ODEs and the Poincaré-Bendixson Theorem in Isabelle/HOL

Fabian Immler and Yong Kiam Tan

Carnegie Mellon University

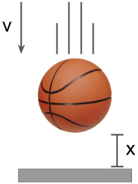
Formal Methods in Mathematics 2020

Recap: Ordinary Differential Equations (ODEs)

Ordinary differential equations (ODEs) provide mathematical models of real world phenomena.

ODE model:

$\dot{x} = v, \dot{v} = -g$

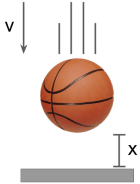


Recap: Ordinary Differential Equations (ODEs)

Ordinary differential equations (ODEs) provide mathematical models of real world phenomena.

ODE model:

$\dot{x} = v, \dot{v} = -g$



ODE solution:

$$x(t) = x_0 + v_0 t - \frac{g}{2} t^2$$

$$v(t) = v_0 - gt$$

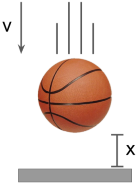
Properties of the ball's falling motion can be deduced from these solutions.

Recap: Ordinary Differential Equations (ODEs)

Ordinary differential equations (ODEs) provide mathematical models of real world phenomena.

ODE model:

$\dot{x} = v, \dot{v} = -g$



ODE solution:

$$x(t) = x_0 + v_0 t - \frac{g}{2} t^2$$

$$v(t) = v_0 - gt$$

Properties of the ball's falling motion can be deduced from these solutions.

A model of glycolysis:

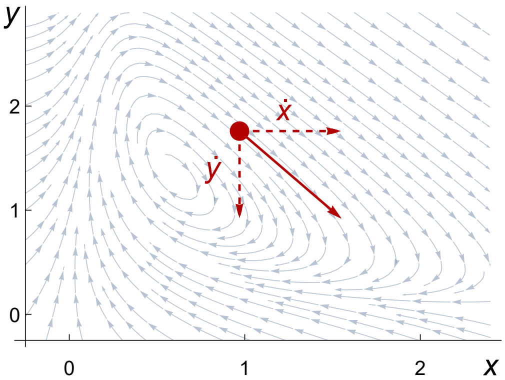
$$\dot{x} = -x + ay + x^2 y$$

$$\dot{y} = b - ay - x^2 y$$

ODE solution: ???

How can we deduce properties *without* knowing the solution?

ODEs and Dynamical Systems

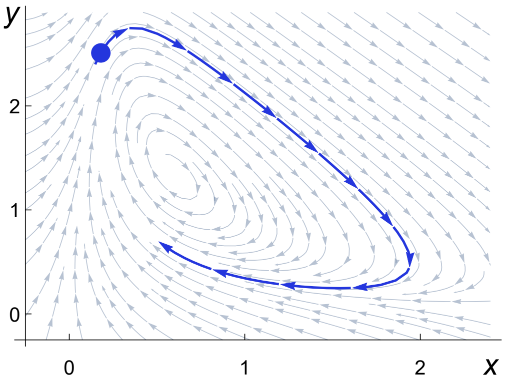


A model of glycolysis:

$$\dot{x} = -x + ay + x^2 y$$
$$\dot{y} = b - ay - x^2 y$$

Approaches:

ODEs and Dynamical Systems



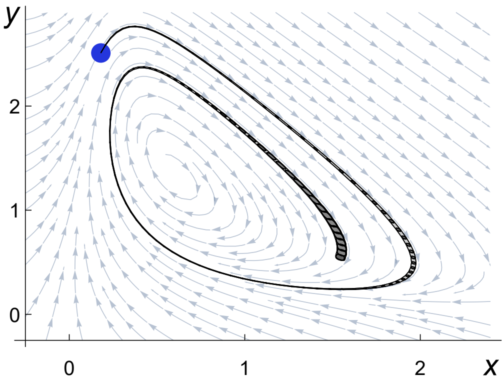
A model of glycolysis:

$$\dot{x} = -x + ay + x^2 y$$
$$\dot{y} = b - ay - x^2 y$$

Approaches:

- simulation – approximate

ODEs and Dynamical Systems



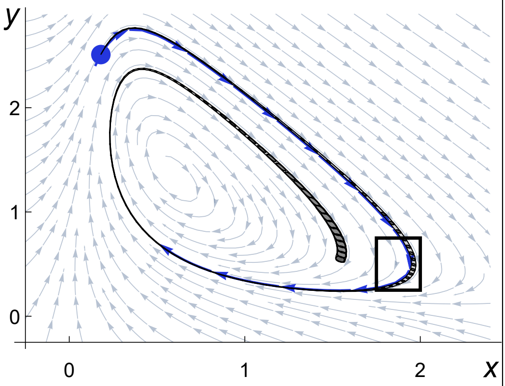
A model of glycolysis:

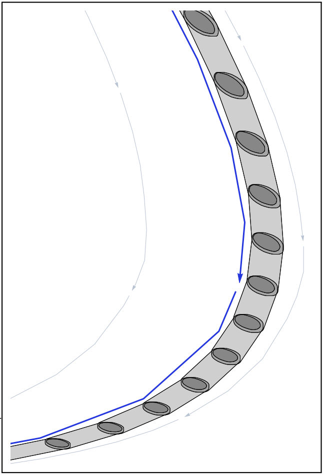
$$\dot{x} = -x + ay + x^2 y$$
$$\dot{y} = b - ay - x^2 y$$

Approaches:

- simulation – approximate
- rigorous numerics – finite time

ODEs and Dynamical Systems





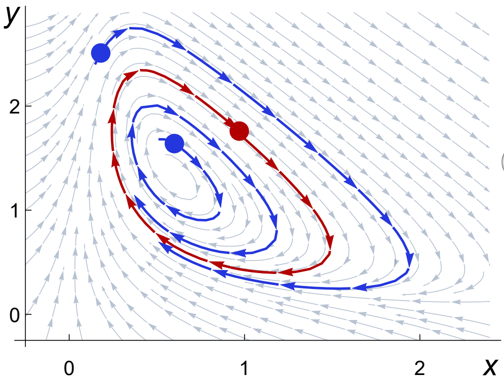
A model of glycolysis:

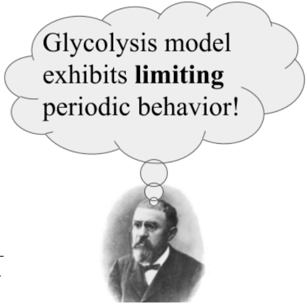
$$\dot{x} = -x + ay + x^2 y$$
$$\dot{y} = b - ay - x^2 y$$

Approaches:

- simulation – approximate
- rigorous numerics – finite time

ODEs and Dynamical Systems





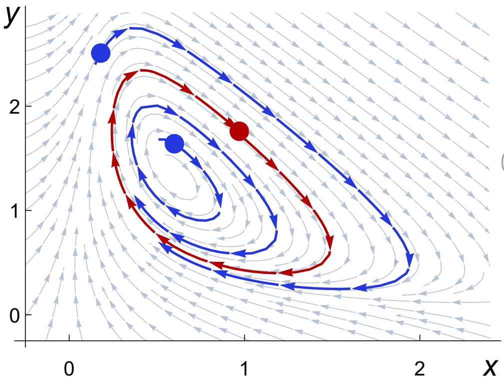
A model of glycolysis:

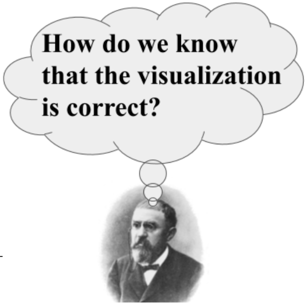
$$\dot{x} = -x + ay + x^2 y$$
$$\dot{y} = b - ay - x^2 y$$

Approaches:

- simulation – approximate
- rigorous numerics – finite time

ODEs and Dynamical Systems





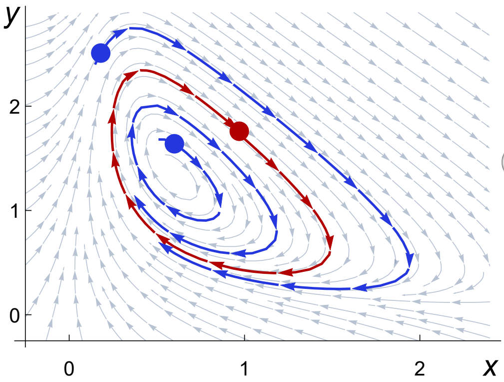
A model of glycolysis:

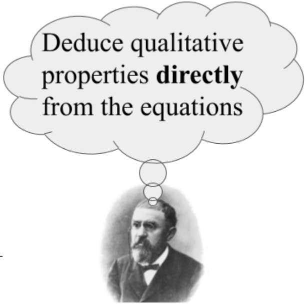
$$\dot{x} = -x + ay + x^2 y$$
$$\dot{y} = b - ay - x^2 y$$

Approaches:

- simulation – approximate
- rigorous numerics – finite time

ODEs and Dynamical Systems





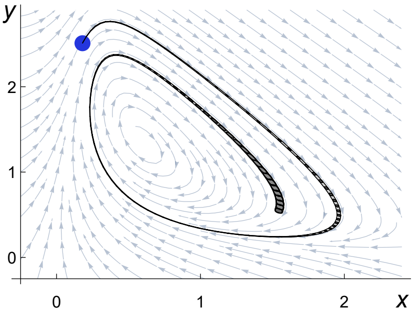
A model of glycolysis:

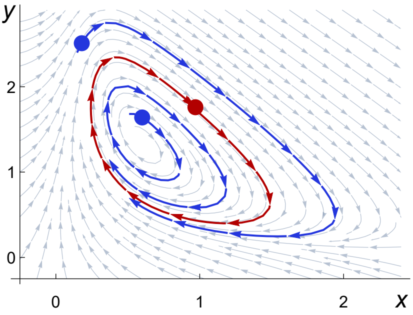
$$\dot{x} = -x + ay + x^2 y$$
$$\dot{y} = b - ay - x^2 y$$

Approaches:

- simulation – approximate
- rigorous numerics – finite time
- deduction directly from equations

Formalization in Isabelle/HOL





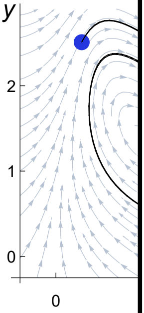
Theorem (Rigorous Numerics)

Solution from initial value is contained in enclosure for time $[0, t_{\mathrm{end}}]$

Theorem (Poincaré-Bendixson)

(Under mild assumptions) trajectories of planar dynamical systems are either periodic or tend towards a periodic trajectory.

Formalization in Isabelle/HOL



Teschl

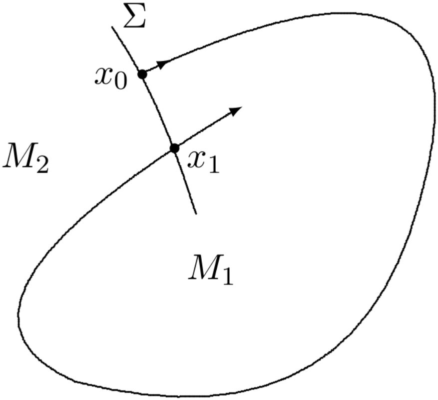


Figure 7.7. Proof of Lemma 7.9

Theorem (… Bendixson)

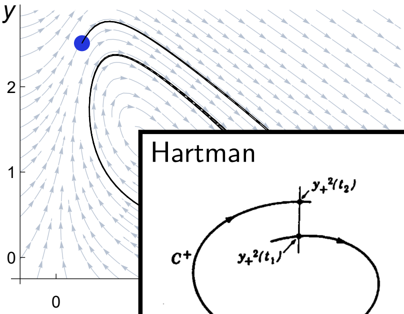
Solution fr … ons)

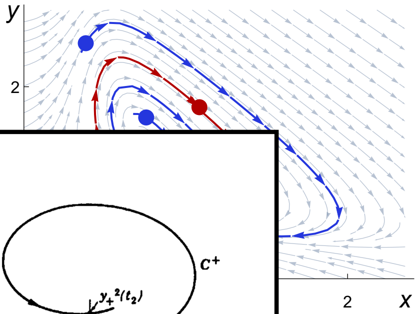
contained i … lynamical

$[0, t_{\text{end}}]$ … *odic or*

c

Formalization in Isabelle/HOL





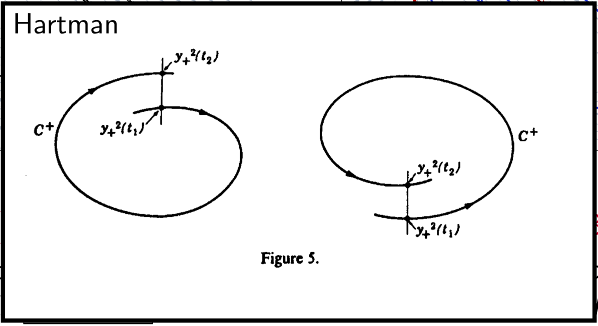


Figure 5.

