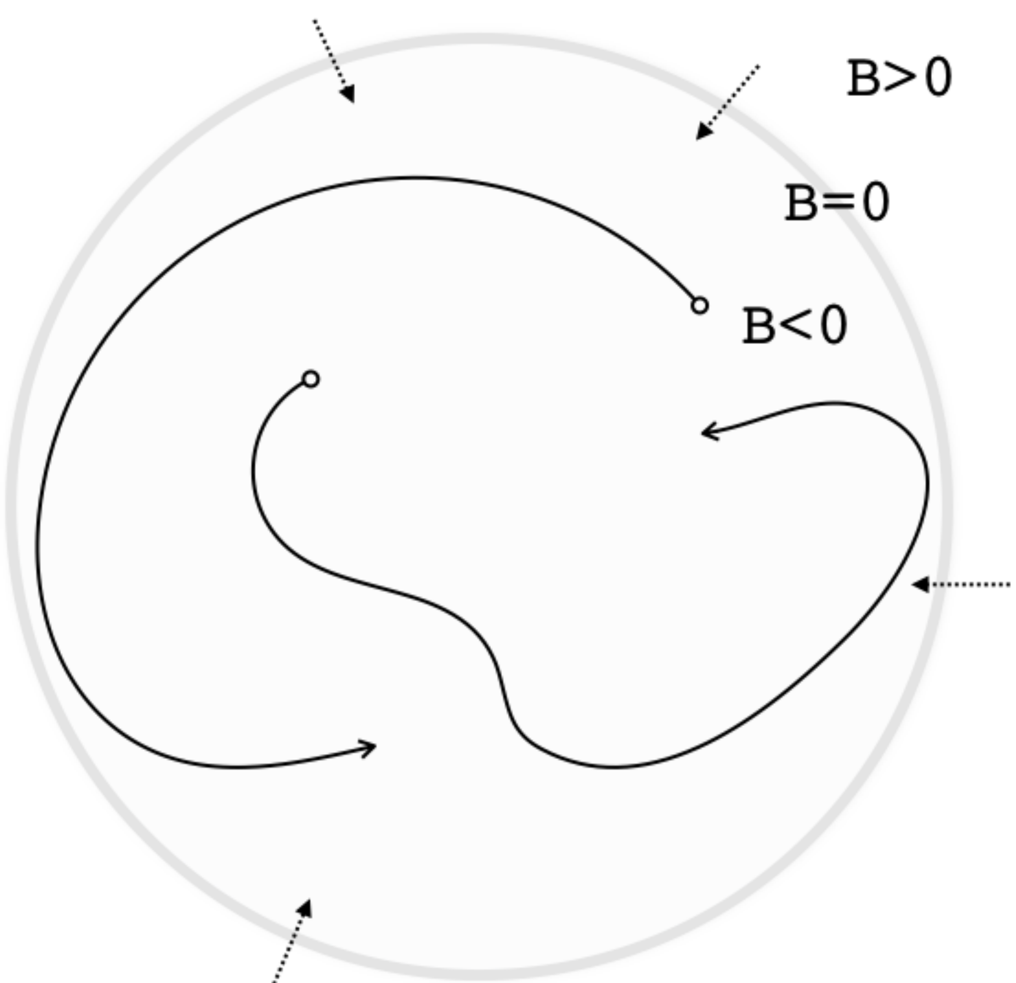
Delta-decisions

- In practice, delta-decisions are all we need for many problems in verification, optimization, etc.
- Reachability/Safety questions can be encoded, with answers "safe" or "not robustly-safe" (a delta-perturbation makes the system unsafe)
- dReal, dReach, etc.



Difficulty with induction

- However, induction fails under numerical errors!

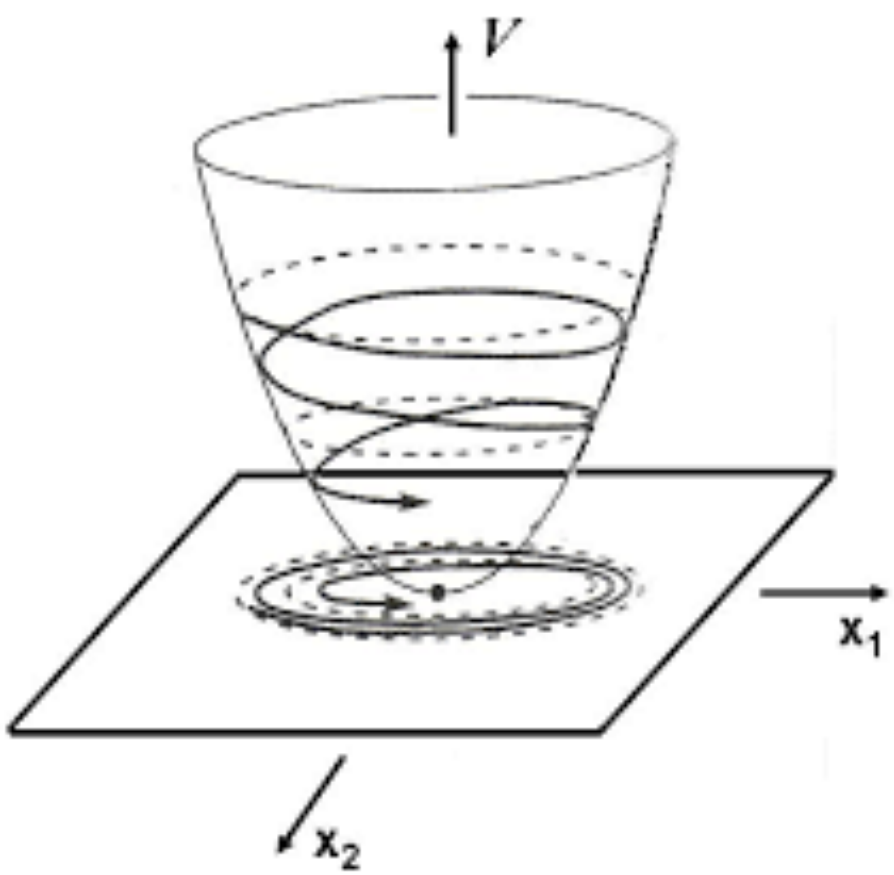


$$B(x) = 0 \to \nabla_f B(x) < 0$$

- dReal always gives spurious counterexamples

Difficulty with induction

- However, induction fails under numerical errors!



$$V(x) > 0, \forall x \in D \backslash \{0\}$$

$$\nabla_f V(x) < 0, \forall x \in D \backslash \{0\}$$

$$V(0) = 0, \dot{V}(0) = 0$$

Difficulty with induction

- But again, precise checking is unrealistic (high nonlinearity, disturbances,...)

$$
\begin{aligned}
\dot{p} &= c_1\left(2\hat{u}_1\sqrt{\frac{p}{c_{11}} - \left(\frac{p}{c_{11}}\right)^2} - \left(c_3 + c_4c_2p + c_5c_2p^2 + c_6c_2^2p\right)\right)\\
\dot{r} &= 4\left(\frac{c_3 + c_4c_2p + c_5c_2p^2 + c_6c_2^2p}{c_{13}(c_3 + c_4c_2p_{est} + c_5c_2p_{est}^2 + c_6c_2^2p_{est})(1 + i + c_{14}(r - c_{16}))} - r\right)\\
\dot{p}_{est} &= c_1\left(2\hat{u}_1\sqrt{\frac{p}{c_{11}} - \left(\frac{p}{c_{11}}\right)^2} - c_{13}\left(c_3 + c_4c_2p_{est} + c_5c_2p_{est}^2 + c_6c_2^2p_{est}\right)\right)\\
\dot{i} &= c_{15}(r - c_{16})
\end{aligned}
$$

(Example: powertrain control system)

Our fix to this problem

- We redefine the inductive proof rules over continuous domains to robustify them

Epsilon-Lyapunov and Epsilon-Barrier functions

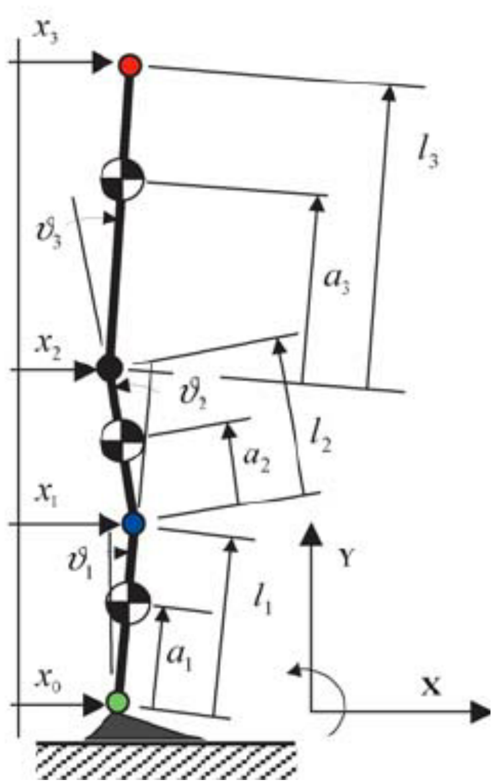
[Gao et al. CAV'19]