Theorem (… Bendixson)

Solution fr … *contained i* … $[0, t_{\text{end}}]$

… *ns)* … *ynamical systems are either periodic or tend towards a periodic trajectory.*

Formalization in Isabelle/HOL

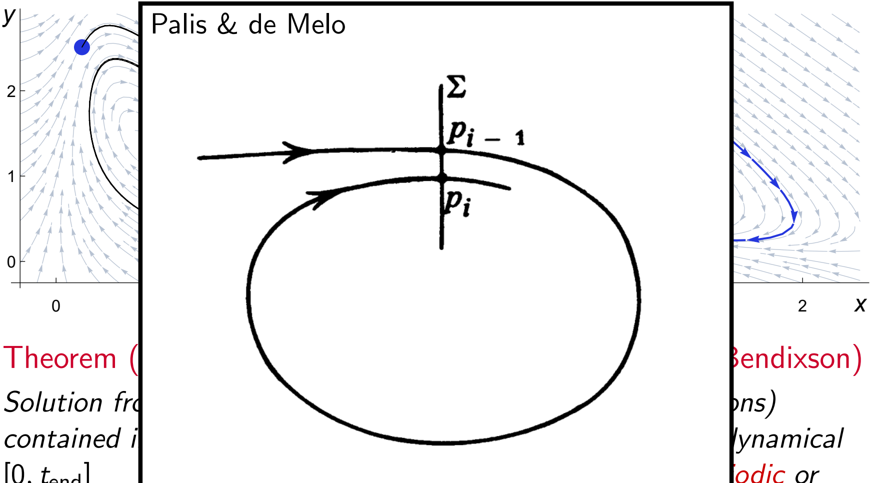
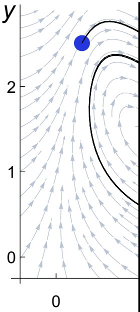


Figure 10

Theorem (… Bendixson)

Solution fr … ons)
contained i … lynamical
$[0, t_{\text{end}}]$ … *iodic or*
… *c*

Formalization in Isabelle/HOL



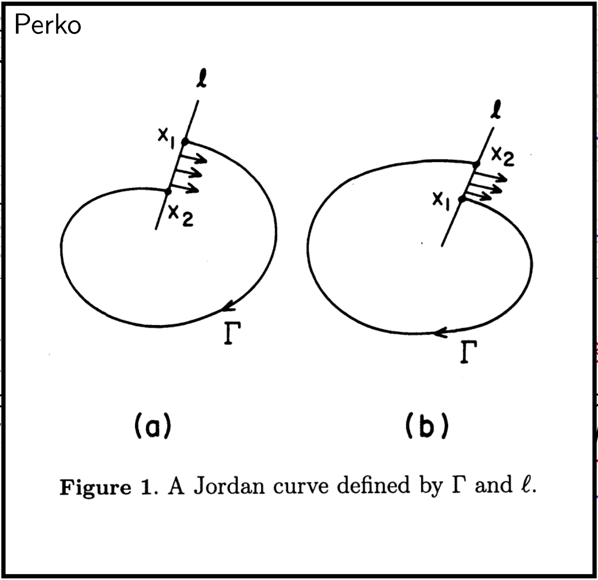
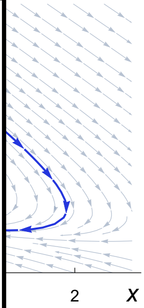


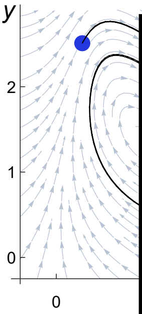
Figure 1. A Jordan curve defined by Γ and ℓ.



Theorem (… Bendixson)

Solution fr … ons)
contained i … dynamical
$[0, t_{end}]$ … *odic or*
… *c*

Formalization in Isabelle/HOL



Wiggins

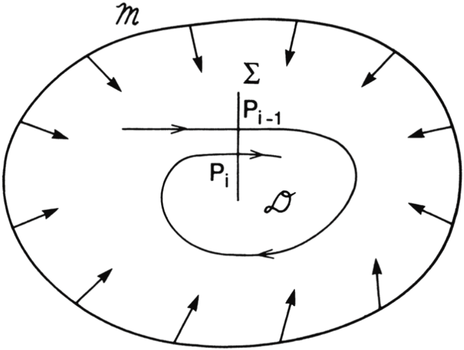
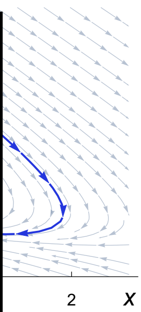


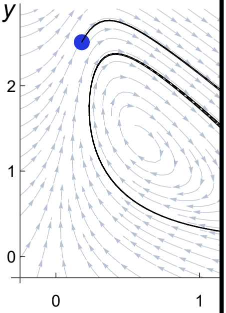
FIGURE 9.0.1.



Theorem (… Bendixson)

Solution fr … ons)
contained i … dynamical
$[0, t_{end}]$ … *odic* *or*
c

Formalization in



Dumortier...

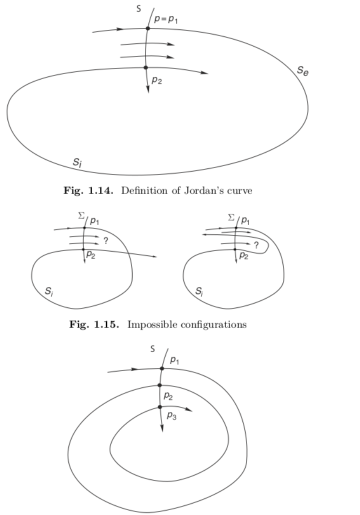
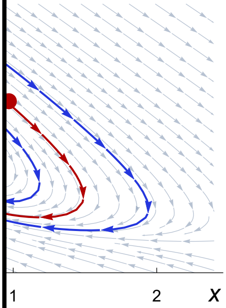


Fig. 1.14. Definition of Jordan's curve

Fig. 1.15. Impossible configurations

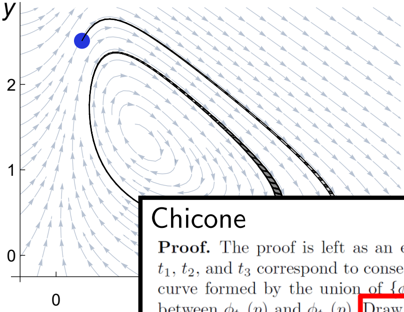
Fig. 1.16. Possible configuration

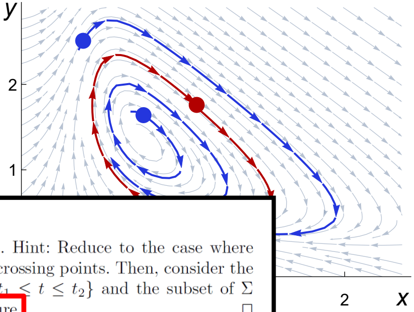


Theorem (Rigoro ... caré-Bendixson)

Solution from init ... umptions)
contained in encl ... *lanar dynamical*
$[0, t_{\text{end}}]$ ... *er periodic or*
periodic

Formalization in Isabelle/HOL





Chicone

Proof. The proof is left as an exercise. Hint: Reduce to the case where t_1, t_2, and t_3 correspond to consecutive crossing points. Then, consider the curve formed by the union of $\{\phi_t(p) : t_1 \leq t \leq t_2\}$ and the subset of Σ between $\phi_{t_1}(p)$ and $\phi_{t_2}(p)$. Draw a picture. □

Theorem (

Solution from initial value is contained in enclosure for time $[0, t_{\text{end}}]$

endixson)

(Under mild assumptions) trajectories of planar dynamical systems are either periodic or tend towards a periodic trajectory.

Outline

ODEs in Isabelle/HOL

The Poincaré-Bendixson Theorem

- Formalization Challenges
- The Monotonicity Lemma
- Example

Conclusion

Outline

ODEs in Isabelle/HOL

Definition

ODE $f : \mathbb{R}^n \to \mathbb{R}^n$

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

for f locally Lipschitz, autonomous/non-autonomous, C^1

Results

ODEs in Isabelle/HOL

Definition

ODE $f : \mathbb{R}^n \to \mathbb{R}^n$

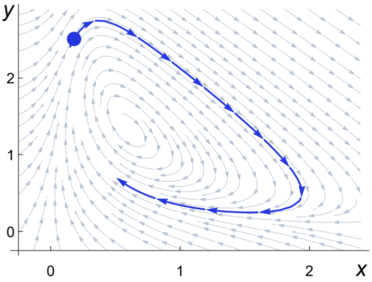
$$\dot{x} = f(x)$$

for f locally Lipschitz, autonomous/non-autonomous, C^1

Results

existence of solution $\phi(x_0, t)$

- $\phi(x_0, 0) = x_0$
- $\frac{\partial}{\partial t}\phi(x_0, t) = f(\phi(x_0, t))$



ODEs in Isabelle/HOL

Definition

ODE $f : \mathbb{R}^n \to \mathbb{R}^n$

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

for f locally Lipschitz, autonomous/non-autonomous, C^1

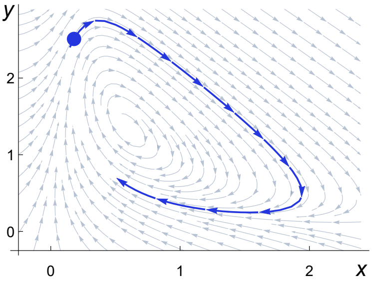
Results

existence of solution $\phi(x_0, t)$

- $\phi(x_0, 0) = x_0$
- $\frac{\partial}{\partial t}\phi(x_0, t) = f(\phi(x_0, t))$

challenge: functional analysis:
ϕ = fixed point of Picard-iteration

$P : \mathcal{C}^{[[t_0;t_1],\mathbb{R}^n]} \to \mathcal{C}^{[[t_0;t_1],\mathbb{R}^n]}$
$P(\psi) = (t \mapsto x_0 + \int_{t_0}^{t} f(\psi(\tau))\mathrm{d}\tau)$



ODEs in Isabelle/HOL

Definition

ODE $f : \mathbb{R}^n \to \mathbb{R}^n$

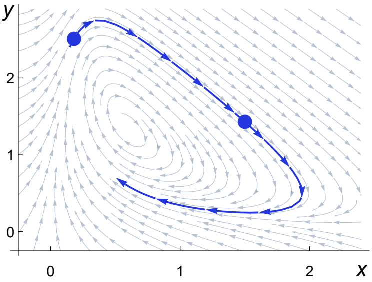
$$\dot{x} = f(x)$$

for f locally Lipschitz, autonomous/non-autonomous, C^1

Results

flow: group action

- $\phi(x_0, 0) = x_0$
- $\phi(\phi(x_0, s), t) = \phi(x_0, s + t)$



ODEs in Isabelle/HOL

Definition

ODE $f : \mathbb{R}^n \to \mathbb{R}^n$

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

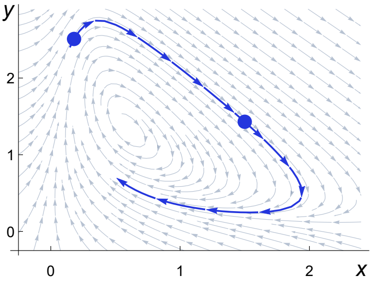
for f locally Lipschitz, autonomous/non-autonomous, C^1

Results

flow: group action

- $\phi(x_0, 0) = x_0$
- $\phi(\phi(x_0, s), t) = \phi(x_0, s + t)$

nice:
algebraic reasoning
tedious:
$t, s, t + s \in \text{existence\_ivl}(x_0)$



ODEs in Isabelle/HOL

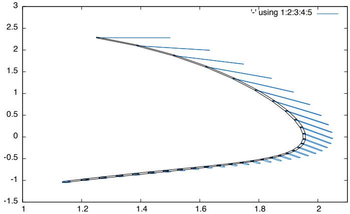
Definition

ODE $f : \mathbb{R}^n \to \mathbb{R}^n$

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

for f locally Lipschitz, autonomous/non-autonomous, C^1

Results



flow: differentiability

- $\frac{\partial}{\partial x_0}\phi(x_0, t) = A(t)$
- variational equation:
  $\dot{A} = \mathrm{D}f|_{\phi(x_0,t)} \cdot A$, $A : \mathbb{R}^{n \times n}$

ODEs in Isabelle/HOL

Definition

ODE $f : \mathbb{R}^n \to \mathbb{R}^n$

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

for f locally Lipschitz, autonomous/non-autonomous, C^1

Results

flow: differentiability

- $\frac{\partial}{\partial x_0}\phi(x_0, t) = A(t)$
- variational equation:
  $\dot{A} = \mathrm{D}f|_{\phi(x_0,t)} \cdot A$, $A : \mathbb{R}^{n\times n}$

challenge: module system

ODE $f : \mathbb{R}^n \to \mathbb{R}^n$
$\phi : \mathbb{R} \to \mathbb{R}^n$

Var.ODE $(\lambda A.\ \mathrm{D}f|_{\phi(x_0,t)} \cdot A)$: $\mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$
Var.$\phi : \mathbb{R} \to \mathbb{R}^{n\times n}$

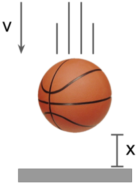
Lemma $\mathrm{D}\phi_t|_{x_0} = \mathrm{Var}.\phi(t)$

Hybrid Systems in Isabelle/HOL

hybrid = continuous + discrete

ODE model:

$\dot{x} = v, \dot{v} = -g$



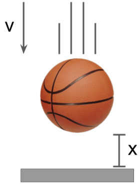
Hybrid Systems in Isabelle/HOL

hybrid = continuous + discrete

ODE model:

$\dot{x} = v, \dot{v} = -g$

domain: $x >= 0$



Hybrid Systems in Isabelle/HOL

hybrid = continuous + discrete

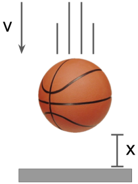
ODE model:

$\dot{x} = v, \dot{v} = -g$

domain: $x >= 0$

discrete control:

$v \leftarrow -v$ when $x = 0$



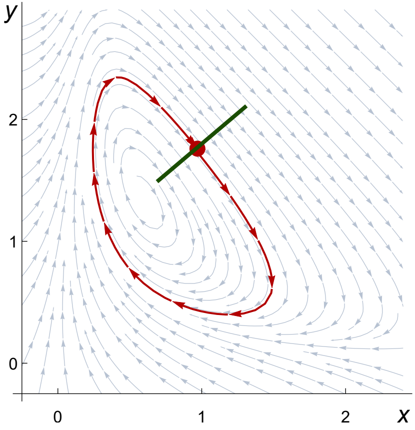
Poincaré map

The main mathematical tool to talk about discrete switches at a **Poincaré section** (smooth surface)

Usual Definition

at periodic orbit

- return time of **periodic point** = period



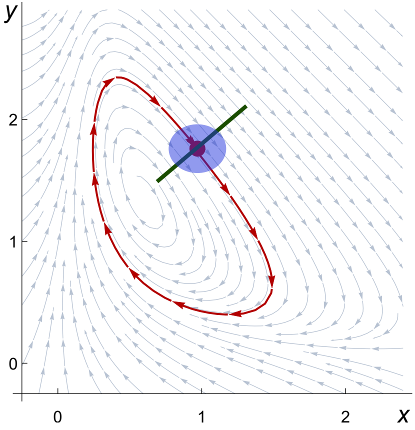
Poincaré map

The main mathematical tool to talk about discrete switches at a **Poincaré section** (smooth surface)

Usual Definition

at periodic orbit

- return time of **periodic point** = period
- solve return time in neighborhood with implicit function theorem



Poincaré map

The main mathematical tool to talk about discrete switches at a **Poincaré section** (smooth surface)