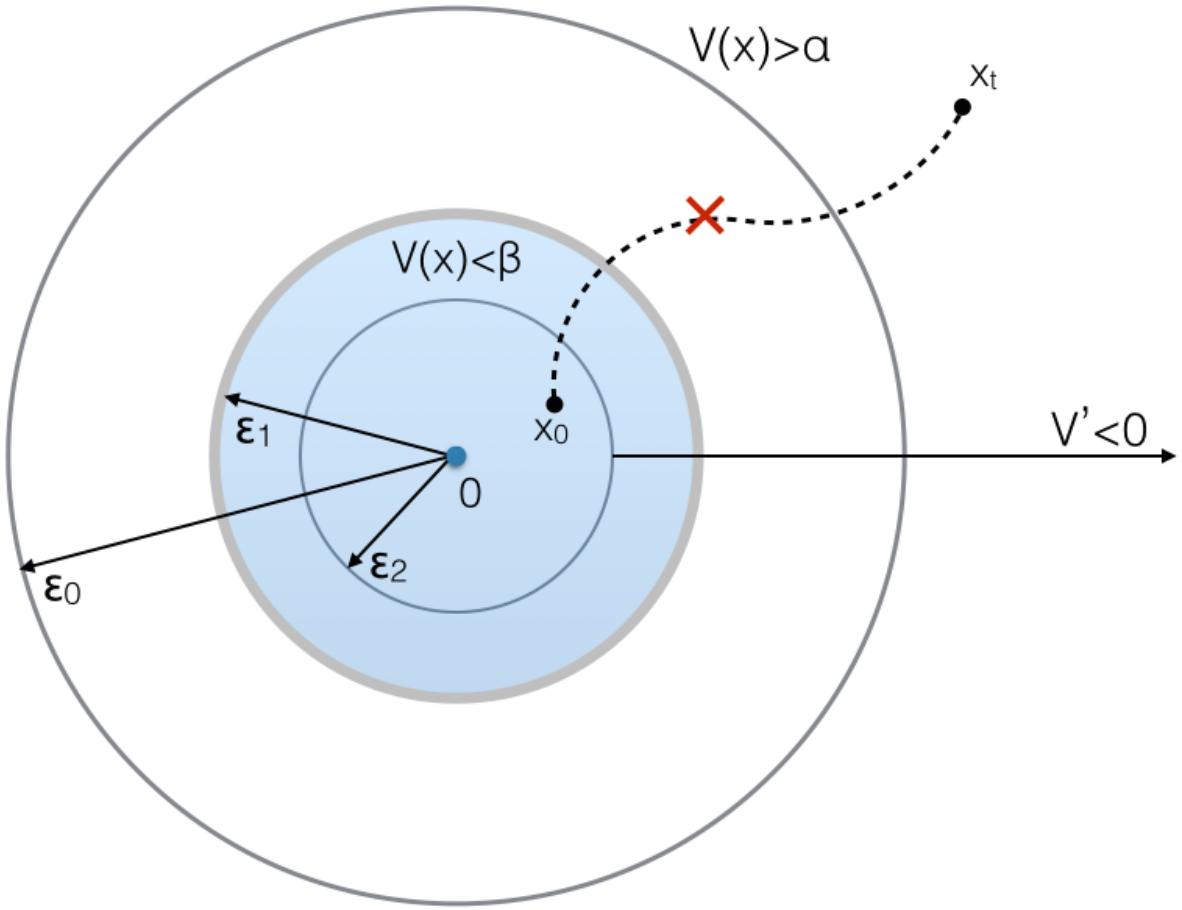
Our fix to this problem

- Three robust proof rules (epsilon-inductive conditions) for stability and safety
- For any epsilon, there exists a bound D, such that for any delta<D, delta-decision procedures are sound and complete for checking the epsilon-invariance conditions

Epsilon-Stability

- In practice, we can allow the system to oscillate within an epsilon-ball around the origin





Relaxing Stability and Strengthening LF

- Relax stability to allow small perturbation (epsilon-stability)
- Strengthen Lyapunov conditions to allow small numerical errors (epsilon-Lyapunov)
- Prove epsilon-Lyapunov implies epsilon-stability
- Prove epsilon-delta completeness

Epsilon-Stability

- Relaxation: allow the system to oscillate within an epsilon-ball around the origin

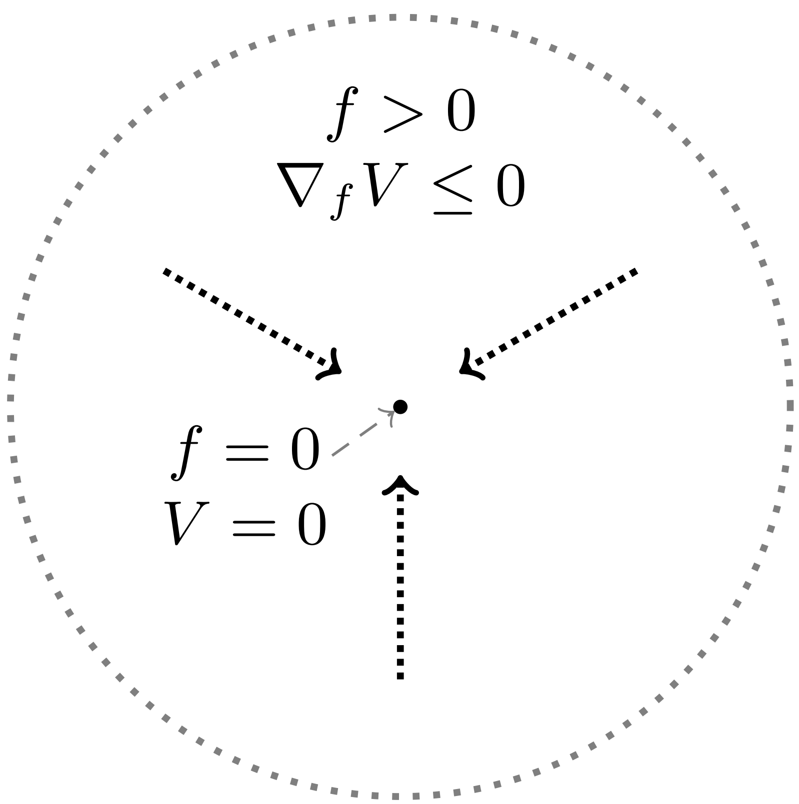
$$\mathsf{Stable}(f) \equiv_{df} \forall^{(0,\infty)}\tau\exists^{(0,\infty)}\delta\forall^{D}x_0\forall^{[0,\infty)}t\Big(\|x_0\| < \delta \rightarrow \|F(x_0,t)\| < \tau\Big)$$

$$\mathsf{Stable}_{\varepsilon}(f) \equiv_{df} \forall^{[\varepsilon,\infty)}\tau\exists^{(0,\infty)}\delta\forall^{D}x_0\forall^{[0,\infty)}t\Big(\|x_0\| < \delta \rightarrow \|F(x_0,t)\| < \tau\Big)$$