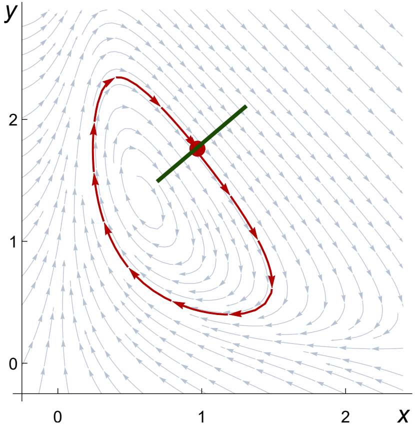
Usual Definition

at periodic orbit

- return time of **periodic point** = period
- solve return time in neighborhood with implicit function theorem

Formalized Definition

- first return time τ
  $\phi(\mathbf{x_0}, \tau(\mathbf{x_0})) \in \mathbf{S}$
  $\forall t < \tau(\mathbf{x_0}).\ \phi(\mathbf{x_0}, t) \notin \mathbf{S}$



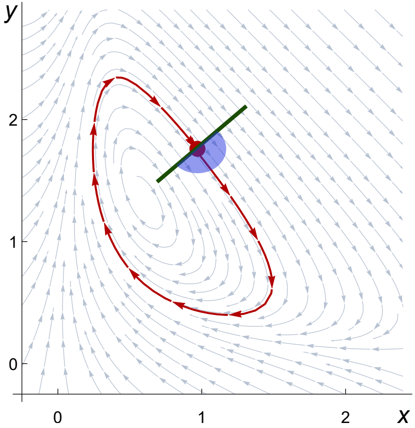
Poincaré map

The main mathematical tool to talk about discrete switches at a **Poincaré section** (smooth surface)

Usual Definition

at periodic orbit

- return time of **periodic point** = period
- solve return time in neighborhood with implicit function theorem



Formalized Definition

- first return time τ
  $\phi(\mathbf{x_0}, \tau(\mathbf{x_0})) \in \mathbf{S}$
  $\forall t < \tau(\mathbf{x_0}).\ \phi(\mathbf{x_0}, t) \notin \mathbf{S}$
- $\mathcal{C}^1$ on and outside of $\mathbf{S}$ (continuous above/at)

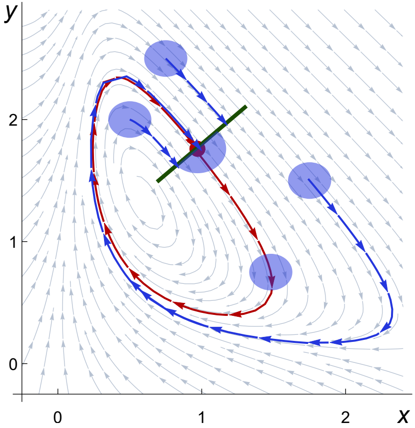
Poincaré map

The main mathematical tool to talk about discrete switches at a **Poincaré section** (smooth surface)

Usual Definition

at periodic orbit

- return time of **periodic point** = period
- solve return time in neighborhood with implicit function theorem

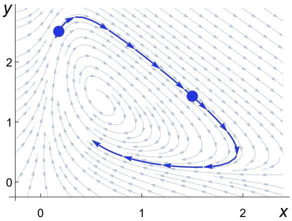


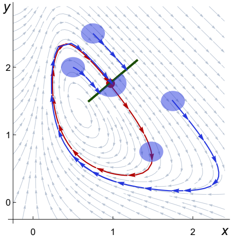
Formalized Definition

- first return time τ
  $\phi(\mathbf{x_0}, \tau(\mathbf{x_0})) \in \mathbf{S}$
  $\forall t < \tau(\mathbf{x_0}).\ \phi(\mathbf{x_0}, t) \notin \mathbf{S}$
- $\mathcal{C}^1$ on and outside of $\mathbf{S}$ (continuous above/at)

Summary of Abstract Results

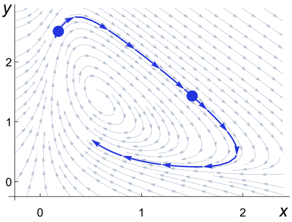
- unique solution, flow ϕ, Poincaré map

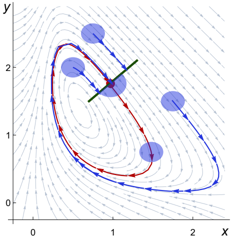




Summary of Abstract Results

- unique solution, flow ϕ, Poincaré map

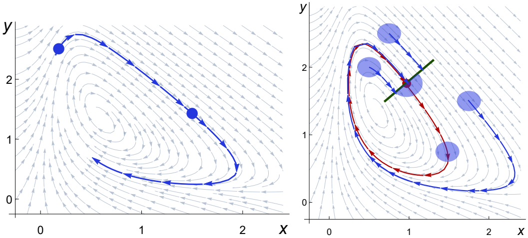




- applications:
  - Formally Verified Differential Dynamic Logic [Bohrer et. al.]
  - Verifying Hybrid Systems with Modal Kleene Algebra [Munive, Struth]
  - Towards Verification of Cyber-Physical Systems with UTP and Isabelle/HOL [Foster, Woodcock]

Summary of Abstract Results

- unique solution, flow ϕ, Poincaré map



- applications:
  - Formally Verified Differential Dynamic Logic [Bohrer et. al.]
  - Verifying Hybrid Systems with Modal Kleene Algebra [Munive, Struth]
  - Towards Verification of Cyber-Physical Systems with UTP and Isabelle/HOL [Foster, Woodcock]

Problem

- simulation provides important insights
- not (directly) amenable to formalization

Rigorous Numerical Methods

formalize simulations?

- in principle, could verify $\|\text{simulation}(x_0, t) - \phi(x_0, t)\| \leq \mathcal{O}(e^t)$

Rigorous Numerical Methods

formalize simulations?

- in principle, could verify $\|\text{simulation}(x_0, t) - \phi(x_0, t)\| \leq \mathcal{O}(e^t)$
- overly pessimistic!

Rigorous Numerical Methods

formalize simulations?

- in principle, could verify $\|\text{simulation}(x_0, t) - \phi(x_0, t)\| \leq \mathcal{O}(e^t)$
- overly pessimistic!

propagate error bounds on the fly

Rigorous Numerical Methods

formalize simulations?

- in principle, could verify $\|\text{simulation}(x_0, t) - \phi(x_0, t)\| \leq \mathcal{O}(e^t)$
- overly pessimistic!

propagate error bounds on the fly

- stable systems damp errors

Rigorous Numerical Methods

formalize simulations?

- in principle, could verify $\|\text{simulation}(x_0, t) - \phi(x_0, t)\| \leq \mathcal{O}(e^t)$
- overly pessimistic!

propagate error bounds on the fly

- stable systems damp errors
- like simulation, but rigorous

Rigorous Numerical Methods

formalize simulations?

- in principle, could verify $\|\text{simulation}(x_0, t) - \phi(x_0, t)\| \leq \mathcal{O}(e^t)$
- overly pessimistic!

propagate error bounds on the fly

- stable systems damp errors
- like simulation, but rigorous

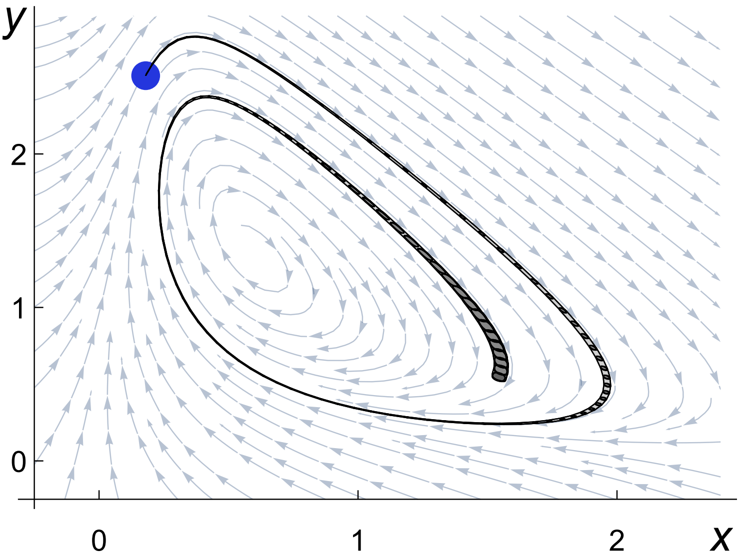
Rigor

for

$e^t)$

pr

A Rigorous (and Verified) Simulation



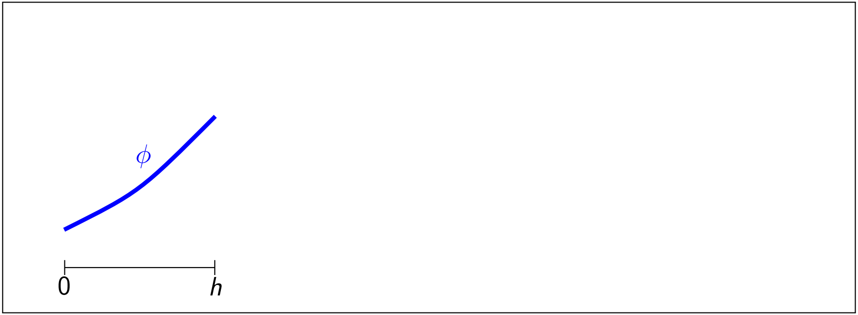
A Rigorous (and Verified) Simulation

Examples.thy (%HOME%\work\IsabelleODE\)

```
schematic_goal g_fas:
"[(- (X!0) + 8 / 100 * (X!1) + (X!0)^2 * (X!1)),( 6 / 10 - 8 / 100 * (X!1) - (X!0)^2 * (X!1))] =
  interpret_floatariths ?fas X"
  by (reify_floatariths)

concrete_definition g_fas uses g_fas

interpretation g_ode: ode_interpretation true_form UNIV g_fas
  "(λ(x, y). (gx x y, gy x y)::real*real)"
  "d::2" for d
  by unfold_locales (auto simp: g_fas_def less_Suc_eq_0_disj nth_Basis_list_prod Basis_list_real_def
      gx_def gy_def eval_nat_numeral
      mk_ode_ops_def eucl_of_list_prod power2_eq_square intro!: isFDERIV_I)

lemma ptout: "t ∈ {13 .. 13} ⟶ (x, y) ∈ {(0.18, 2.51) .. (0.18, 2.51)} ⟶
   t ∈ g.existence_ivl0 (x, y) ∧ g.flow0 (x, y) t ∈ {(1.51, 0.51) .. (1.57, 0.58)}"
  by (tactic ‹ode_bnds_tac @{thms g_fas_def} 30 40 20 12 [(0, 1, "0x000000")] "-" @{context} 1›)
```

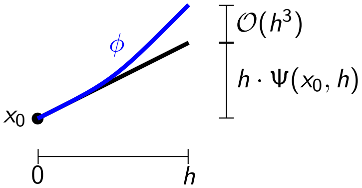
```
1.522387 5.40937e-1 0x000000
1.523253 5.395244e-1 0x000000
1.525231 5.367902e-1 0x000000
1.52759 5.339942e-1 0x000000
1.531406 5.301116e-1 0x000000
1.5329 5.288828e-1 0x000000
1.536989 5.260615e-1 0x000000
1.545201 5.219829e-1 0x000000
1.549437 5.205454e-1 0x000000
1.552498 5.198054e-1 0x000000
1.556858 5.198054e-1 0x000000
1.560703 5.21248e-1 0x000000
1.561017 5.214269e-1 0x000000
1.562223 5.22797e-1 0x000000
1.563608 5.253099e-1 0x000000
1.565401 5.294668e-1 0x000000
1.5655 5.29763e-1 0x000000
1.565983 5.313015e-1 0x000000
1.566295 5.324315e-1 0x000000
1.56657 5.334455e-1 0x000000
1.566856 5.346362e-1 0x000000
1.567238 5.367166e-1 0x000000
1.567547 5.415899e-1 0x000000
1.567547 5.426622e-1 0x000000

# (1.51947 5.198049e-1) .. (1.567547 5.746059e-1); devs: 26; tdev: (2.40384e-2 2.740012e-2)
1.569999 5.799999e-1 0x000000
1.51 5.799999e-1 0x000000
1.51 5.1e-1 0x000000
1.569999 5.1e-1 0x000000
1.569999 5.799999e-1 0x000000

# (1.509999 5.099997e-1) .. (1.57 5.800004e-1); devs: 2; tdev: (3e-2 3.500002e-2)
```

A Verified ODE Solver



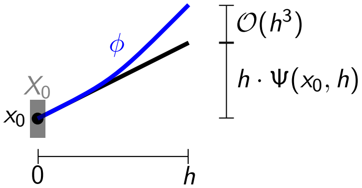
A Verified ODE Solver

Guaranteed **R**unge-**K**utta methods [Bouissou et. al.]:



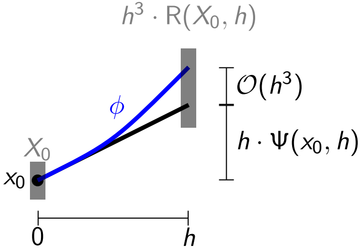
A Verified ODE Solver

Guaranteed **R**unge-**K**utta methods [Bouissou et. al.]:



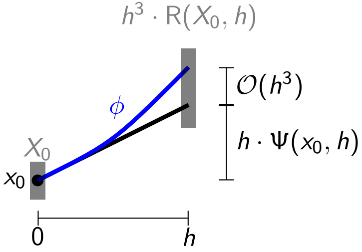
A Verified ODE Solver

Guaranteed **R**unge-**K**utta methods [Bouissou et. al.]:



A Verified ODE Solver

Guaranteed **R**unge-**K**utta methods [Bouissou et. al.]:

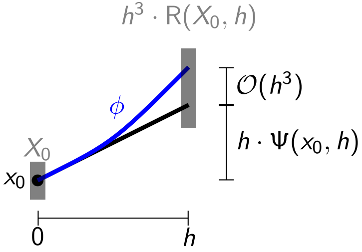


Theorem (Rigorous Euler method)

$$\forall x_0 \in X_0.\ \phi(x_0, h) \in X_0 + h \cdot f(X_0, h) + h^2 \cdot R(X_0, h)$$

A Verified ODE Solver

Guaranteed **R**unge-**K**utta methods [Bouissou et. al.]:



Theorem (Rigorous Euler method)

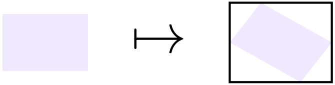
$$\forall x_0 \in X_0.\ \phi(x_0, h) \in X_0 + h \cdot f(X_0, h) + h^2 \cdot R(X_0, h)$$

Algorithm

Evaluate in interval/affine arithmetic

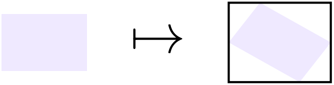
Affine Arithmetic

- wrapping effect of intervals:

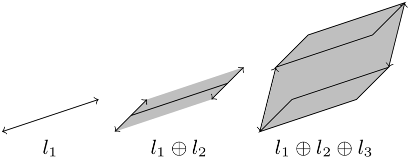


Affine Arithmetic

- wrapping effect of intervals:



- therefore zonotopes: $\{\ell_0 + \sum_i \varepsilon_i \cdot \ell_i \mid \varepsilon_i \in [-1;1]\}$



Verification

Techniques

- Refinement

Verification

Techniques

- Refinement
- Refinement

Verification

Techniques

- Refinement
- Refinement
- Refinement

Verification

Techniques

- Refinement
- Refinement
- Refinement

Example

Verification

Techniques

- Refinement
- Refinement
- Refinement

Example

- $\phi(X_0, h) \subseteq R$

Verification

Techniques

- Refinement
- Refinement
- Refinement

Example

- $\phi(X_0, h) \subseteq R$
- R defined as Runge-Kutta remainder

Verification

Techniques

- Refinement
- Refinement
- Refinement

Example

- $\phi(X_0, h) \subseteq R$
- R defined as Runge-Kutta remainder
- Runge-Kutta implemented in affine arithmetic
  (on real numbers)

Verification

Techniques

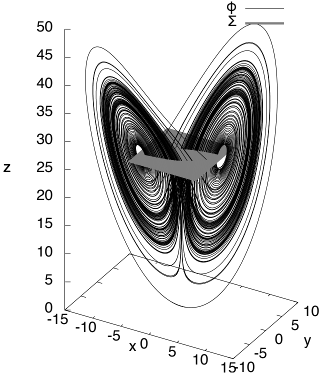
- Refinement
- Refinement
- Refinement

Example

- $\phi(X_0, h) \subseteq R$
- R defined as Runge-Kutta remainder
- Runge-Kutta implemented in affine arithmetic (on real numbers)
- Runge-Kutta implemented in affine arithmetic (on floating point numbers)

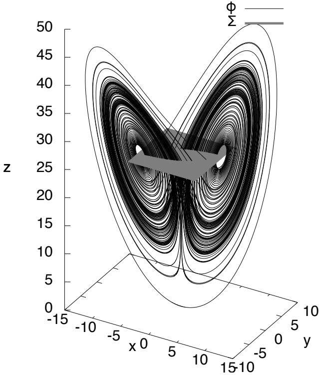
Smale's 14th Problem

- 
  Lorenz (1963): is this chaos?



Smale's 14th Problem

- 
  Lorenz (1963): is this chaos?
- 
  Tucker (2002): Yes:

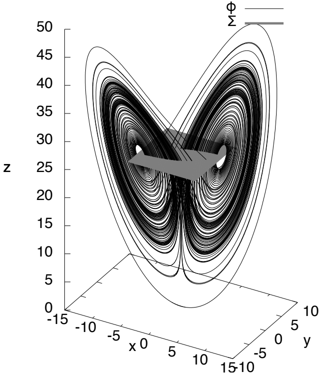


Smale's 14th Problem

- 

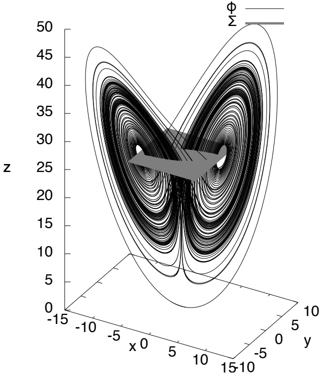
  Lorenz (1963): is this chaos?
- 

  Tucker (2002): Yes:
  - 1 paragraph combining standard results



Smale's 14th Problem

- 
  Lorenz (1963): is this chaos?
- 
  Tucker (2002): Yes:
  - 1 paragraph combining standard results
  - normal form theory (25 pages)

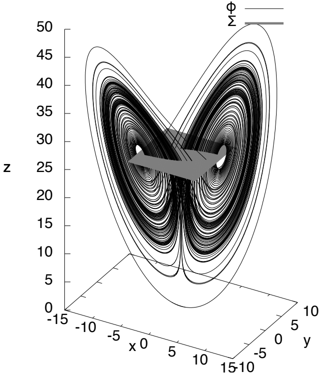


Smale's 14th Problem

- 

  Lorenz (1963): is this chaos?
- 

  Tucker (2002): Yes:
  - 1 paragraph combining standard results
  - normal form theory (25 pages)
  - C++ Program (24 pages, 3800+8800 lines of numerical code)



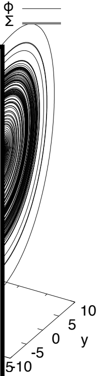
Smale's 14th Problem

Lor

Tuc

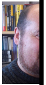
code)

+1255	+727	8	2	1	+0.0000000000000000e+00	+1.0000000000949800e+01	+1.45716683960320864e-01	+7.23395908441154978e-01	+1241	+725	0	+1247	+725	0
+1243	+725	8	2	1	+9.04432164386413384e-01	+1.08990638557001169e+01	+1.12826302448460050e-01	+7.32922798665620823e-01	+1229	+723	0	+1235	+725	0
+1241	+725	8	2	1	+8.65974433594971504e-01	+1.09243804337963759e+01	+1.14232002982602596e-01	+7.35890694255760480e-01	+1225	+723	0	+1231	+723	0
+1245	+725	8	2	1	+9.33652813609950560e-01	+1.08564425045962221e+01	+1.12032998352967234e-01	+7.30345897238957265e-01	+1231	+723	0	+1237	+725	0
+1247	+725	8	2	1	+9.17163872676489689e-01	+1.08633491856049176e+01	+1.12032998352967234e-01	+7.26924114320113057e-01	+1233	+723	0	+1239	+725	0
+1229	+723	8	2	1	+1.34247894561929781e+00	+1.05313232162328863e+01	+1.29418029950727875e-01	+7.54571855908492628e-01	+1213	+721	0	+1219	+723	0
+1231	+723	8	2	1	+1.19647410213745875e+00	+1.06791840420694140e+01	+1.26647798223389868e-01	+7.50373126840981897e-01	+1215	+721	0	+1221	+723	0
+1233	+723	8	2	1	+1.28761654345589527e+00	+1.05901896694667812e+01	+1.23469247471207411e-01	+7.48421357280370847e-01	+1217	+723	0	+1223	+723	0
+1233	+725	8	2	1	+1.28673364256860578e+00	+1.05893067685794744e+01	+1.23469247471207411e-01	+7.50548407957166930e-01	+1217	+721	0	+1223	+723	0
+1235	+723	8	2	1	+1.18876865155143063e+00	+1.06489827902127576e+01	+1.20430339980009432e-01	+7.44869112154949775e-01	+1221	+723	0	+1227	+723	0
+1235	+725	8	2	1	+1.18876865155143063e+00	+1.06489827902127576e+01	+1.20430339980009432e-01	+7.47065981062575823e-01	+1219	+723	0	+1225	+723	0
+1227	+723	8	2	1	+1.94841251968012030e+00	+9.97044148543221986e+00	+1.35729051243448112e-01	+7.62413217460477632e-01	+1211	+721	0	+1217	+723	0
+1225	+723	8	2	1	+1.92829312345800696e+00	+1.00454586321757464e+01	+1.38139930981519971e-01	+7.64437621005419676e-01	+1209	+721	0	+1215	+723	0
+1231	+725	8	2	1	+1.28705061049140213e+00	+1.05896237365022650e+01	+1.26647798223389868e-01	+7.53297980711110204e-01	+1215	+721	0	+1221	+723	0
+1237	+723	8	2	1	+5.13852709361507620e-01	+1.16177102259377687e+01	+1.17436945681185359e-01	+7.33424659229890752e-01	+1223	+723	0	+1229	+723	0
+1237	+725	8	2	1	+5.13852709361507620e-01	+1.16177102259377687e+01	+1.17436945681185359e-01	+7.35554731214881463e-01	+1221	+723	0	+1227	+723	0
+1239	+725	8	2	1	+9.74336197595183084e-01	+1.08383021904519943e+01	+1.16030721090066555e-01	+7.39605882198143161e-01	+1223	+723	0	+1229	+723	0
+1239	+723	8	2	1	+9.74336197595183084e-01	+1.08383021904519943e+01	+1.16030721090066555e-01	+7.37364796046096060e-01	+1225	+723	0	+1231	+723	0
+1215	+721	8	2	1	+2.26675512898328346e+00	+9.70300573608879802e+00	+1.49211454035692054e-01	+7.78606715232423974e-01	+1199	+721	0	+1205	+721	0
+1215	+723	8	2	1	+2.26675512898328346e+00	+9.70300573608879802e+00	+1.49211454035692054e-01	+7.80503916903835804e-01	+1197	+721	0	+1203	+721	0
+1217	+721	8	2	1	+2.18011020560196744e+00	+9.85764135145373643e+00	+1.47414580964050412e-01	+7.74779758727375589e-01	+1201	+721	0	+1207	+721	0
+1217	+723	8	2	1	+2.18011020560196744e+00	+9.85764135145373643e+00	+1.47414580964050412e-01	+7.76763013923620682e-01	+1201	+721	0	+1207	+721	0
+1213	+721	8	2	1	+2.31581153090996716e+00	+9.76712317163869059e+00	+1.51713362803840596e-01	+7.80613414608990364e-01	+1197	+719	0	+1203	+721	0
+1213	+723	8	2	1	+2.31581153090996716e+00	+9.76712317163869059e+00	+1.51713362803840596e-01	+7.82519362745410896e-01	+1195	+719	0	+1201	+721	0
+1219	+721	8	2	1	+2.17021494624861155e+00	+9.77926705150194309e+00	+1.44950455558024383e-01	+7.72842865290566006e-01	+1203	+721	0	+1209	+721	0
+1219	+723	8	2	1	+2.17021494624861155e+00	+9.77926705150194309e+00	+1.44950455558024383e-01	+7.74710497562079881e-01	+1203	+721	0	+1209	+721	0
+1221	+723	8	2	1	+2.10735923351365306e+00	+9.84315118249275578e+00	+1.42314246988610282e-01	+7.71546926645802711e-01	+1205	+721	0	+1211	+721	0
+1223	+723	8	2	1	+2.06232132583564942e+00	+9.86915743936594581e+00	+1.40552899008119153e-01	+7.68738536201678202e-01	+1207	+721	0	+1213	+721	0
+1211	+721	8	2	1	+2.35199837868200179e+00	+9.77981931564045226e+00	+1.53959056674509709e-01	+7.83054025439292256e-01	+1195	+719	0	+1201	+721	0
+1209	+721	8	2	1	+2.33002216652272010e+00	+9.74709432650195673e+00	+1.56679516692303400e-01	+7.85697849570291762e-01	+1193	+719	0	+1197	+721	0
+1211	+723	8	2	1	+2.30925450786004971e+00	+9.76056614858882909e+00	+1.53959056674509709e-01	+7.85063749098254271e-01	+1193	+719	0	+1199	+721	0
+1201	+721	8	2	1	+9.10322925604501121e-01	+1.16701478428117049e+01	+1.50111569449324994e-01	+7.77339786266858512e-01	+1183	+719	0	+1189	+719	0
+1199	+721	8	2	1	+8.31114049336871497e-01	+1.18016455045809643e+01	+1.50111569449324994e-01	+7.78347956791915929e-01	+1181	+719	0	+1187	+719	0
+1203	+721	8	2	1	+6.62824766671242349e-01	+1.19195407611455924e+01	+1.50510998396797574e-01	+7.72660672698785356e-01	+1185	+719	0	+1191	+719	0
+1205	+721	8	2	1	+2.46843989411816889e+00	+9.66009133382044816e+00	+1.62006106189354293e-01	+7.91345996363102766e-01	+1187	+719	0	+1193	+719	0
+1197	+721	8	2	1	+1.07808591619835648e+00	+1.15365903681404340e+01	+1.50111569449324994e-01	+7.83155208768489319e-01	+1179	+719	0	+1185	+719	0
+1207	+721	8	2	1	+2.31582085709478003e+00	+9.73835921482021405e+00	+1.59860863709815754e-01	+7.88228209229020149e-01	+1189	+719	0	+1195	+721	0
+1197	+719	8	2	1	+1.07365456071629750e+00	+1.15321590126583970e+01	+1.50111569449324994e-01	+7.81485969412628823e-01	+1179	+719	0	+1185	+719	0
+1199	+719	8	2	1	+8.22872122766404979e-01	+1.17934035780105280e+01	+1.50111569449324994e-01	+7.76829073608271803e-01	+1181	+719	0	+1187	+719	0
+1195	+719	8	2	1	+1.17819275846607363e+00	+1.14509759175445431e+01	+1.50215701709965094e-01	+7.84453911450972541e-01	+1177	+717	0	+1183	+719	0
+1195	+721	8	2	1	+1.17819275846607363e+00	+1.14509759175445431e+01	+1.50215701709965094e-01	+7.86195135981753834e-01	+1177	+717	0	+1183	+719	0
+1193	+719	8	2	1	+1.22158196221105309e+00	+1.14201508842560671e+01	+1.50674724578006930e-01	+7.86985356425363825e-01	+1175	+717	0	+1181	+719	0
+1193	+721	8	2	1	+1.22187048887099925e+00	+1.14204394109160141e+01	+1.50674724578006930e-01	+7.88571357929574779e-01	+1175	+717	0	+1181	+719	0
+1185	+719	8	2	1	+1.26443879388083147e+00	+1.14832732137744529e+01	+1.59864030322748318e-01	+7.94856511820003830e-01	+1167	+717	0	+1171	+717	0
+1187	+719	8	2	1	+1.39256592874030938e+00	+1.12785066266825069e+01	+1.57075885990791558e-01	+7.94725169459753022e-01	+1169	+717	0	+1175	+717	0
+1183	+719	8	2	1	+1.52646479742651864e+00	+1.11873338224643923e+01	+1.62338325536125511e-01	+7.99787727393603265e-01	+1163	+717	0	+1169	+717	0
+1189	+719	8	2	1	+1.11145261241266624e+00	+1.15922117283251911e+01	+1.54392947930311691e-01	+7.89692894387518107e-01	+1171	+717	0	+1177	+719	0
+1181	+719	8	2	1	+1.39381183385798368e+00	+1.13914798060205484e+01	+1.65289795201190359e-01	+7.99882313100254816e-01	+1161	+717	0	+1167	+717	0
+1191	+719	8	2	1	+1.07848767174582383e+00	+1.16163308293322718e+01	+1.51371980846002480e-01	+7.87305900084174337e-01	+1173	+717	0	+1179	+719	0
+1179	+719	8	2	1	+1.41850196188733157e+00	+1.13751380793181127e+01	+1.67402604129068100e-01	+8.02046042811012438e-01	+1159	+717	0	+1165	+717	0
+1177	+717	8	2	1	+1.48891624910606191e+00	+1.13356177164279970e+01	+1.69509117257236330e-01	+8.03013494657171645e-01	+1157	+715	0	+1163	+717	0