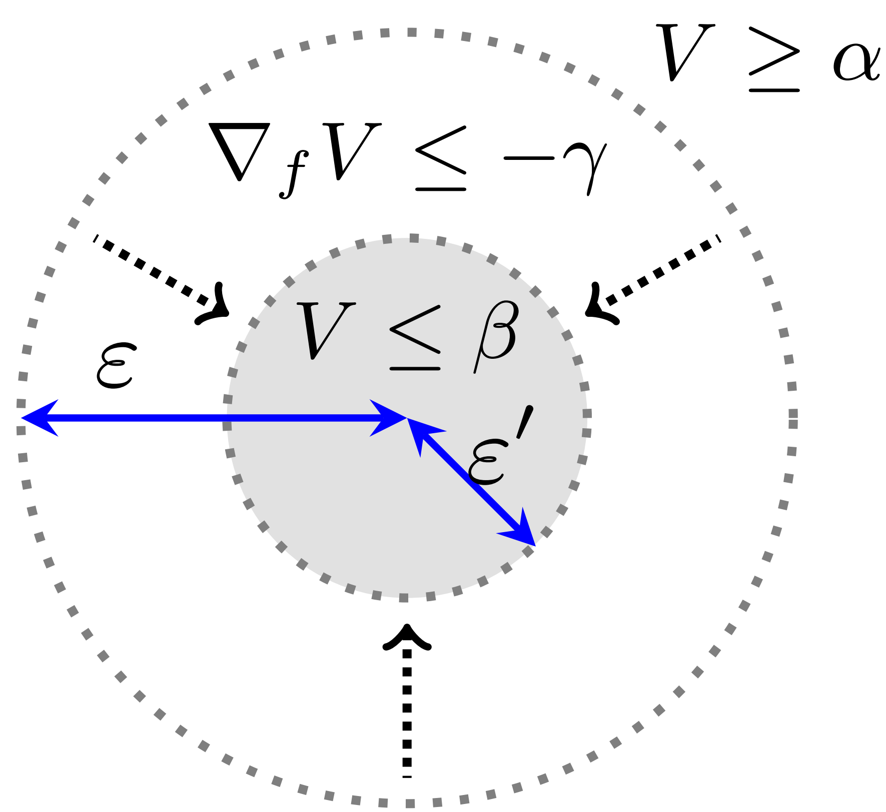
the only difference

Epsilon-Lyapunov functions

- Extend point-based requirements to neighborhoods



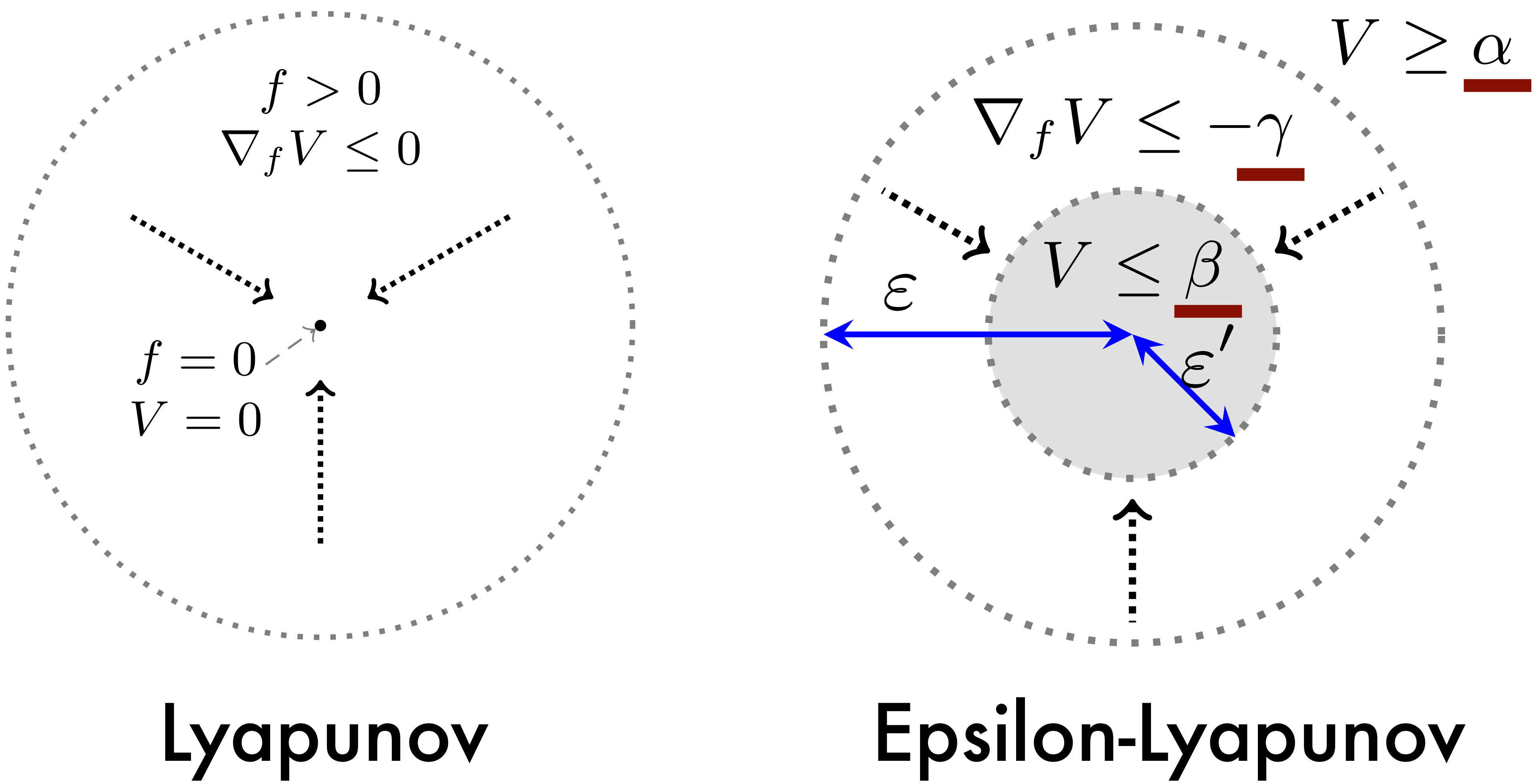
Lyapunov



Epsilon-Lyapunov

Epsilon-Lyapunov functions

- Extend point-based requirements to neighborhoods



Epsilon-Lyapunov functions

- Extend point-based requirements to neighborhoods

$$\mathsf{LF}(f,V) \equiv_{df} \ (V(0)=0) \wedge (f(0)=0) \wedge \forall^{D\setminus\{0\}} x \left(V(x) > 0 \wedge \nabla_f V(x) \leq 0 \right)$$

$$\mathsf{LF}_\varepsilon(f,V) \equiv_{df} \ \exists^{(0,\varepsilon)}\varepsilon' \exists^{(0,\infty)}\alpha \exists^{(0,\alpha)}\beta \exists^{(0,\infty)}\gamma$$

$$\forall^{D\setminus\mathcal{B}_\varepsilon} x \left(V(x) \geq \underline{\alpha} \right) \wedge \forall^{\mathcal{B}_{\varepsilon'}} x \left(V(x) \leq \underline{\beta} \right)$$

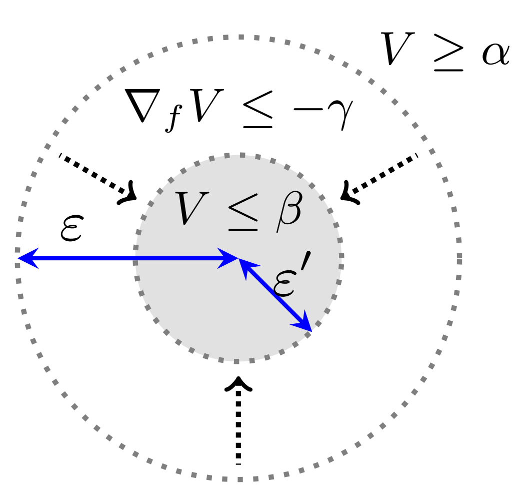
$$\wedge \forall^{D\setminus\mathcal{B}_{\varepsilon'}} x \left(\nabla_f V(x) \leq \underline{-\gamma} \right)$$

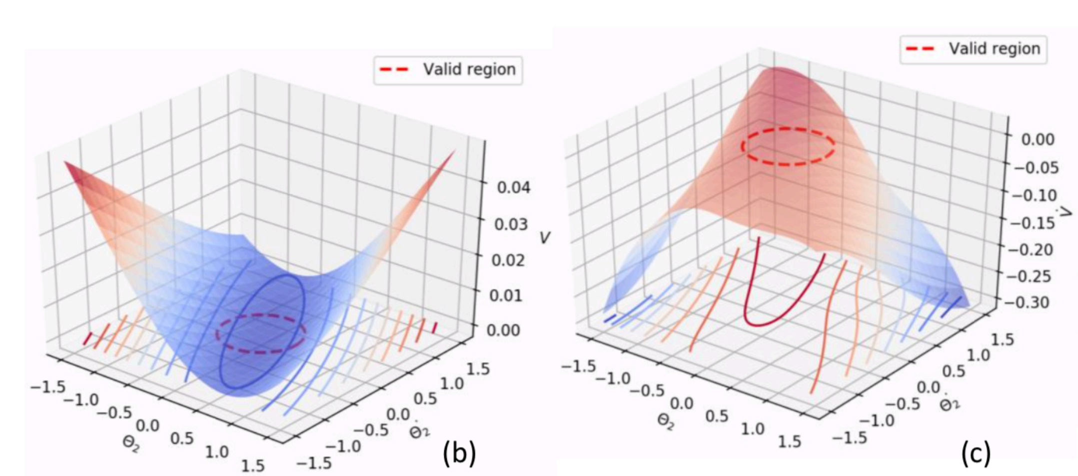
Epsilon-Lyapunov functions

Theorem 1. *If there exists an ε-Lyapunov function V for a dynamical system defined by f, then the system is ε-stable. Namely,* $\mathsf{LF}_\varepsilon(f, V) \rightarrow \mathsf{Stable}_\varepsilon(f)$.

Theorem 2 (Soundness). *If a δ-complete decision procedure confirms that* $\mathsf{LF}_\varepsilon(f, V)$ *is* true *then V is indeed an ε-Lyapunov function, and f is ε-stable.*

Theorem 3 (Relative Completeness). *For any $\varepsilon \in \mathbb{R}_+$, if* $\mathsf{LF}_\varepsilon(f, V)$ *is true then there exists $\delta \in \mathbb{Q}_+$ such that any δ-complete decision procedure must return that* $\mathsf{LF}_\varepsilon(f, V)$ *is* true.



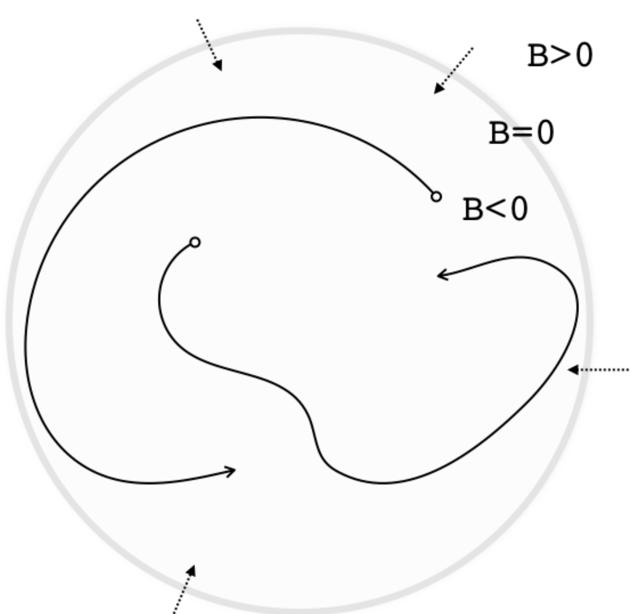


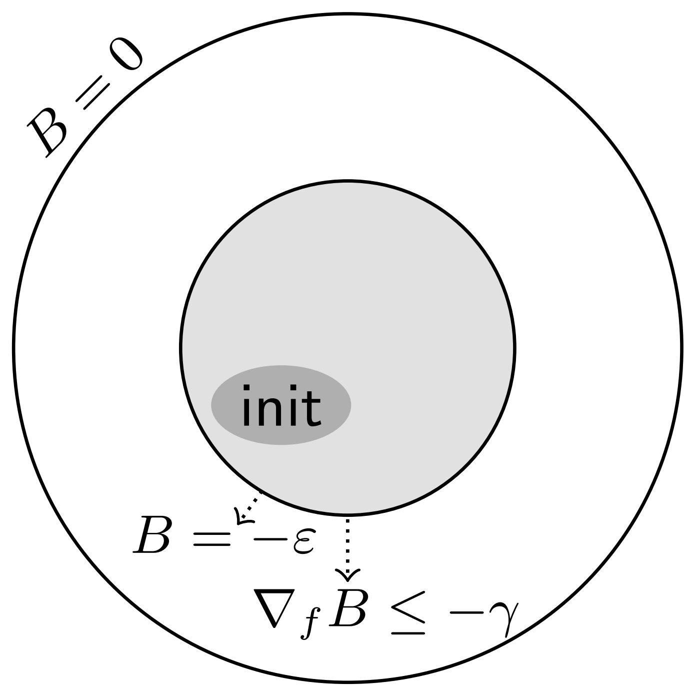
Safety and epsilon-barrier functions

- Similarly, we define two robust barrier function conditions that are stronger, sufficient for the normal notion of safety
- Prove epsilon-delta completeness