Smale's 14th Problem

Lor

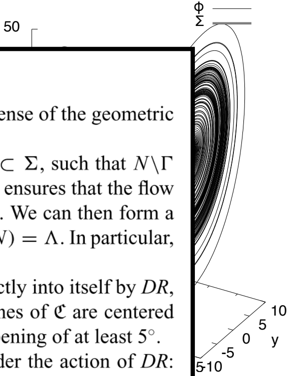


Tuc

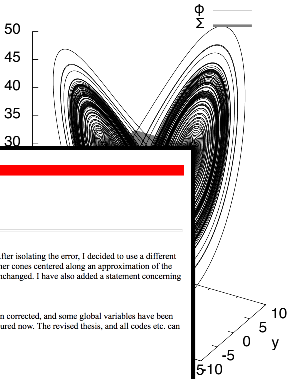
The global properties we will prove are the following:

- The return map R exists, and it is well defined in the sense of the geometric model.
- There exists a compact subset of the return plane, $N \subset \Sigma$, such that $N \backslash \Gamma$ is *forward invariant* under R, i.e., $R(N \backslash \Gamma) \subset N$. This ensures that the flow has an attracting set $\mathcal{A}$ with a large basin of attraction. We can then form a cross-section of the attracting set: $\mathcal{A} \cap \Sigma = \bigcap_{n=0}^{\infty} R^n(N) = \Lambda$. In particular, Λ is an attracting set for R.
- On N, there exists a cone field $\mathfrak{C}$ which is mapped strictly into itself by DR, i.e., for all $x \in N$, $DR(x) \cdot \mathfrak{C}(x) \subset \mathfrak{C}(R(x))$. The cones of $\mathfrak{C}$ are centered along an approximation of Λ, and each cone has an opening of at least 5°.
- The tangent vectors in $\mathfrak{C}$ are eventually expanded under the action of DR: there exists $C > 0$ and $\lambda > 1$ such that for all $v \in \mathfrak{C}(x)$, $x \in N$, we have $|DR^n(x)v| \geq C\lambda^n|v|, n \geq 0$. In fact, the expansion is strong enough to ensure that R is topologically transitive on Λ. This is equivalent to having a dense orbit, and therefore proves that Λ is an attractor.

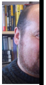
code)



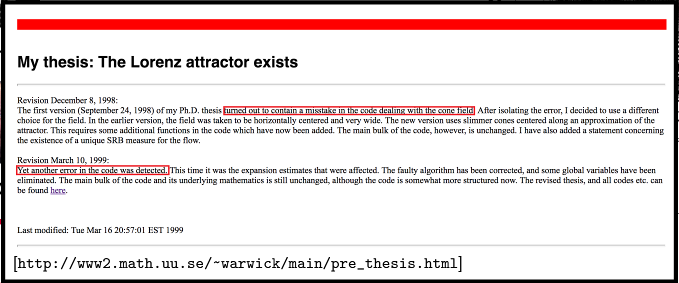
Smale's 14th Problem



- Lor



- Tu
  - normal form theory (25 pages)
  - C++ Program (24 pages, 3800+8800 lines of numerical code)



My thesis: The Lorenz attractor exists

Revision December 8, 1998:
The first version (September 24, 1998) of my Ph.D. thesis turned out to contain a misstake in the code dealing with the cone field. After isolating the error, I decided to use a different choice for the field. In the earlier version, the field was taken to be horizontally centered and very wide. The new version uses slimmer cones centered along an approximation of the attractor. This requires some additional functions in the code which have now been added. The main bulk of the code, however, is unchanged. I have also added a statement concerning the existence of a unique SRB measure for the flow.

Revision March 10, 1999:
Yet another error in the code was detected. This time it was the expansion estimates that were affected. The faulty algorithm has been corrected, and some global variables have been eliminated. The main bulk of the code and its underlying mathematics is still unchanged, although the code is somewhat more structured now. The revised thesis, and all codes etc. can be found here.

Last modified: Tue Mar 16 20:57:01 EST 1999

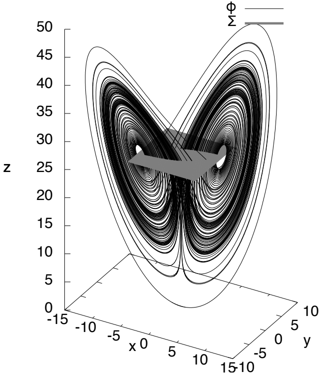
[http://www2.math.uu.se/~warwick/main/pre_thesis.html]

Smale's 14th Problem

- 

  Lorenz (1963): is this chaos?
- 

  Tucker (2002): Yes:
  - 1 paragraph combining standard results
  - normal form theory (25 pages)
  - C++ Program (24 pages, 3800+8800 lines of numerical code)

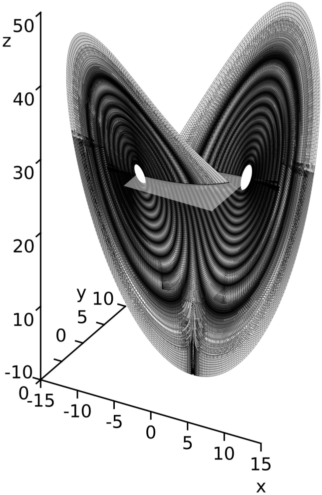


Smale's 14th Problem

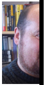
- 

  Lorenz (1963): is this chaos?
- 

  Tucker (2002): Yes:
  - 1 paragraph combining standard results
  - normal form theory (25 pages)
  - C++ Program (24 pages, 3800+8800 lines of numerical code)
- Immler (2018):
  verified numerical computations

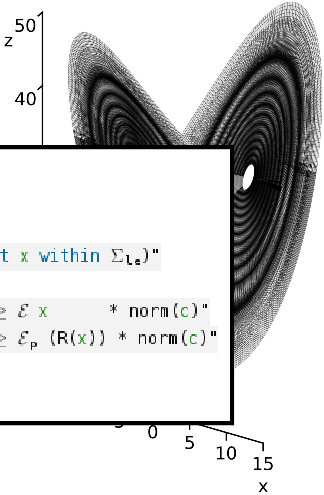


Smale's 14th Problem

- 
  Lor
- 
  Tuc
  -

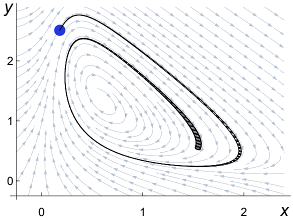
```
theorem lorenz_bounds:
  "∀x ∈ N - Γ. x returns_to Σ"
  "∀x ∈ N - Γ. R(x) ∈  N"
  "∀x ∈ N - Γ. (R has_derivative DR(x)) (at x within Σ_le)"
  "∀x ∈ N - Γ. DR(x)`(ℭ x) ⊆ ℭ(R(x))"
  "∀x ∈ N - Γ. ∀c ∈ ℭ(x). norm (DR(x) c) ≥ ℰ x      * norm(c)"
  "∀x ∈ N - Γ. ∀c ∈ ℭ(x). norm (DR(x) c) ≥ ℰ_p (R(x)) * norm(c)"
  if normal_form_correct
```

  - normal form theory (25 pages)
  - C++ Program (24 pages, 3800+8800 lines of numerical code)
- Immler (2018): verified numerical computations



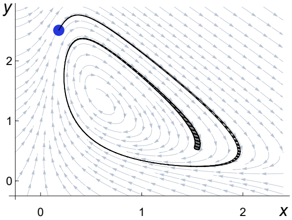
Summary of Rigorous Numerical Results

- Theorem: computed enclosures contain solution



Summary of Rigorous Numerical Results

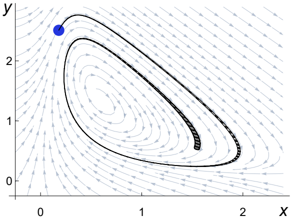
- Theorem: computed enclosures contain solution



- Applications:
  - Smale's 14th problem
  - (motion planning for autonomous vehicles)
  - (ARCH-Software Competition)

Summary of Rigorous Numerical Results

- Theorem: computed enclosures contain solution



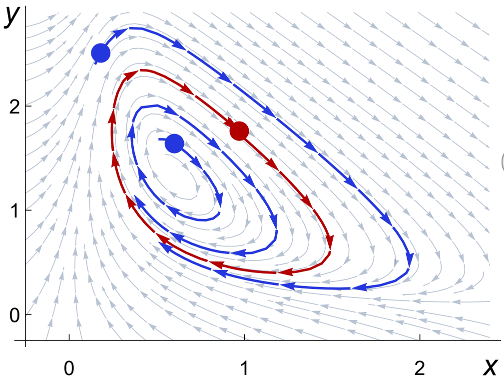
- Applications:
  - Smale's 14th problem
  - (motion planning for autonomous vehicles)
  - (ARCH-Software Competition)

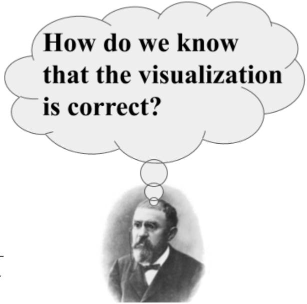
Problem

- concrete values, bounds, finite time

Outline

The Poincaré-Bendixson Theorem





Theorem (Poincaré-Bendixson)

(Under mild assumptions) trajectories of planar dynamical systems are either periodic or tend towards a periodic trajectory.

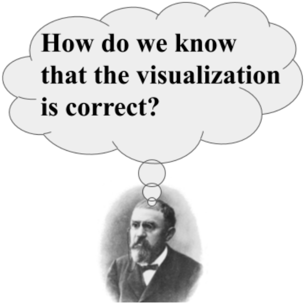
The Poincaré-Bendixson Theorem

In our paper/formalization:

```
theorem poincare_bendixson:
assumes xK: "compact K" "K ⊆ X" "x ∈ X"
 "trapped_forward x K"
assumes "0 ∉ f ‘ (ω_limit_set x)"
obtains y where
 "periodic_orbit y"
 "flow0 y ‘ UNIV = ω_limit_set x"
```

The final theorem (some proof steps omitted) shows that a limit cycle exists within the trapping region gK, and thus that Sel'kov's model exhibits limiting periodic behavior:

```
theorem g_has_limit_cycle:
obtains y where
 "g.limit_cycle y" "g.flow0 y ‘ UNIV ⊆ gK"
```



Theorem (Poincaré-Bendixson)

(Under mild assumptions) trajectories of planar dynamical systems are either periodic or tend towards a periodic trajectory.

Outline

Formalization Challenges (the “easy” ones)

Theorem (Poincaré-Bendixson)

(Under mild assumptions) trajectories of planar dynamical systems are either periodic or tend towards a periodic trajectory.

1. Has substantial prerequisite formalized mathematics, e.g.: the Jordan curve theorem, (real) analysis, ODEs.

Formalization Challenges (the “easy” ones)

Theorem (Poincaré-Bendixson)

(Under mild assumptions) trajectories of planar dynamical systems are either periodic or tend towards a periodic trajectory.

1. Has substantial prerequisite formalized mathematics, e.g.: the Jordan curve theorem, (real) analysis, ODEs.

   ✓Isabelle/HOL and the Archive of Formal Proofs (AFP) meet these prerequisites.

Formalization Challenges (the “easy” ones)

Theorem (Poincaré-Bendixson)

(Under mild assumptions) trajectories of planar dynamical systems are either periodic or tend towards a periodic trajectory.

1. Has substantial prerequisite formalized mathematics, e.g.: the Jordan curve theorem, (real) analysis, ODEs.

   ✓Isabelle/HOL and the Archive of Formal Proofs (AFP) meet these prerequisites.

2. Needs formalization of key dynamical systems concepts, e.g.: limit sets of trajectories, periodic orbits.

Formalization Challenges (the “easy” ones)

Theorem (Poincaré-Bendixson)

(Under mild assumptions) trajectories of planar dynamical systems are either periodic or tend towards a periodic trajectory.

1. Has substantial prerequisite formalized mathematics, e.g.: the Jordan curve theorem, (real) analysis, ODEs.

   ✓Isabelle/HOL and the Archive of Formal Proofs (AFP) meet these prerequisites.

2. Needs formalization of key dynamical systems concepts, e.g.: limit sets of trajectories, periodic orbits.

   ✓Mostly involves formalizing of (real) analysis-type arguments following standard presentations in textbooks.

Formalization Challenges (this talk)

Theorem (Poincaré-Bendixson)

(Under mild assumptions) trajectories of planar dynamical systems are either periodic or tend towards a periodic trajectory.

3. Textbook proofs rely *heavily on sketches*, especially for a key lemma that is fundamental to the plane.

Formalization Challenges (this talk)

Theorem (Poincaré-Bendixson)

(Under mild assumptions) trajectories of planar dynamical systems are either periodic or tend towards a periodic trajectory.

3. Textbook proofs rely *heavily on sketches*, especially for a key lemma that is fundamental to the plane.

Hartman:

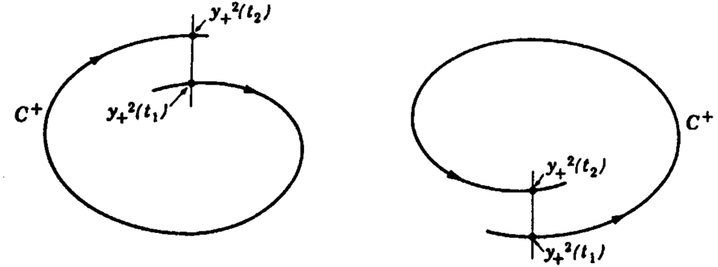


Figure 5.

Formalization Challenges (this talk)

Theorem (Poincaré-Bendixson)

(Under mild assumptions) trajectories of planar dynamical systems are either periodic or tend towards a periodic trajectory.