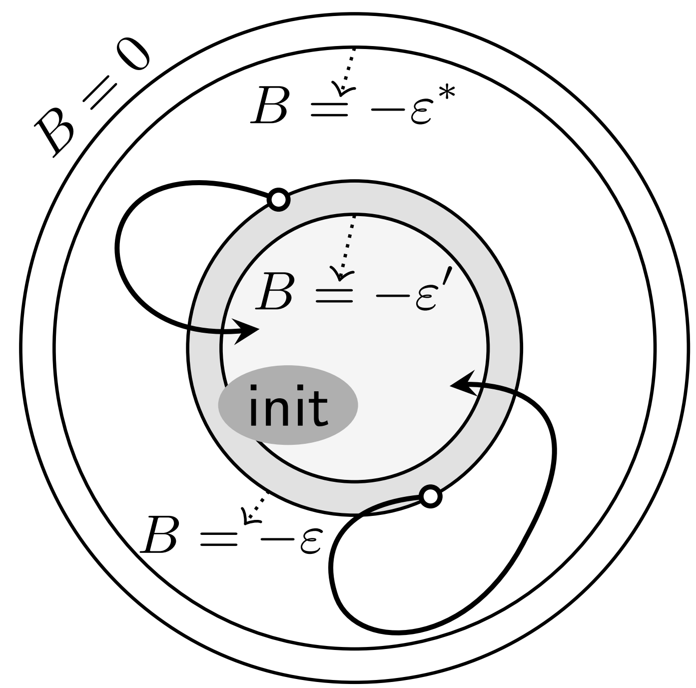
Safety and epsilon-barrier functions

- Ensure that the system goes back into the invariant set “near” the boundary





Type 1 ε-Barrier



Type 2 ε-Barrier

Safety and epsilon-barrier functions

Type 1:

$$\text{Barrier}_{\varepsilon}(f, \text{init}, B) \equiv_{df} \forall^{D} x\Big(\text{init}(x) \rightarrow B(x) \leq -\varepsilon\Big)$$
$$\wedge\, \exists^{(0,\infty)}\gamma \forall^{D} x\Big(B(x) = -\varepsilon \rightarrow \nabla_f B(x) \leq -\gamma\Big)$$

Type 2:

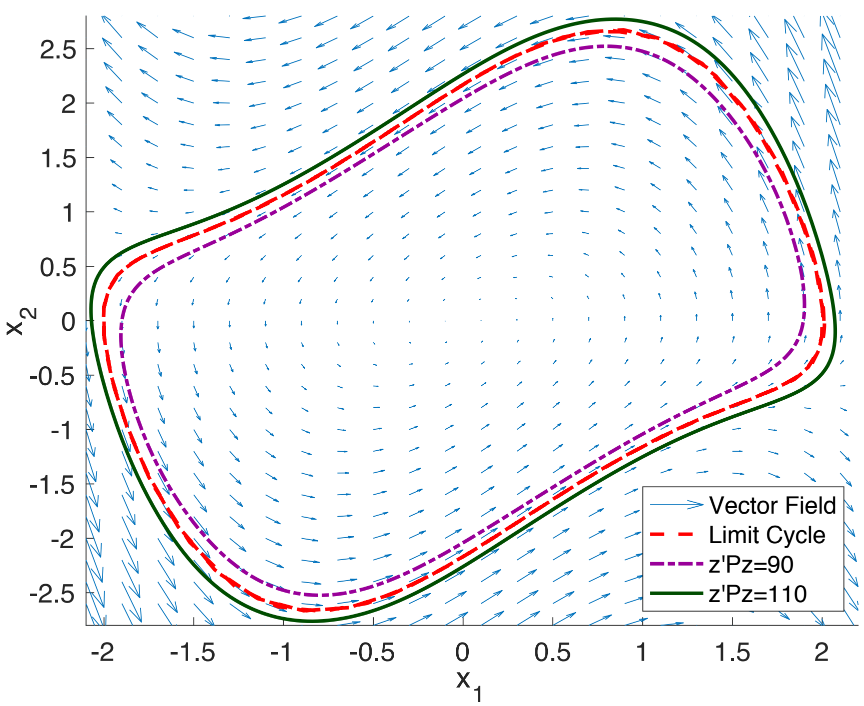
$$\text{Barrier}_{T,\varepsilon}(f, \text{init}, B) \equiv_{df} \forall^{D} x\Big(\text{init}(x) \rightarrow B(x) \leq -\varepsilon\Big)$$
$$\wedge\, \exists^{(0,\varepsilon]}\varepsilon^{*} \forall^{D} x \forall^{[0,T]} t\Big((B(x) = -\varepsilon) \rightarrow B(F(x,t)) \leq -\varepsilon^{*}\Big)$$
$$\wedge\, \exists^{(\varepsilon,\infty)}\varepsilon' \forall^{D} x\Big((B(x) = -\varepsilon) \rightarrow B(F(x,T)) \leq -\varepsilon'\Big)$$

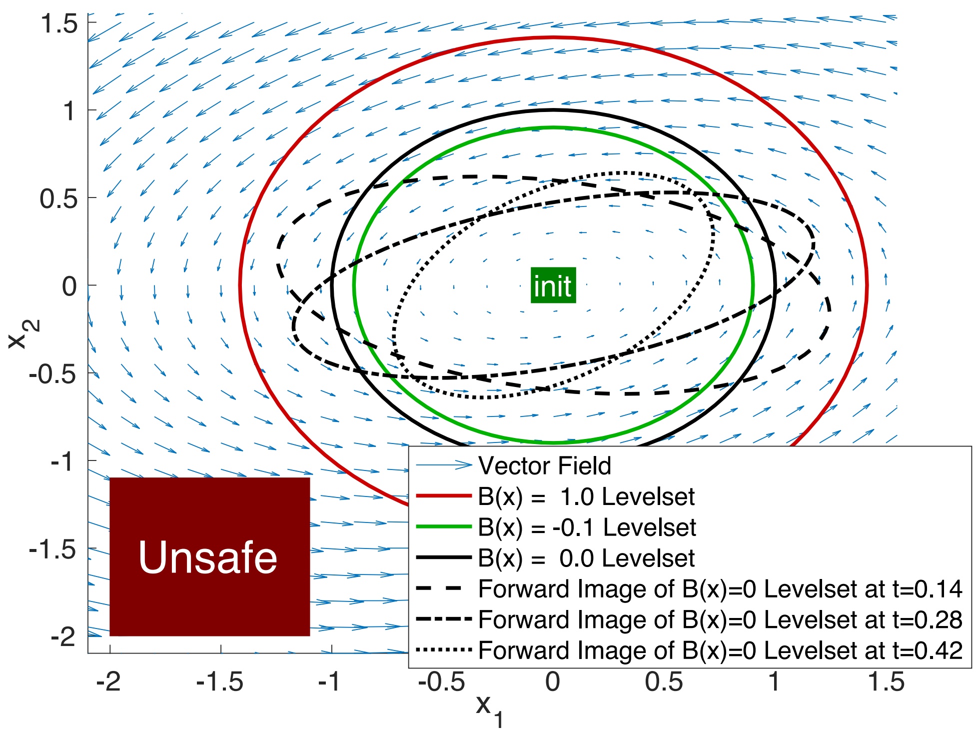
Safety and epsilon-barrier functions

Theorem 4. *For any* $\varepsilon \in \mathbb{R}_+$, $\mathsf{Barrier}_\varepsilon(f, \mathsf{init}, B) \to \mathsf{Safe}(f, \mathsf{init}, B)$.

Theorem 6. *For any* $T, \varepsilon \in \mathbb{R}_+$, $\mathsf{Barrier}_{T,\varepsilon}(f, \mathsf{init}, B) \to \mathsf{Safe}(f, \mathsf{init}, B)$.

Theorem 7. *For any* $\varepsilon \in \mathbb{R}_+$, *there exists* $\delta \in \mathbb{Q}_+$ *such that* $\mathsf{Barrier}_{T,\varepsilon}(f, \mathsf{init}, B)$ *is a* δ*-robust formula.*





Experiments (various nonlinear systems)

Example	α	β	γ	ε	ε'	Time (s)
T.R. Van der Pol	2.10×10^{-23}	1.70×10^{-23}	10^{-25}	10^{-12}	5×10^{-13}	0.05
Norm. Pend.	7.07×10^{-23}	3.97×10^{-23}	10^{-50}	10^{-12}	5×10^{-13}	0.01
Moore-Greitzer	2.95×10^{-19}	2.55×10^{-19}	10^{-20}	10^{-10}	5×10^{-11}	0.04

Table 1: Results for the ε-Lyapunov functions. Each Lyapunov function is of the form $z^T Pz$, where z is a vector of monomials over the state variables. We report the constant values satisfying the ε-Lyapunov conditions, and the time that verification of each example takes (in seconds).

Example	ℓ	ε	γ	degree(z)	size of P	Time (s)
T.R. Van der Pol	90	10^{-5}	10^{-5}	3	9×9	6.47
Norm. Pend.	$[0.1, 10]$	10^{-2}	10^{-2}	1	2×2	0.08
Moore-Greitzer	$[1.0, 10]$	10^{-1}	10^{-1}	4	5×5	13.80
PTC	0.01	10^{-5}	10^{-5}	2	14×14	428.75

Experiments (powertrain control)

Example	ℓ	ε	γ	degree(z)	size of P	Time (s)
PTC	0.01	10^{-5}	10^{-5}	2	14×14	428.75

$$
\begin{aligned}
\dot{p} &= c_1\left(2\hat{u_1}\sqrt{\frac{p}{c_{11}} - \left(\frac{p}{c_{11}}\right)^2} - \left(c_3 + c_4c_2p + c_5c_2p^2 + c_6c_2^2p\right)\right)\\
\dot{r} &= 4\left(\frac{c_3 + c_4c_2p + c_5c_2p^2 + c_6c_2^2p}{c_{13}(c_3 + c_4c_2p_{est} + c_5c_2p_{est}^2 + c_6c_2^2p_{est})(1 + i + c_{14}(r - c_{16}))} - r\right)\\
\dot{p}_{est} &= c_1\left(2\hat{u_1}\sqrt{\tfrac{p}{c_{11}} - \left(\tfrac{p}{c_{11}}\right)^2} - c_{13}\left(c_3 + c_4c_2p_{est} + c_5c_2p_{est}^2 + c_6c_2^2p_{est}\right)\right)\\
\dot{i} &= c_{15}(r - c_{16})
\end{aligned}
$$

From verification to synthesis

- Once the proof rules can be checked, we can further automate control design.

$$\exists p\, \exists q\, \forall x\ \Phi(f, u(p, x), V(q, x))$$

- Find parameters for control $u(p, x)$ and proof certificate $V(q, x)$ so that the inductive conditions in Φ are true over all states.

From verification to synthesis

$$\exists p \exists q \forall x \; \Phi(f, u(p, x), V(q, x))$$

- In general we can try solving these formulas in the delta-decision framework. [Kong et al. CAV'18]
- But it is very hard to scale, because p and especially q can be very high-dimensional.

From verification to synthesis

$$\exists p \, \exists q \, \forall x \;\; \Phi(f, u(p, x), V(q, x))$$

- We need cheap algorithms to search for p and q.
- We can often afford full SMT solving over x.
- Also, the form of u and V matter a lot.

From verification to synthesis

$$\exists p \, \exists q \, \forall x \;\; \Phi(f, u(p, x), V(q, x))$$

- The standard approach is to assume V is a sum-of-squares polynomial and the search can be done through semidefinite programming.
- In practice, it is very brittle. (checking rarely passes)

Crazy attempt: use neural networks

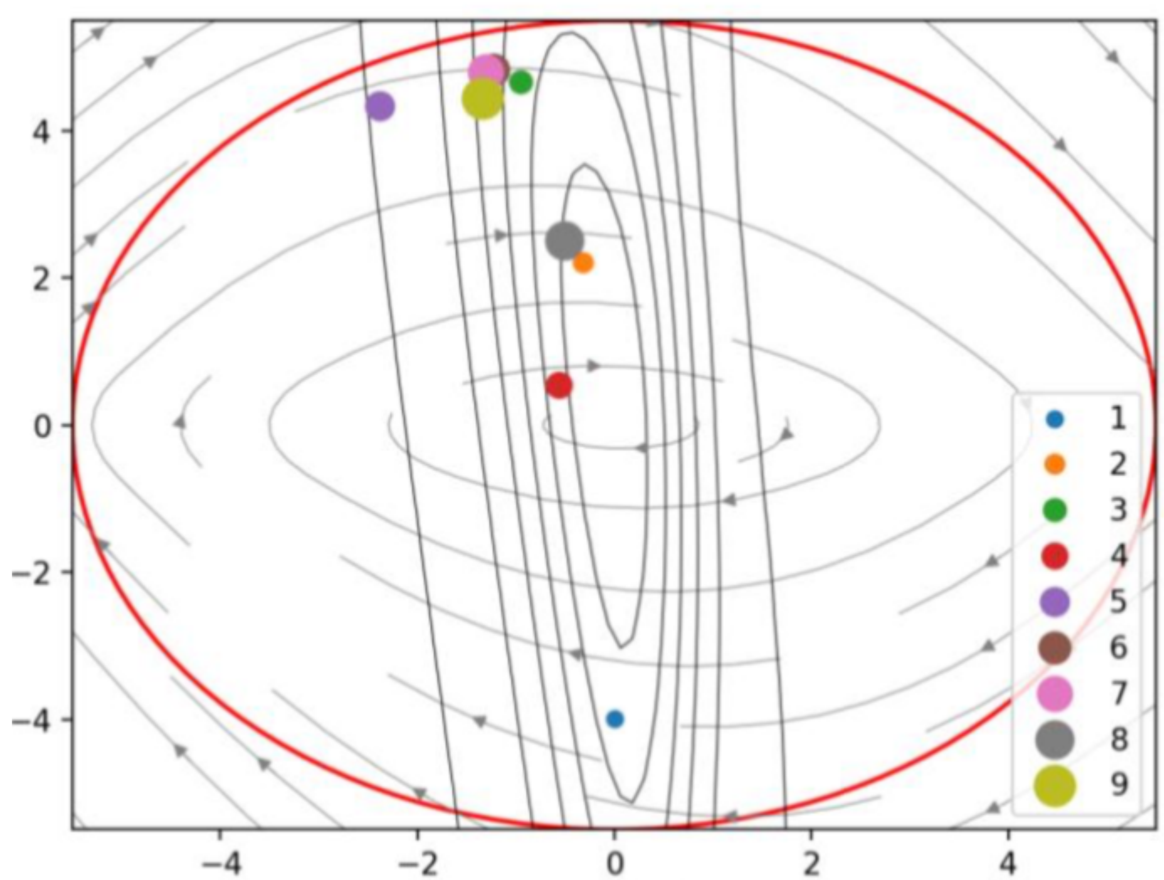
$$\exists p \, \exists q \, \forall x \;\; \Phi(f, u(p,x), V(q,x))$$

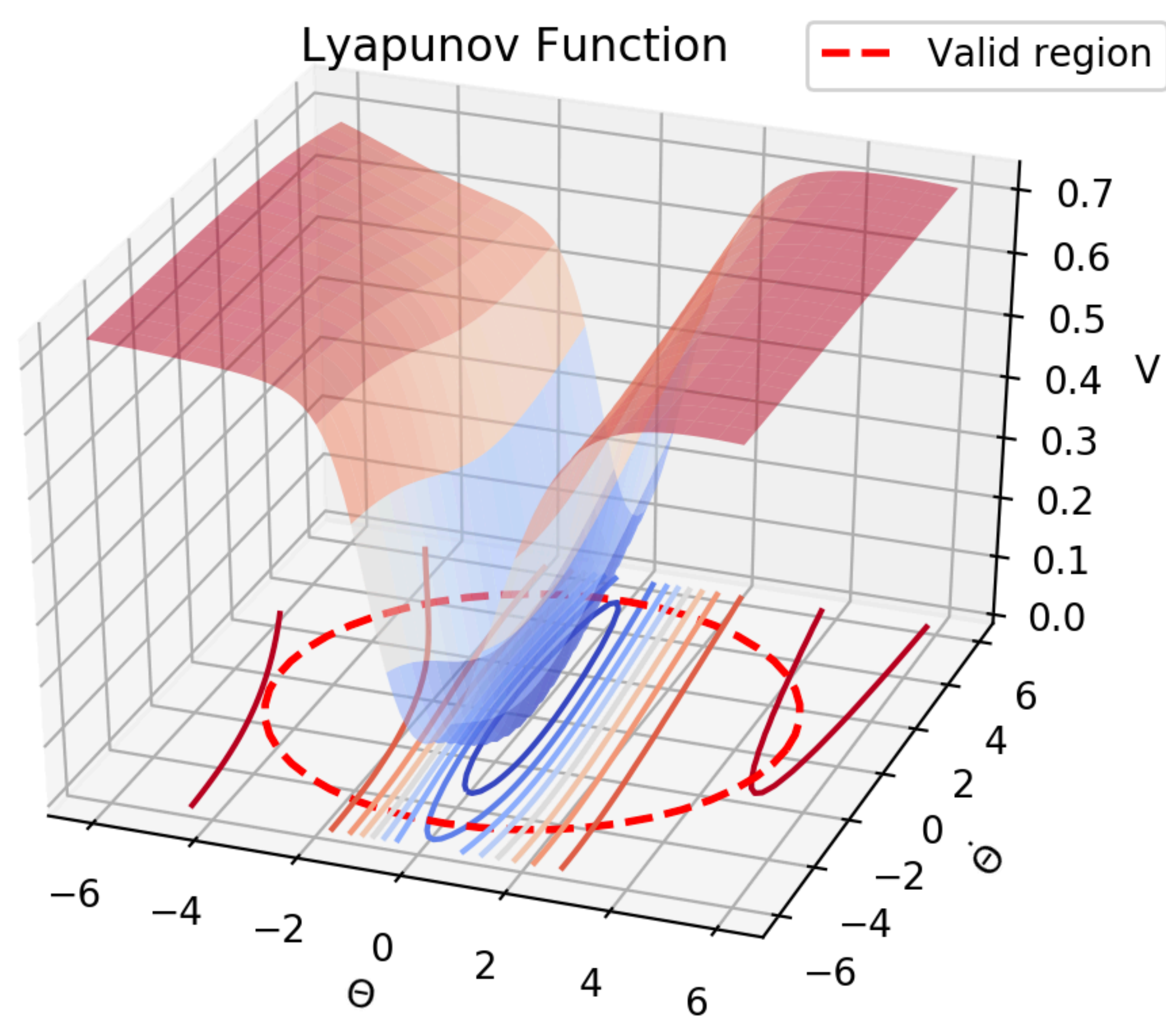
- Instead of asking V to be a polynomial, let it be a neural network.
- Use the verifier/falsifier to enforce the inductive conditions and produce training sets.