3. Textbook proofs rely *heavily on sketches*, especially for a key lemma that is fundamental to the plane.

Palis & de Melo:

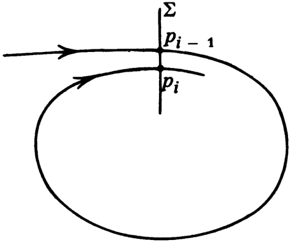


Figure 10

Formalization Challenges (this talk)

Theorem (Poincaré-Bendixson)

(Under mild assumptions) trajectories of planar dynamical systems are either periodic or tend towards a periodic trajectory.

3. Textbook proofs rely *heavily on sketches*, especially for a key lemma that is fundamental to the plane.

Perko:

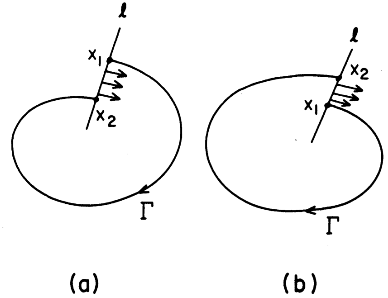


Figure 1. A Jordan curve defined by Γ and ℓ.

Formalization Challenges (this talk)

Theorem (Poincaré-Bendixson)

(Under mild assumptions) trajectories of planar dynamical systems are either periodic or tend towards a periodic trajectory.

3. Textbook proofs rely *heavily on sketches*, especially for a key lemma that is fundamental to the plane.

Wiggins:

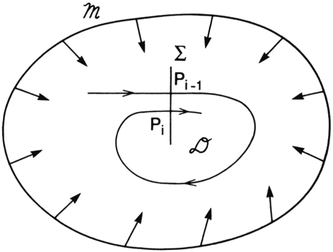


FIGURE 9.0.1.

Formalization Challenges (this talk)

Theorem (Poincaré-Bendixson)

(Under mild assumptions) trajectories of planar dynamical systems are either periodic or tend towards a periodic trajectory.

3. Textbook proofs rely *heavily on sketches*, especially for a key lemma that is fundamental to the plane.

Chicone:

Proof. The proof is left as an exercise. Hint: Reduce to the case where t_1, t_2, and t_3 correspond to consecutive crossing points. Then, consider the curve formed by the union of $\{\phi_t(p) : t_1 \leq t \leq t_2\}$ and the subset of Σ between $\phi_{t_1}(p)$ and $\phi_{t_2}(p)$. Draw a picture. $\square$

Formalization Challenges (this talk)

Theorem (Poincaré-Bendixson)

(Under mild assumptions) trajectories of planar dynamical systems are either periodic or tend towards a periodic trajectory.

3. Textbook proofs rely *heavily on sketches*, especially for a key lemma that is fundamental to the plane.

   ? How can we formalize these sketches in a proof assistant?

Formalization Challenges (this talk)

Theorem (Poincaré-Bendixson)

(Under mild assumptions) trajectories of planar dynamical systems are either periodic or tend towards a periodic trajectory.

3. Textbook proofs rely *heavily on sketches*, especially for a key lemma that is fundamental to the plane.

   ? How can we formalize these sketches in a proof assistant?

4. Textbook proofs argue *by symmetry* and present only one of the (several) cases required.

   ? How can we effectively formalize these symmetries in order to minimize duplicated proof effort?

Outline

The Monotonicity Lemma (Textbook Proof)

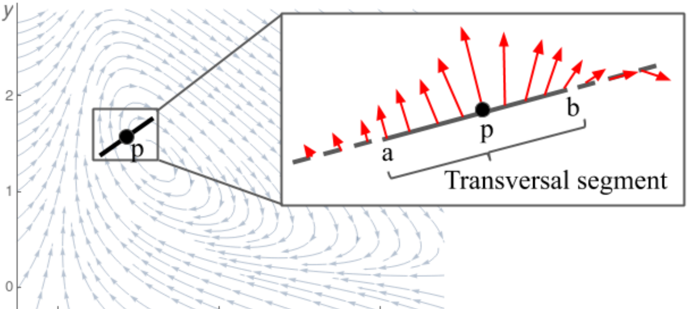
Lemma (Monotonicity)

If a trajectory (also called a flow) intersects a transversal segment, it does so monotonically in the order of the segment.

Definition (Transversal Segment)

A *transversal segment* is a (closed) 2D line segment where the RHS of the ODE is nowhere zero along the segment.

Use as Poincaré section!



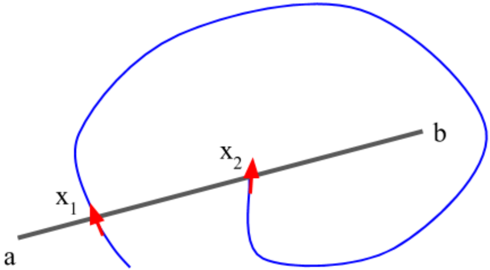
The Monotonicity Lemma (Textbook Proof)

Lemma (Monotonicity)

If a trajectory (also called a flow) intersects a transversal segment, it does so monotonically in the order of the segment.

Proof.

Suppose trajectory from x_1 on the transversal touches the transversal again at x_2:



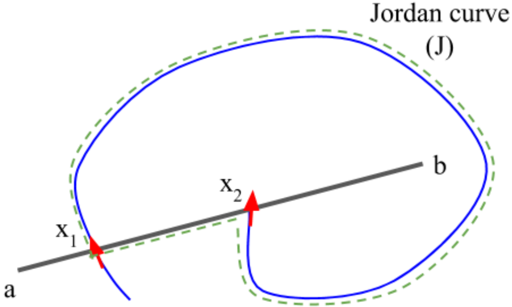
The Monotonicity Lemma (Textbook Proof)

Lemma (Monotonicity)

If a trajectory (also called a flow) intersects a transversal segment, it does so monotonically in the order of the segment.

Proof.

Construct the Jordan curve J formed by the trajectory and the segment between x_1, x_2:



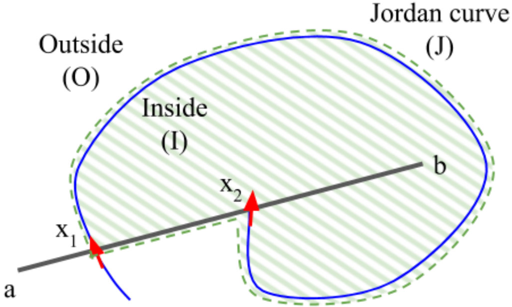
The Monotonicity Lemma (Textbook Proof)

Lemma (Monotonicity)

If a trajectory (also called a flow) intersects a transversal segment, it does so monotonically in the order of the segment.

Proof.

By the Jordan curve theorem, J separates the plane into an inside I and outside O:



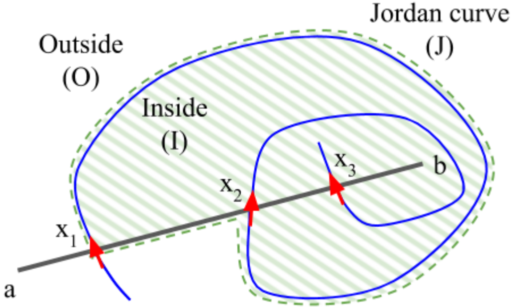
The Monotonicity Lemma (Textbook Proof)

Lemma (Monotonicity)

If a trajectory (also called a flow) intersects a transversal segment, it does so monotonically in the order of the segment.

Proof.

Any further intersection at x_3 must happen inside by construction, so the intersections are ordered $x_1 \leq x_2 \leq x_3$:



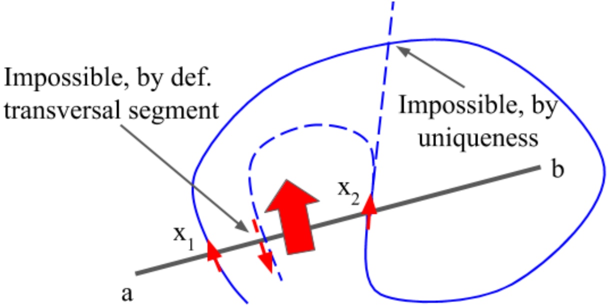
The Monotonicity Lemma (Textbook Proof)

Lemma (Monotonicity)

If a trajectory (also called a flow) intersects a transversal segment, it does so monotonically in the order of the segment.

Proof.

Any further intersection at x_3 must happen inside by construction, so the intersections are ordered $x_1 \leq x_2 \leq x_3$:



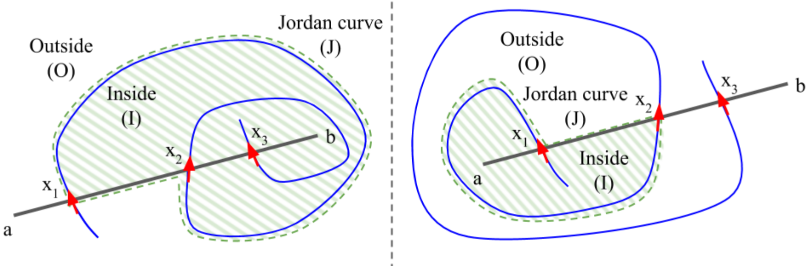
The Monotonicity Lemma (Textbook Proof)

Lemma (Monotonicity)

If a trajectory (also called a flow) intersects a transversal segment, it does so monotonically in the order of the segment.

Proof.

Not quite done! There are several other cases, but the argument for them is symmetric: □



(Left) Flow stays inside past x_2

(Right) Flow stays outside past x_2

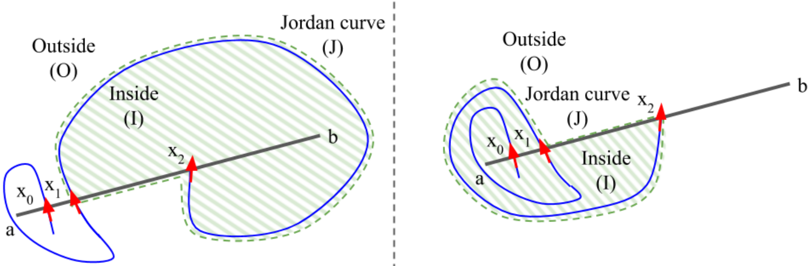
The Monotonicity Lemma (Textbook Proof)

Lemma (Monotonicity)

If a trajectory (also called a flow) intersects a transversal segment, it does so monotonically in the order of the segment.

Proof.

Not quite done! There are several other cases, but the argument for them is symmetric: □



(Left) Flow stays outside before x_1

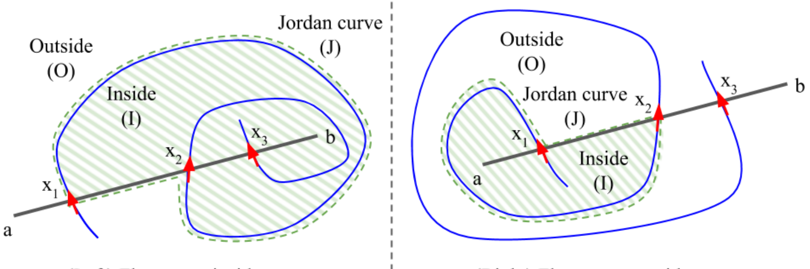
(Right) Flow stays inside before x_1

The Monotonicity Lemma (Formal Proof)

(Recall) Formalization Challenge:

3. Textbook proofs rely *heavily on sketches*, especially for a key lemma that is fundamental to the plane.

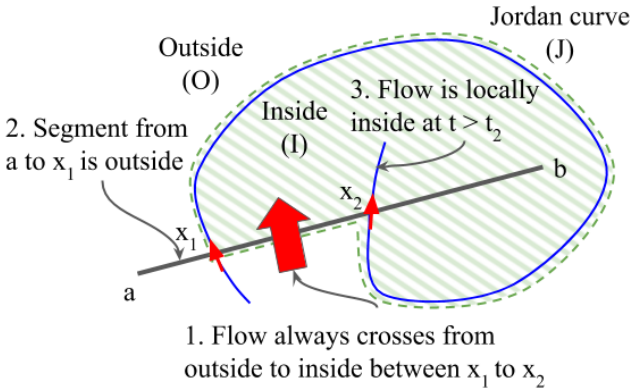
Subtle Claim: these are the *only* possibilities that can occur for J.



(Left) Flow stays inside past x_2 (Right) Flow stays outside past x_2

The Monotonicity Lemma (Formal Proof)

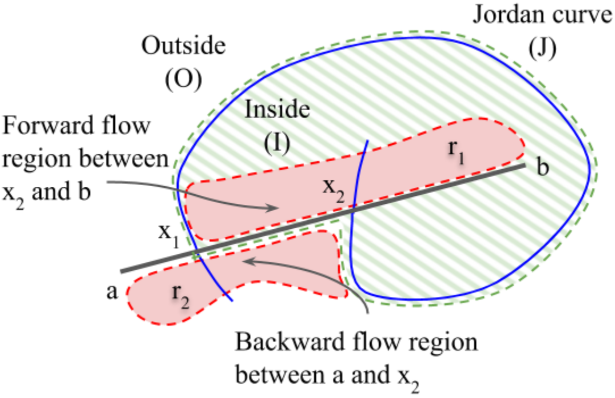
Three pieces of information are needed for the (Left) case:



Symmetrically for (Right) case, e.g., flow always crosses from inside to outside between x_1 to x_2.

The Monotonicity Lemma (Formal Proof)

We use a "flow region" construction, (Left) case shown here:



Key Idea: Flow regions r_1, r_2 must lie on opposite sides. This implies all three pieces of information (for each case, respectively).

The Monotonicity Lemma (Formal Proof)

(Recall) Formalization Challenge:

4. Textbook proofs argue *by symmetry* and present only one of the (several) cases required.

Key Idea: reverse flows to obtain other cases by symmetry.

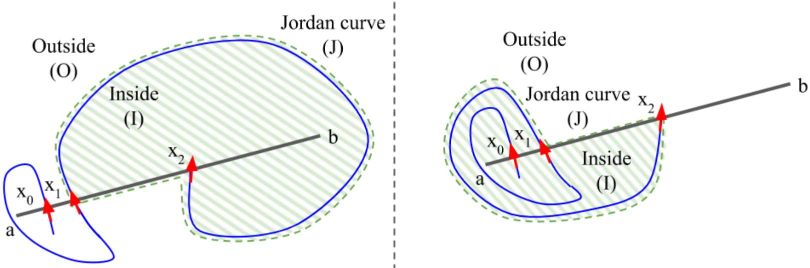
The Monotonicity Lemma (Formal Proof)

(Recall) Formalization Challenge:

4. Textbook proofs argue *by symmetry* and present only one of the (several) cases required.

Key Idea: reverse flows to obtain other cases by symmetry.