[Chang et al. NeurIPS'19]

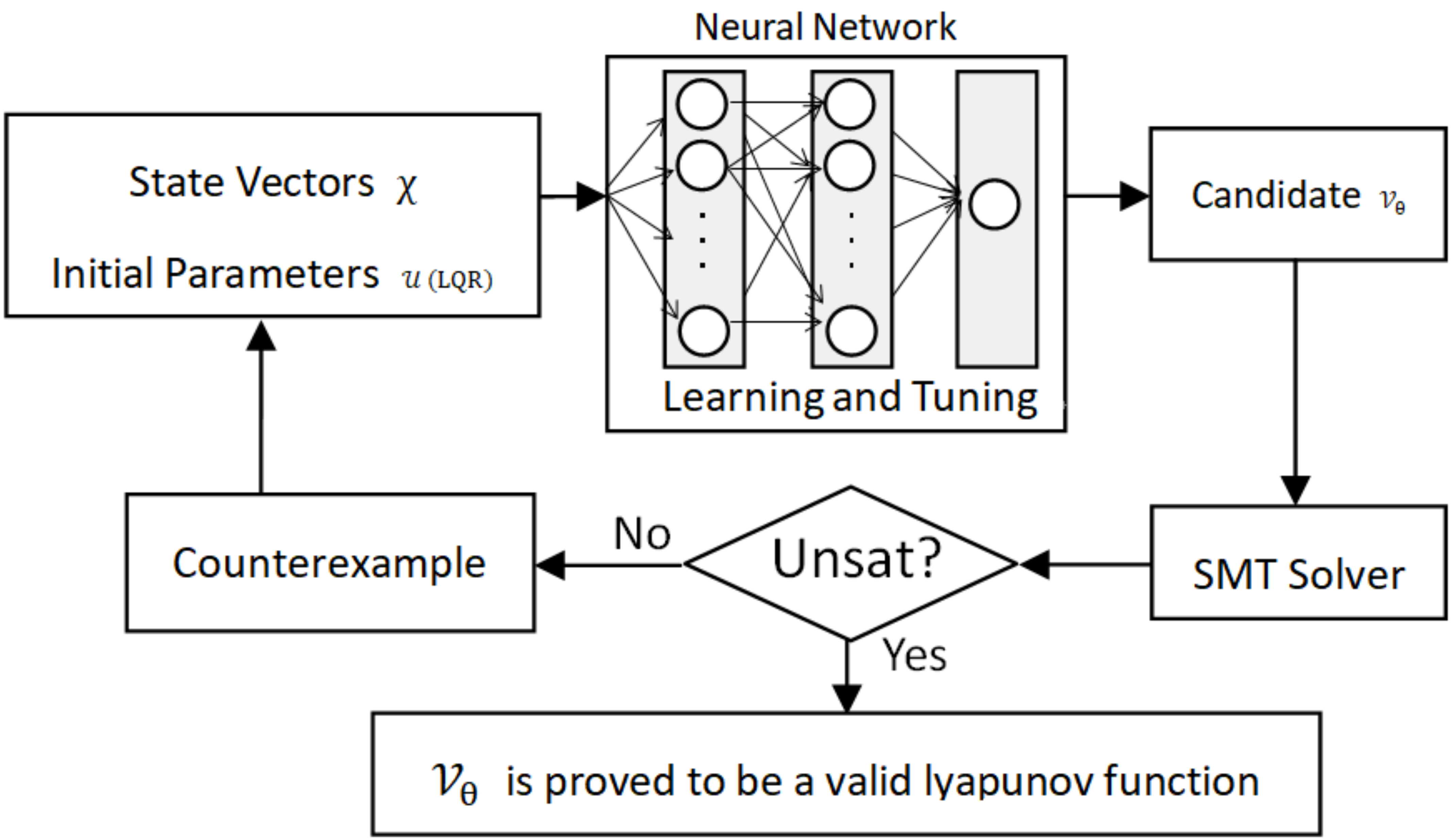
Crazy attempt: use neural networks

Require the neural network V to satisfy the inductive conditions on samples and counterexamples. Just use cheap gradient descent.



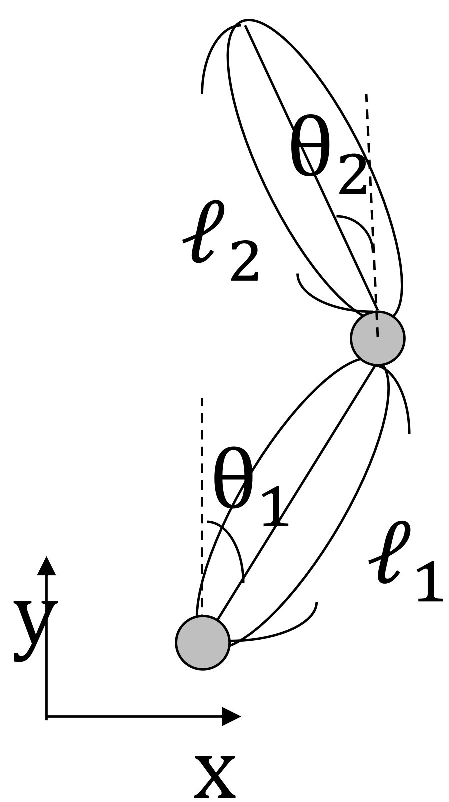


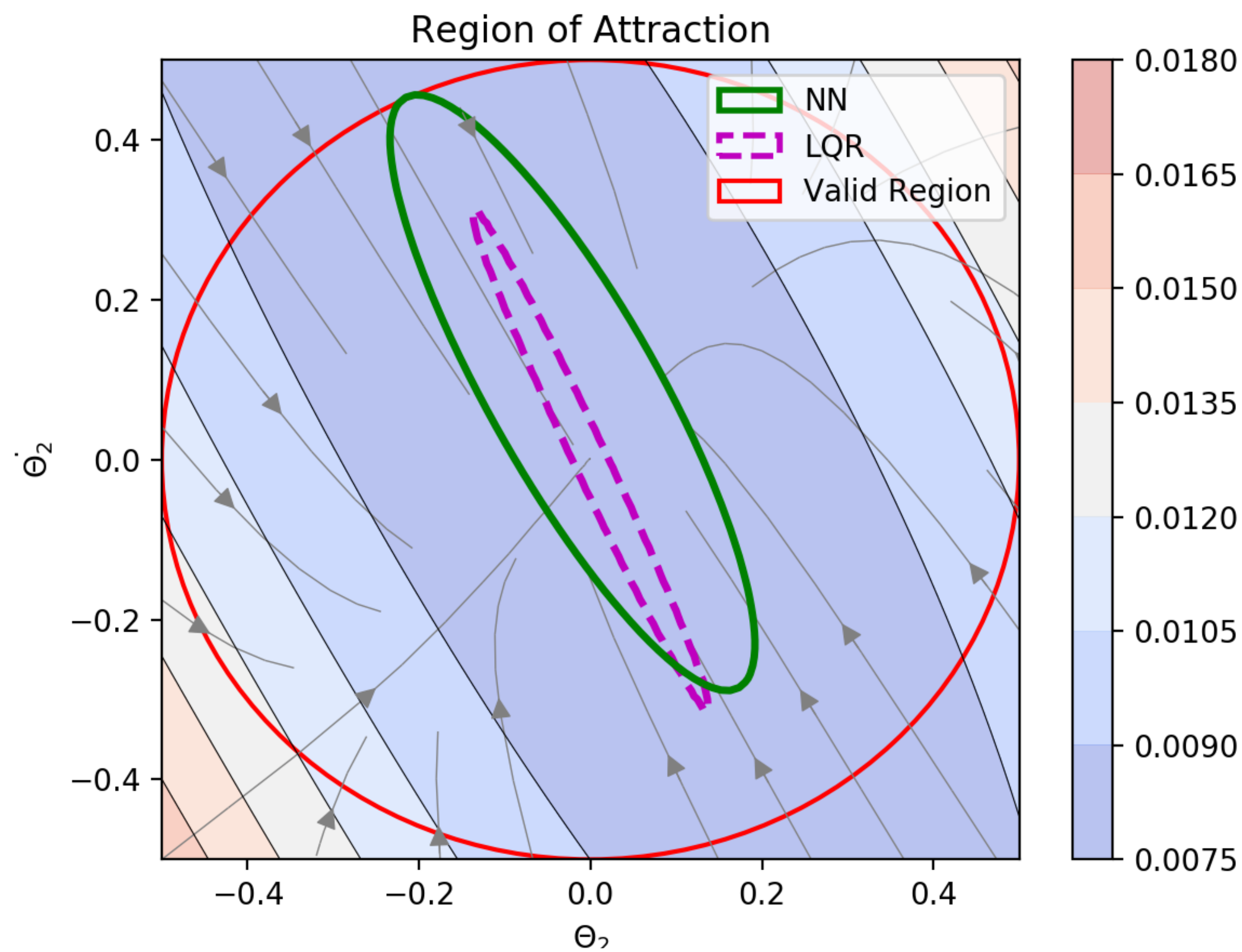
Crazy attempt: use neural networks



Crazy attempt: use neural networks

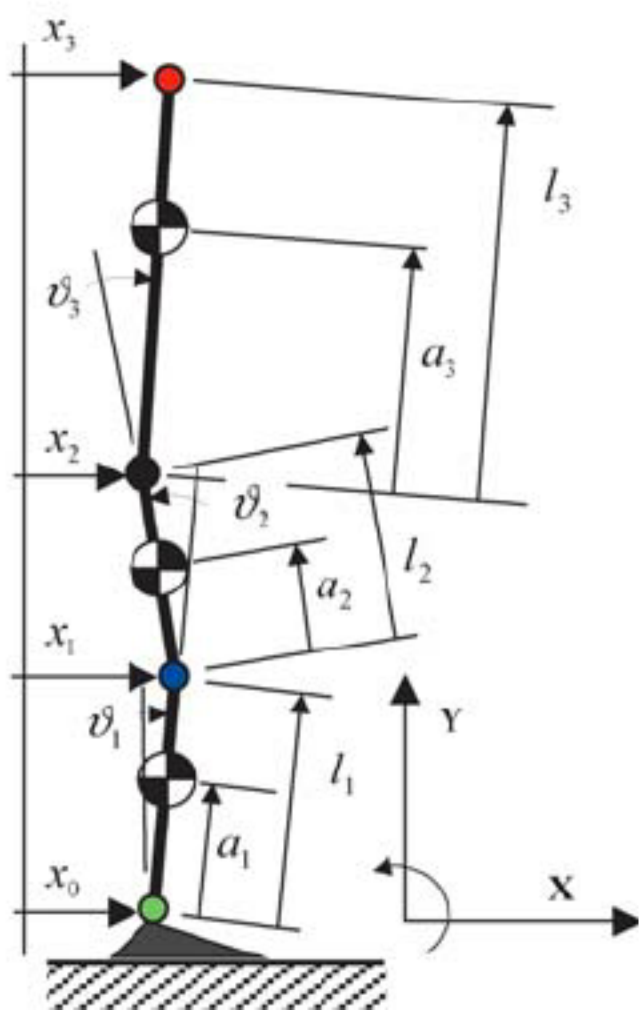
Quite amazingly it worked on many hard examples.

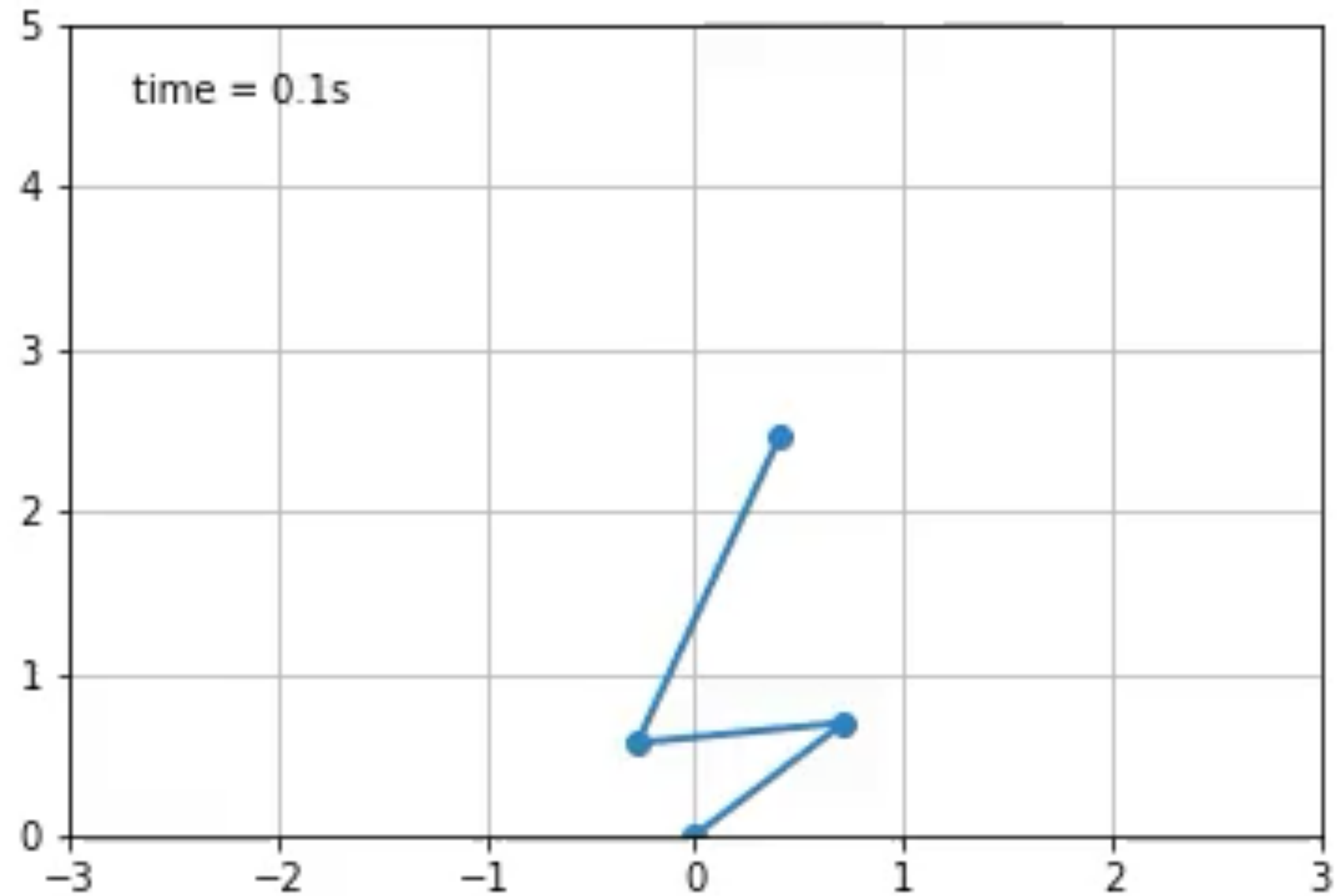




(humanoid balancing)