To show:



(Left) Flow stays outside before x_1 (Right) Flow stays inside before x_1

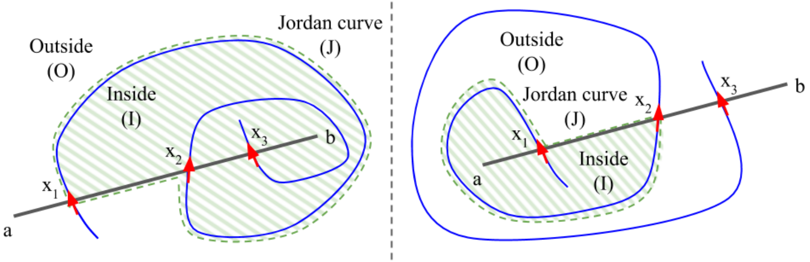
The Monotonicity Lemma (Formal Proof)

(Recall) Formalization Challenge:

4. Textbook proofs argue *by symmetry* and present only one of the (several) cases required.

Key Idea: reverse flows to obtain other cases by symmetry.

For any flow, we know (from previous slides):



(Left) Flow stays inside past x_2 (Right) Flow stays outside past x_2

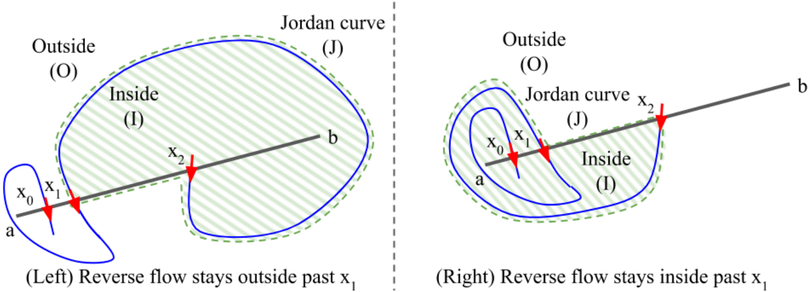
The Monotonicity Lemma (Formal Proof)

(Recall) Formalization Challenge:

4. Textbook proofs argue *by symmetry* and present only one of the (several) cases required.

Key Idea: reverse flows to obtain other cases by symmetry.

In particular, for the reversed flow:



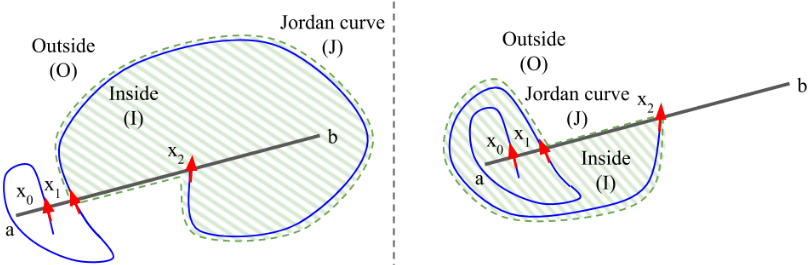
The Monotonicity Lemma (Formal Proof)

(Recall) Formalization Challenge:

4. Textbook proofs argue *by symmetry* and present only one of the (several) cases required.

Key Idea: reverse flows to obtain other cases by symmetry.

Reversing a flow twice yields the flow itself:



(Left) Flow stays outside before x_1 (Right) Flow stays inside before x_1

The Monotonicity Lemma (Formal Proof)

(Recall) Formalization Challenge:

4. Textbook proofs argue *by symmetry* and present only one of the (several) cases required.

Key Idea: reverse flows to obtain other cases by symmetry.

Using (sub)locales

ODE $f : \mathbb{R}^n \to \mathbb{R}^n$

ϕ, Thm $P(\phi(x_0, t))$

The Monotonicity Lemma (Formal Proof)

(Recall) Formalization Challenge:

4. Textbook proofs argue *by symmetry* and present only one of the (several) cases required.

Key Idea: reverse flows to obtain other cases by symmetry.

Using (sub)locales

ODE $f : \mathbb{R}^n \to \mathbb{R}^n$

ϕ, Thm $P(\phi(x_0, t))$

rev.ODE $-f : \mathbb{R}^n \to \mathbb{R}^n$

ϕ_{-f}, Thm $P(\phi_{-f}(x_0, t))$

The Monotonicity Lemma (Formal Proof)

(Recall) Formalization Challenge:

4. Textbook proofs argue *by symmetry* and present only one of the (several) cases required.

Key Idea: reverse flows to obtain other cases by symmetry.

Using (sub)locales

ODE $f : \mathbb{R}^n \to \mathbb{R}^n$

ϕ, Thm $P(\phi(x_0, t))$

rev.ODE $-f : \mathbb{R}^n \to \mathbb{R}^n$

ϕ_{-f}, Thm $P(\phi_{-f}(x_0, t))$

$\Downarrow$

export and rewrite $\phi_{-f}(x_0, t) = \phi(x_0, -t)$

$\Downarrow$

The Monotonicity Lemma (Formal Proof)

(Recall) Formalization Challenge:

4. Textbook proofs argue *by symmetry* and present only one of the (several) cases required.

Key Idea: reverse flows to obtain other cases by symmetry.

Using (sub)locales

ODE $f : \mathbb{R}^n \to \mathbb{R}^n$

ϕ, Thm $P(\phi(x_0, t))$

rev.ODE $-f : \mathbb{R}^n \to \mathbb{R}^n$

ϕ_{-f}, Thm $P(\phi_{-f}(x_0, t))$

$\Downarrow$

export and rewrite $\phi_{-f}(x_0, t) = \phi(x_0, -t)$

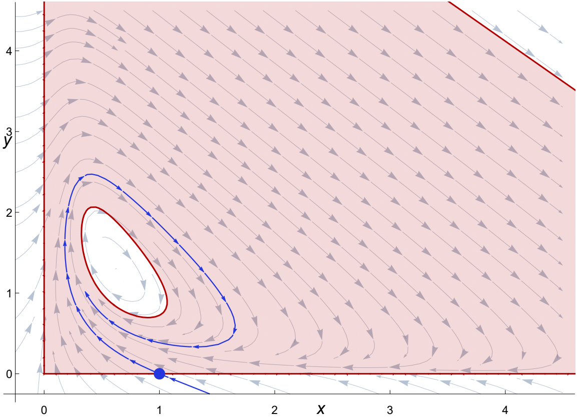
$\Downarrow$

rev.Thm: $P(\phi(x_0, -t))$

Outline

Example Application

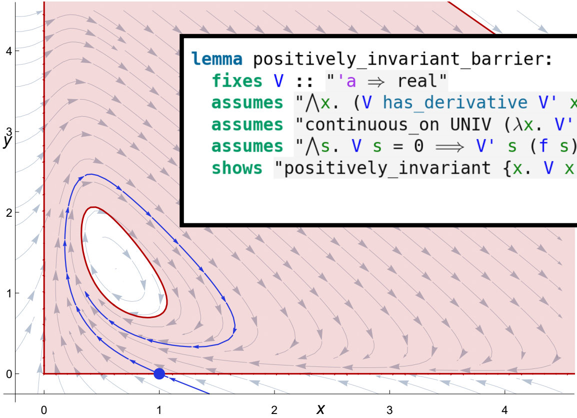
```
corollary poincare_bendixson_limit_cycle:
  assumes "compact K" "K ⊆ X"
  assumes "x ∈ K" "positively_invariant K"
  assumes "0 ∉ f ` K"
  assumes "flow0 x t ∉ K"
  obtains y where "limit_cycle y" "flow0 y ` UNIV ⊆ K"
```



- comparison principle, barrier certificate

Example Application

```
corollary poincare_bendixson_limit_cycle:
  assumes "compact K" "K ⊆ X"
  assumes "x ∈ K" "positively_invariant K"
  assumes "0 ∉ f ` K"
  assumes "flow0 x t ∉ K"
  obtains y where "limit_cycle y" "flow0 y ` UNIV ⊆ K"
```

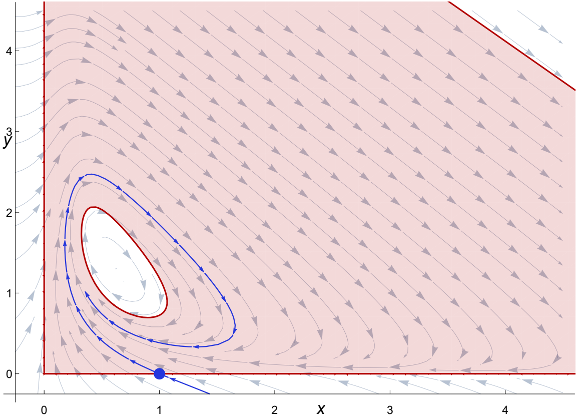



```
lemma positively_invariant_barrier:
  fixes V :: "'a ⇒ real"
  assumes "⋀x. (V has_derivative V' x) (at x)"
  assumes "continuous_on UNIV (λx. V' x (f x))"
  assumes "⋀s. V s = 0 ⟹ V' s (f s) < 0"
  shows "positively_invariant {x. V x ≤ 0}"
```

comparison principle, barrier certificate

Example Application

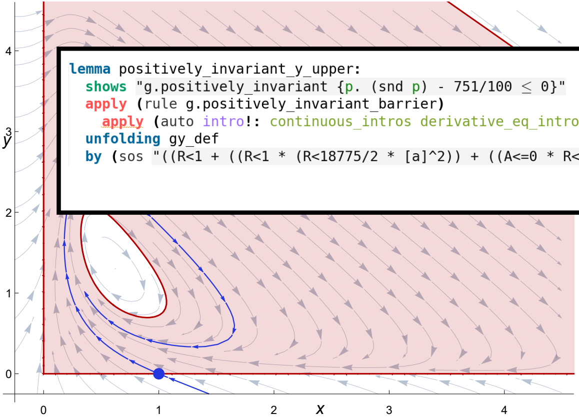
```
corollary poincare_bendixson_limit_cycle:
  assumes "compact K" "K ⊆ X"
  assumes "x ∈ K" "positively_invariant K"
  assumes "0 ∉ f ` K"
  assumes "flow0 x t ∉ K"
  obtains y where "limit_cycle y" "flow0 y ` UNIV ⊆ K"
```



- comparison principle, barrier certificate
- SOS

Example Application

```
corollary poincare_bendixson_limit_cycle:
  assumes "compact K" "K ⊆ X"
  assumes "x ∈ K" "positively_invariant K"
  assumes "0 ∉ f ` K"
  assumes "flow0 x t ∉ K"
  obtains y where "limit_cycle y" "flow0 y ` UNIV ⊆ K"
```



- comparison

```
lemma positively_invariant_y_upper:
  shows "g.positively_invariant {p. (snd p) - 751/100 ≤ 0}"
  apply (rule g.positively_invariant_barrier)
    apply (auto intro!: continuous_intros derivative_eq_intros)
  unfolding gy_def
  by (sos "((R<1 + ((R<1 * (R<18775/2 * [a]^2)) + ((A<=0 * R<1) * (R<1250 * [1]^2)))))")
```

Example Application

```
corollary poincare_bendixson_limit_cycle:
```

```
lemma positively_invariant_trapG:
  shows "g.positively_invariant trapG"
  unfolding trapG_def
  apply (rule g.positively_invariant_le_domain[OF positively_invariant_trapG1 _ p1_has_derivative,
        of "λp. -1.08 - (fst p)^2 + 2 * fst p * snd p"])
  subgoal by (auto intro!: continuous_intros derivative_eq_intros simp add: pos_quad_def)
  apply (auto simp: p1d_def gx_def gy_def trapG1_def pos_quad_def p1_def)
```

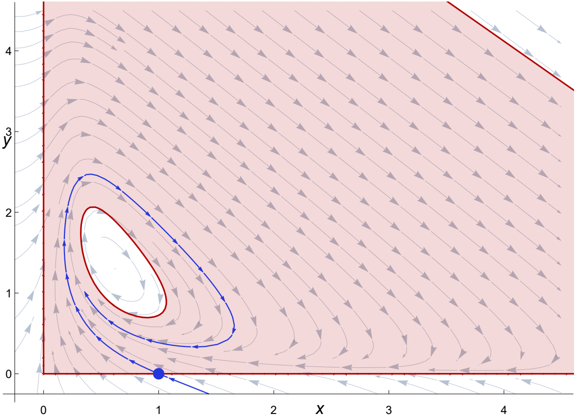
Auto update | Update | Locate | Search:

```
proof (prove)
goal (1 subgoal):
 1. ⋀a b. 0 ≤ b ⟹
          0 ≤ a ⟹
          b * 100 ≤ 751 ⟹
          a * 25 + b * 25 ≤ 203 ⟹
          38 * ((2 * b / 25 - a + a² * b) * a) / 15 - (138 * b / 25 - 69 * a + 69 * (a² * b)) / 38 -
          27 * ((2 * b / 25 - a + a² * b) * a²) / 28 -
          24 * ((2 * b / 25 - a + a² * b) * a ^ 3) / 43 +
          (42 / 5 - 28 * b / 25 - 14 * (a² * b)) / 29 +
          (651 * (a * (3 / 5 - 2 * b / 25 - a² * b)) + 651 * ((2 * b / 25 - a + a² * b) * b)) / 441 +
          (8554 * (a² * (3 / 5 - 2 * b / 25 - a² * b)) + 17108 * ((2 * b / 25 - a + a² * b) * (a * b))) / 2209 -
          (560 * (a ^ 3 * (3 / 5 - 2 * b / 25 - a² * b)) + 1680 * ((2 * b / 25 - a + a² * b) * (a² * b))) / 256 -
          6 * ((3 / 5 - 2 * b / 25 - a² * b) * b) / 17 -
          (36 * (a * ((3 / 5 - 2 * b / 25 - a² * b) * b)) + 18 * ((2 * b / 25 - a + a² * b) * b²)) / 81 -
          (1240 * (a² * ((3 / 5 - 2 * b / 25 - a² * b) * b)) + 1240 * ((2 * b / 25 - a + a² * b) * (a * b²))) / 400 +
          (3 / 5 - 2 * b / 25 - a² * b) * b² / 34 +
          (177 * (a * ((3 / 5 - 2 * b / 25 - a² * b) * b²)) + (2 * b / 25 - a + a² * b) * b ^ 3 * 59) / 3481
          ≤ (- (27 / 25) - a² + 2 * a * b) *
             (- (21 / 34) - 69 * a / 38 + 19 * a² / 15 - 9 * a ^ 3 / 28 - 6 * a ^ 4 / 43 + 14 * b / 29 + 31 * a * b / 21 + 182 * a² * b / 47 -
              35 * a ^ 3 * b / 16 -
              3 * b² / 17 -
              2 * a * b² / 9 -
              31 * a² * b² / 20 +
              b ^ 3 / 102 +
              a * b ^ 3 / 59)
```