Crazy attempt: use neural networks

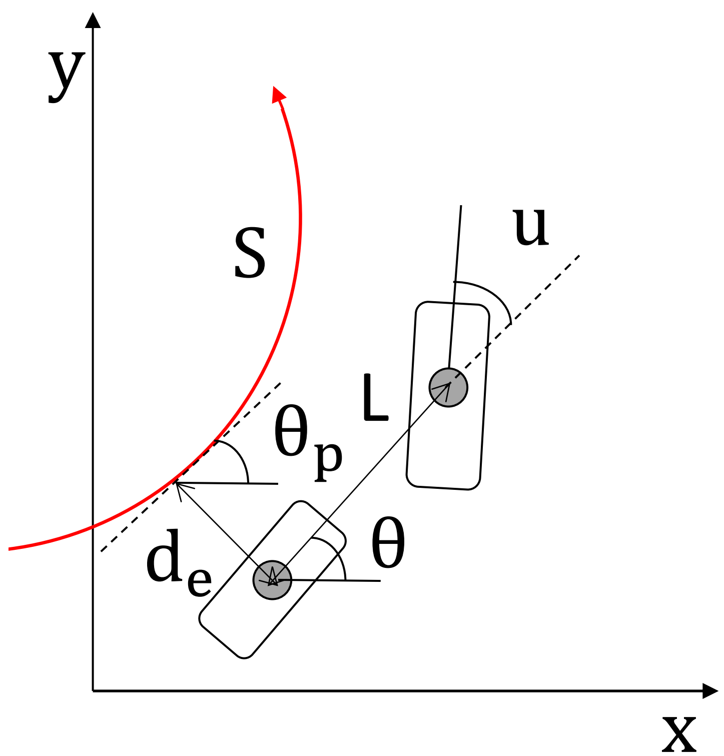


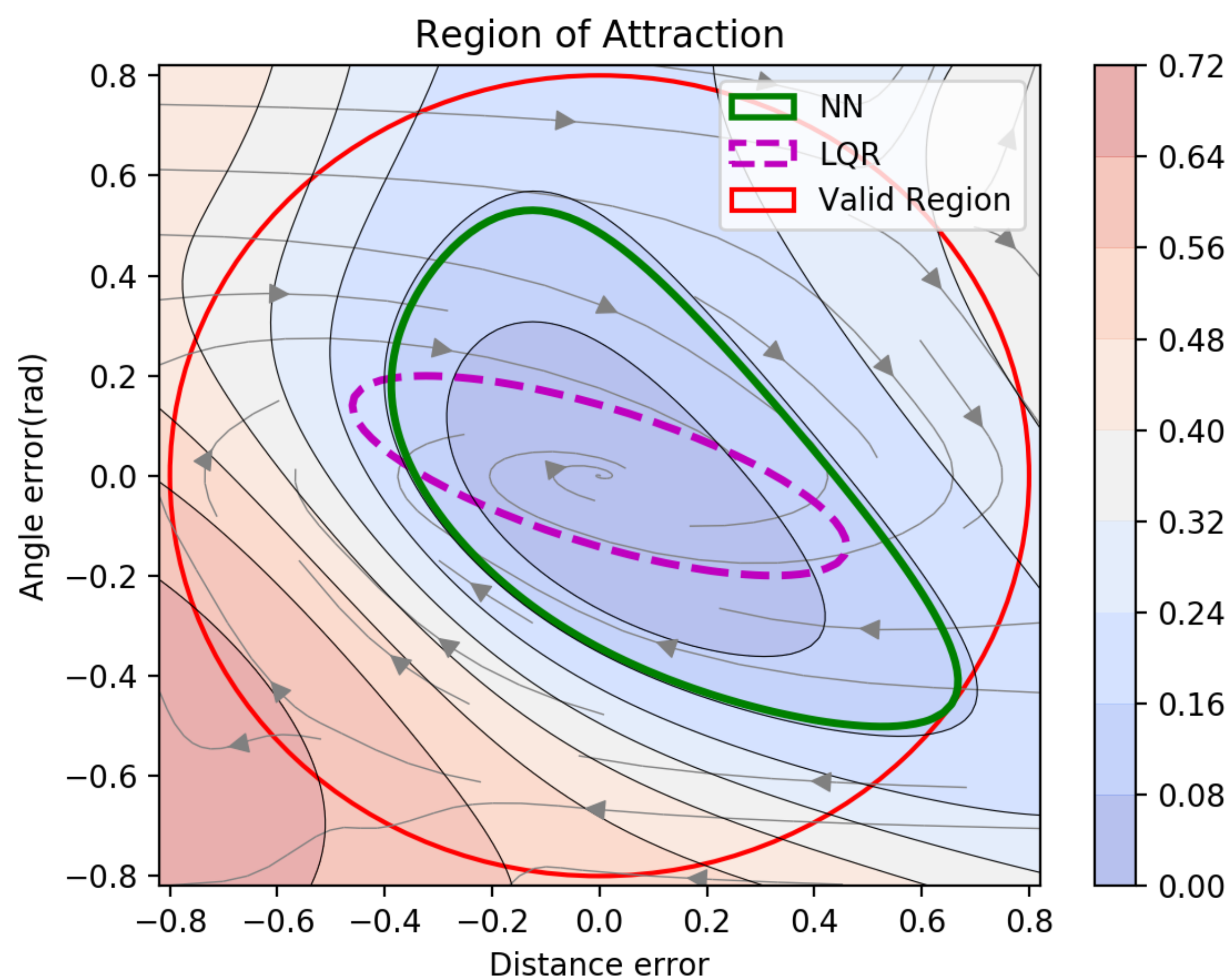


(humanoid balancing)

Crazy attempt: use neural networks

Quite amazingly it worked on many hard examples.

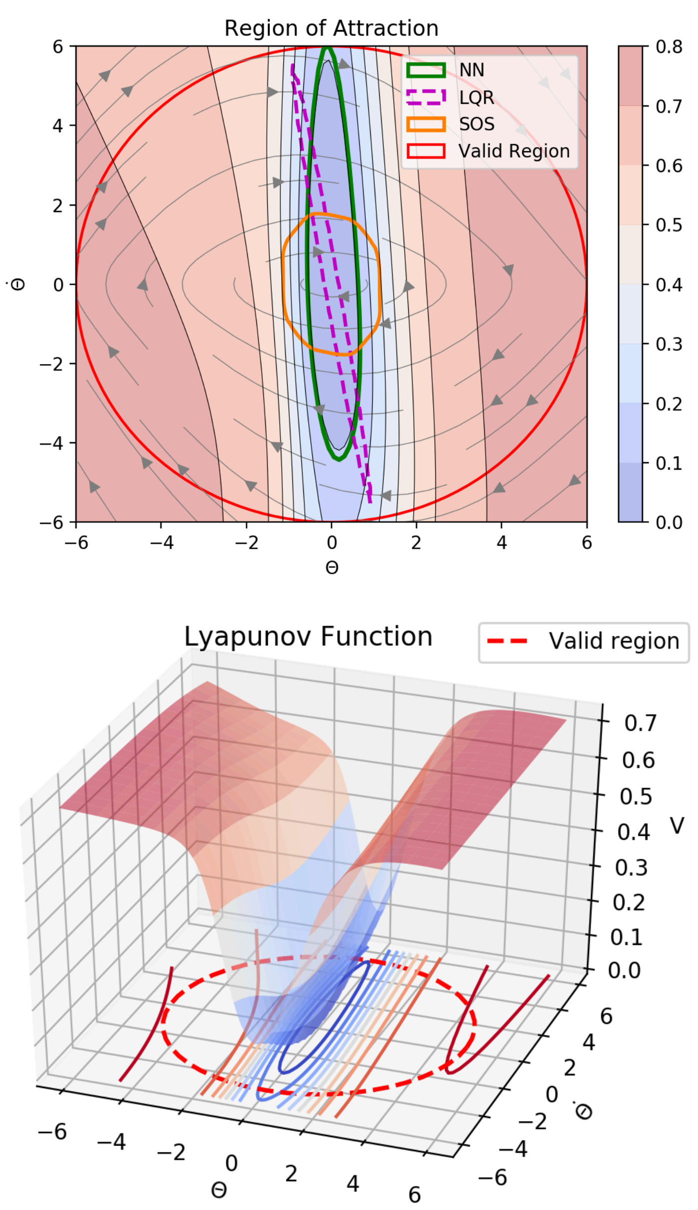


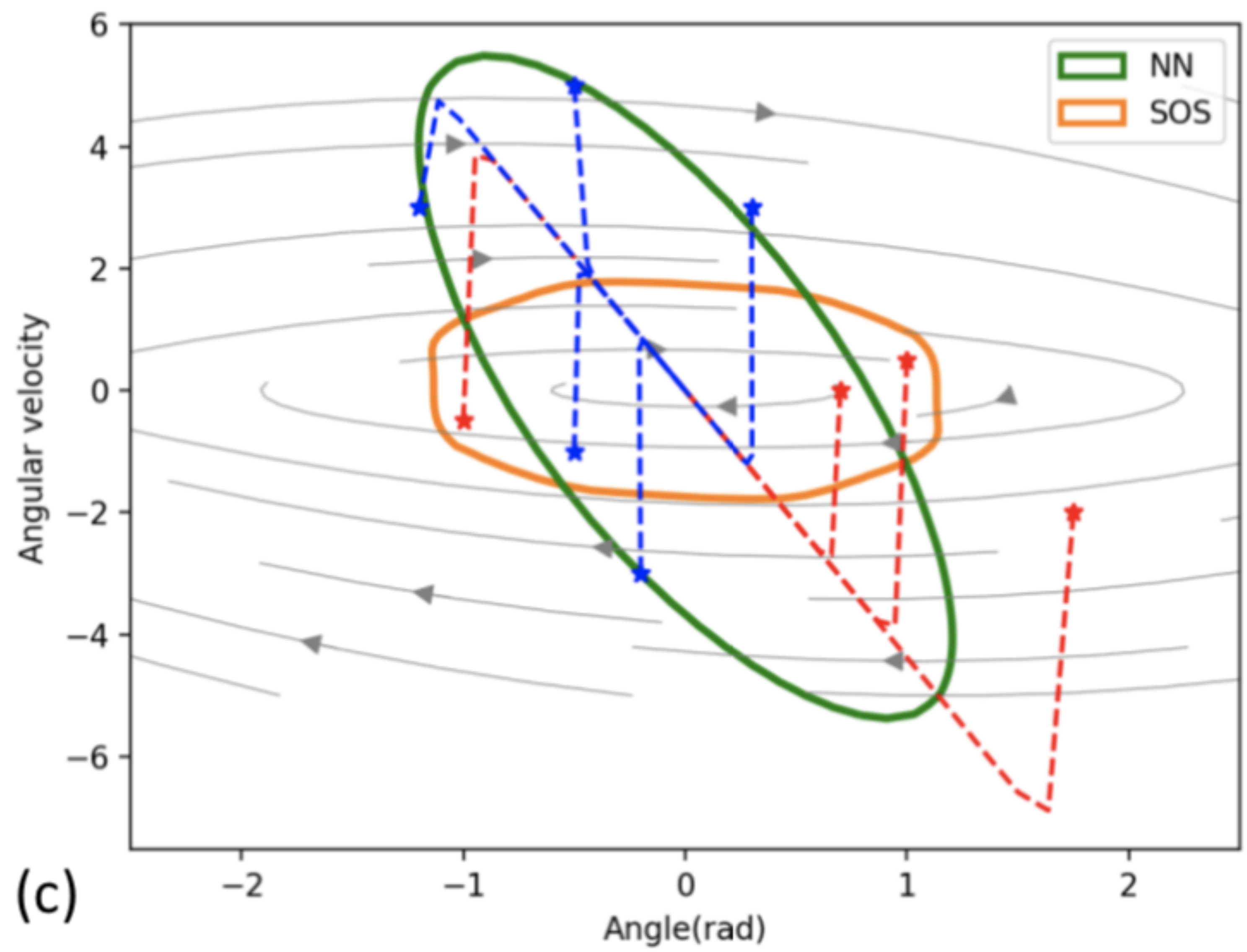


(wheeled vehicle path following)

Crazy attempt: use neural networks

Importantly, it improves previously known RoA.





Conclusion

- For core nonlinear control problems, we can fully automate proofs and designs through reasoning engines and formal tools.
- Improve standard control methods both in performance and reliability guarantees.
- Numerical and probabilistic methods are powerful when their formal basis is established.

Thank you!