0 1 2 *x* 3 4

Example Application

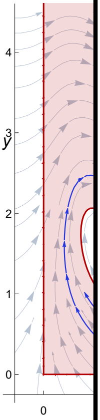
```
corollary poincare_bendixson_limit_cycle:
  assumes "compact K" "K ⊆ X"
  assumes "x ∈ K" "positively_invariant K"
  assumes "0 ∉ f ` K"
  assumes "flow0 x t ∉ K"
  obtains y where "limit_cycle y" "flow0 y ` UNIV ⊆ K"
```

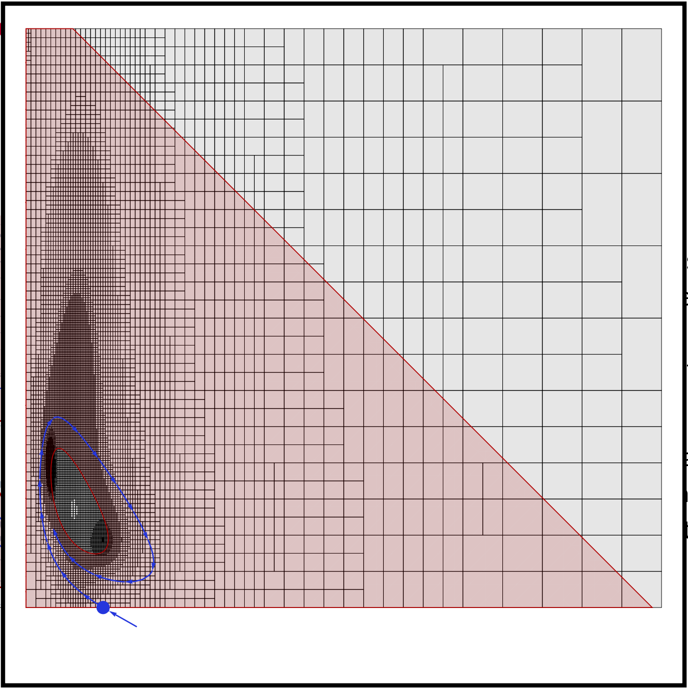


- comparison principle, barrier certificate
- SOS
- branch-and-bound affine arithmetic

Exampl

```
coroll
  assu
  assu
  assu
  assu
  obta
```





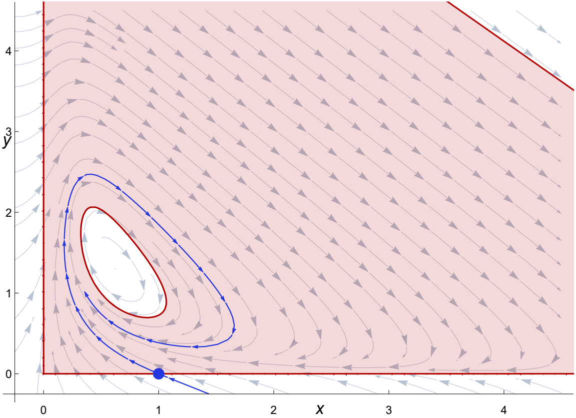
son
e,

te

and-
ffine
tic

Example Application

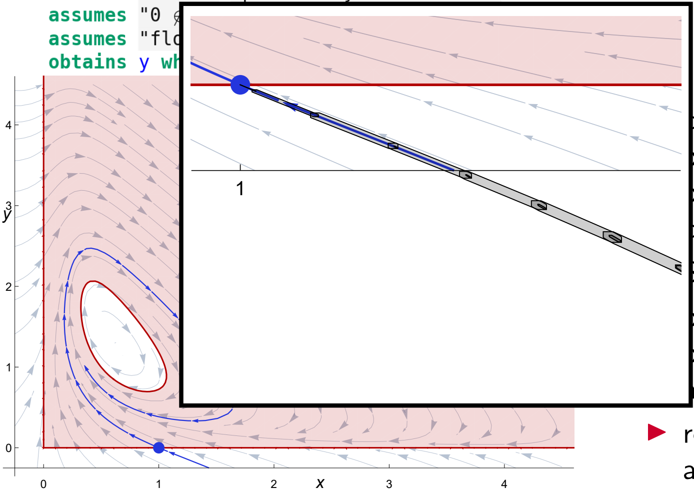
```
corollary poincare_bendixson_limit_cycle:
  assumes "compact K" "K ⊆ X"
  assumes "x ∈ K" "positively_invariant K"
  assumes "0 ∉ f ` K"
  assumes "flow0 x t ∉ K"
  obtains y where "limit_cycle y" "flow0 y ` UNIV ⊆ K"
```



- comparison principle, barrier certificate
- SOS
- branch-and-bound affine arithmetic
- reachability analysis

Example Application

```
corollary poincare_bendixson_limit_cycle:
  assumes "compact K" "K ⊆ X"
  assumes "x ∈ K" "positively invariant K"
  assumes "0 ∉
  assumes "flo
  obtains y wh
```



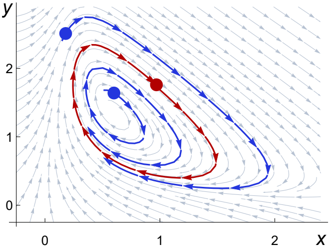
- omparison rinciple, arrier ertificate
- SOS
- ranch-and-ound affine rithmetic
- reachability analysis

Outline

Summary: Poincaré-Bendixson

Theorem (Poincaré-Bendixson)

(Under mild assumptions) trajectories of planar dynamical systems are either periodic or tend towards a periodic trajectory.




```
theorem poincare_bendixson:
assumes xK: "compact K" "K ⊆ X" "x ∈ X"
 "trapped_forward x K"
assumes "0 ∉ f ‘ (ω_limit_set x)"
obtains y where
 "periodic_orbit y"
 "flow0 y ‘ UNIV = ω_limit_set x"
```

The final theorem (some proof steps omitted) shows that a limit cycle exists within the trapping region gK, and thus that Sel’kov’s model exhibits limiting periodic behavior:

```
theorem g_has_limit_cycle:
obtains y where
 "g.limit_cycle y" "g.flow0 y ‘ UNIV ⊆ gK"
```

Formalization Challenges (the “easy” ones)

Theorem (Poincaré-Bendixson)

(Under mild assumptions) trajectories of planar dynamical systems are either periodic or tend towards a periodic trajectory.

1. Has substantial prerequisite formalized mathematics, e.g.: the Jordan curve theorem, (real) analysis, ODEs.

   ✓Isabelle/HOL and the Archive of Formal Proofs (AFP) meet these prerequisites.

2. Needs formalization of key dynamical systems concepts, e.g.: limit sets of trajectories, periodic orbits.

   ✓Mostly involves formalizing of (real) analysis-type arguments following standard presentations in textbooks.

Formalization Challenges (the "easy" ones)

Theorem (Poincaré-Bendixson)

(Under mild assumptions) trajectories of planar dynamical systems are either periodic or tend towards a periodic trajectory.

1. Has substantial prerequisite formalized mathematics, e.g.: the Jordan curve theorem, (real) analysis, ODEs.

   ✓For Yong Kiam (Isabelle/HOL beginner), Sledgehammer and search features were *very useful* for discovering existing lemmas.

2. Needs formalization of key dynamical systems concepts, e.g.: limit sets of trajectories, periodic orbits.

   ✓Mostly involves formalizing of (real) analysis-type arguments following standard presentations in textbooks.

Formalization Challenges (this talk)

Theorem (Poincaré-Bendixson)

(Under mild assumptions) trajectories of planar dynamical systems are either periodic or tend towards a periodic trajectory.

3. Textbook proofs rely *heavily on sketches*, especially for a key lemma that is fundamental to the plane.

   ✓We give the first (as far as we know*) fully rigorous argument for this step, that is amenable to formalization.

4. Textbook proofs argue *by symmetry* and present only one of the (several) cases required.

   ✓Use Isabelle/HOL's locale system to formally reverse flows.

---

*While preparing these slides, we came across a proof in Cronin based on indexes but we have not attempted to formalize it.

Formalization Challenges (this talk)

Theorem (Poincaré-Bendixson)

(Under mild assumptions) trajectories of planar dynamical systems are either periodic or tend towards a periodic trajectory.

3. Textbook proofs rely *heavily on sketches*, especially for a key lemma that is fundamental to the plane.

   ? But our proof is *rather different* from the textbook sketches. Is this unavoidable? Are there cleaner or more abstract proofs?

4. Textbook proofs argue *by symmetry* and present only one of the (several) cases required.

   ✓ Use Isabelle/HOL's locale system to formally reverse flows.

Formalization Challenges (this talk)

Theorem (Poincaré-Bendixson)

(Under mild assumptions) trajectories of planar dynamical systems are either periodic or tend towards a periodic trajectory.

3. Textbook proofs rely *heavily on sketches*, especially for a key lemma that is fundamental to the plane.

   ? But our proof is *rather different* from the textbook sketches. Is this unavoidable? Are there cleaner or more abstract proofs?

4. Textbook proofs argue *by symmetry* and present only one of the (several) cases required.

   ? How easily can this (entire) formalization be done in another proof assistant?

The Role of the Proof Assistant

- agnostic w.r.t. foundations

The Role of the Proof Assistant

- agnostic w.r.t. foundations
- no (deeply prover specific) formalization tricks

The Role of the Proof Assistant

- agnostic w.r.t. foundations
- no (deeply prover specific) formalization tricks
  - expose ordering on line segments

The Role of the Proof Assistant

- agnostic w.r.t. foundations
- no (deeply prover specific) formalization tricks
  - expose ordering on line segments
  - time reversal (module system, locales!)

The Role of the Proof Assistant

- agnostic w.r.t. foundations
- no (deeply prover specific) formalization tricks
  - expose ordering on line segments
  - time reversal (module system, locales!)
  - filters

The Role of the Proof Assistant

- agnostic w.r.t. foundations
- no (deeply prover specific) formalization tricks
  - expose ordering on line segments
  - time reversal (module system, locales!)
  - filters
  - generalizations

The Role of the Proof Assistant

- agnostic w.r.t. foundations
- no (deeply prover specific) formalization tricks
  - expose ordering on line segments
  - time reversal (module system, locales!)
  - filters
  - generalizations
- most important: libraries

The Role of the Proof Assistant

- agnostic w.r.t. foundations
- no (deeply prover specific) formalization tricks
  - expose ordering on line segments
  - time reversal (module system, locales!)
  - filters
  - generalizations
- most important: libraries
- also important: automation

The Role of the Proof Assistant

- agnostic w.r.t. foundations
- no (deeply prover specific) formalization tricks
  - expose ordering on line segments
  - time reversal (module system, locales!)
  - filters
  - generalizations
- most important: libraries
- also important: automation
  - sledgehammer for library search

The Role of the Proof Assistant

- agnostic w.r.t. foundations
- no (deeply prover specific) formalization tricks
  - expose ordering on line segments
  - time reversal (module system, locales!)
  - filters
  - generalizations
- most important: libraries
- also important: automation
  - sledgehammer for library search
  - SOS

The Role of the Proof Assistant

- agnostic w.r.t. foundations
- no (deeply prover specific) formalization tricks
  - expose ordering on line segments
  - time reversal (module system, locales!)
  - filters
  - generalizations
- most important: libraries
- also important: automation
  - sledgehammer for library search
  - SOS
  - reachability analysis for approx ODE

The Role of the Proof Assistant

- agnostic w.r.t. foundations
- no (deeply prover specific) formalization tricks
  - expose ordering on line segments
  - time reversal (module system, locales!)
  - filters
  - generalizations
- most important: libraries
- also important: automation
  - sledgehammer for library search
  - SOS
  - reachability analysis for approx ODE
- would have been helpful: real arithmetic

Future Directions

Connection with Smooth Manifold Theory

Smooth Manifolds and Types to Sets for Linear Algebra in Isabelle/HOL

Fabian Immler
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA, USA
fimmler@cs.cmu.edu

Bohua Zhan
State Key Laboratory of Computer Science
Institute of Software, Chinese Academy of Sciences
Beijing, China
bzhan@ios.ac.cn

Future Directions

Connection with Smooth Manifold Theory

Smooth Manifolds and Types to Sets for Linear Algebra in Isabelle/HOL

Fabian Immler
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA, USA
fimmler@cs.cmu.edu

Bohua Zhan
State Key Laboratory of Computer Science
Institute of Software, Chinese Academy of Sciences
Beijing, China
bzhan@ios.ac.cn

1. ODEs, Poincaré-Bendixson on sphere or 2-manifold

Future Directions

Connection with Smooth Manifold Theory

Smooth Manifolds and Types to Sets for Linear Algebra in Isabelle/HOL

Fabian Immler
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA, USA
fimmler@cs.cmu.edu

Bohua Zhan
State Key Laboratory of Computer Science
Institute of Software, Chinese Academy of Sciences
Beijing, China
bzhan@ios.ac.cn

1. ODEs, Poincaré-Bendixson on sphere or 2-manifold
2. Stable manifold theorem:
   structure of the orbits approaching a hyperbolic fixed point

Future Directions

Connection with Smooth Manifold Theory

Smooth Manifolds and Types to Sets for Linear Algebra in Isabelle/HOL

Fabian Immler
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA, USA
fimmler@cs.cmu.edu

Bohua Zhan
State Key Laboratory of Computer Science
Institute of Software, Chinese Academy of Sciences
Beijing, China
bzhan@ios.ac.cn

1. ODEs, Poincaré-Bendixson on sphere or 2-manifold
2. Stable manifold theorem:
   structure of the orbits approaching a hyperbolic fixed point

Dynamical Systems

1. Planar: Liénard's theorem, Dulac's criterion

Future Directions

Connection with Smooth Manifold Theory

Smooth Manifolds and Types to Sets for Linear Algebra in Isabelle/HOL

Fabian Immler
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA, USA
fimmler@cs.cmu.edu

Bohua Zhan
State Key Laboratory of Computer Science
Institute of Software, Chinese Academy of Sciences
Beijing, China
bzhan@ios.ac.cn

1. ODEs, Poincaré-Bendixson on sphere or 2-manifold
2. Stable manifold theorem:
   structure of the orbits approaching a hyperbolic fixed point

Dynamical Systems

1. Planar: Liénard's theorem, Dulac's criterion
2. Hartman-Grobman theorem:
   linearized system predicts qualitative behavior

Thank you. Questions?