Verified Numerics for ODEs in Isabelle/HOL

Fabian Immler

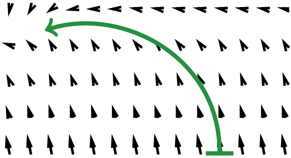
MAP 2016

Technische Universität München

Overview I

Ordinary Differential Equations

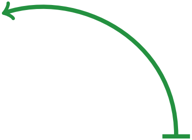
- modeling physics, biology, dynamical systems



Overview I

Ordinary Differential Equations

- modeling physics, biology, dynamical systems
- no closed form solution



Overview I

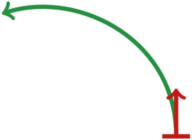
Ordinary Differential Equations

- modeling physics, biology, dynamical systems
- no closed form solution
- numerical algorithms: approximate, finite precision

Overview I

Ordinary Differential Equations

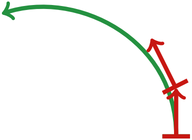
- modeling physics, biology, dynamical systems
- no closed form solution
- numerical algorithms: approximate, finite precision



Overview I

Ordinary Differential Equations

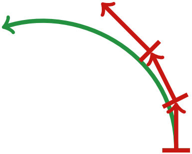
- modeling physics, biology, dynamical systems
- no closed form solution
- numerical algorithms: approximate, finite precision



Overview I

Ordinary Differential Equations

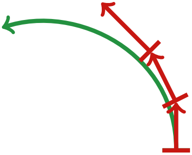
- modeling physics, biology, dynamical systems
- no closed form solution
- numerical algorithms: approximate, finite precision



Overview I

Ordinary Differential Equations

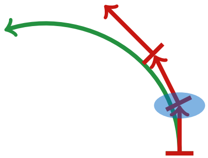
- modeling physics, biology, dynamical systems
- no closed form solution
- numerical algorithms: approximate, finite precision
- rigorous numerical algorithms: enclosures



Overview I

Ordinary Differential Equations

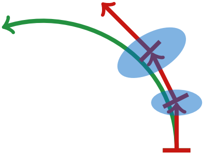
- modeling physics, biology, dynamical systems
- no closed form solution
- numerical algorithms: approximate, finite precision
- rigorous numerical algorithms: enclosures



Overview I

Ordinary Differential Equations

- modeling physics, biology, dynamical systems
- no closed form solution
- numerical algorithms: approximate, finite precision
- rigorous numerical algorithms: enclosures



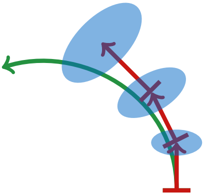
Problem

correctness of computed enclosures?

Overview I

Ordinary Differential Equations

- modeling physics, biology, dynamical systems
- no closed form solution
- numerical algorithms: approximate, finite precision
- rigorous numerical algorithms: enclosures



Problem

correctness of computed enclosures?

Overview II

Formalization and Verification

- formalization of $\mathbb{R}^n$ and ODEs in Isabelle

Overview II

Formalization and Verification

- formalization of $\mathbb{R}^n$ and ODEs in Isabelle
- verification of rigorous numerical algorithms

Overview II

Formalization and Verification

- formalization of $\mathbb{R}^n$ and ODEs in Isabelle
- verification of rigorous numerical algorithms
- refinement to executable specification

Overview II

Formalization and Verification

- formalization of $\mathbb{R}^n$ and ODEs in Isabelle
- verification of rigorous numerical algorithms
- refinement to executable specification
- code generation

Overview II

Formalization and Verification

- formalization of $\mathbb{R}^n$ and ODEs in Isabelle
- verification of rigorous numerical algorithms
- refinement to executable specification
- code generation

Result

highly trusted code

Overview III

Applications/Challenges

- Oil reservoir: stiff

Overview III

Applications/Challenges

- Oil reservoir: stiff
- van-der-Pol: nonlinear

Overview III

Applications/Challenges

- Oil reservoir: stiff
- van-der-Pol: nonlinear
- Lorenz attractor: proof of topological properties based on computed enclosures

Content

General Background

- Isabelle/HOL: interactive theorem prover

General Background

- Isabelle/HOL: interactive theorem prover
- higher order logic

General Background

- Isabelle/HOL: interactive theorem prover
- higher order logic
  - functional programming

General Background

- Isabelle/HOL: interactive theorem prover
- higher order logic
  - functional programming
  - logic

General Background

- Isabelle/HOL: interactive theorem prover
- higher order logic
  - functional programming
  - logic
- formalization

General Background

- Isabelle/HOL: interactive theorem prover
- higher order logic
  - functional programming
  - logic
- formalization
  - definition of concepts

General Background

- Isabelle/HOL: interactive theorem prover
- higher order logic
  - functional programming
  - logic
- formalization
  - definition of concepts
  - statement of properties

General Background

- Isabelle/HOL: interactive theorem prover
- higher order logic
  - functional programming
  - logic
- formalization
  - definition of concepts
  - statement of properties
- verification

General Background

- Isabelle/HOL: interactive theorem prover
- higher order logic
  - functional programming
  - logic
- formalization
  - definition of concepts
  - statement of properties
- verification
  - proofs of properties

General Background

- Isabelle/HOL: interactive theorem prover
- higher order logic
  - functional programming
  - logic
- formalization
  - definition of concepts
  - statement of properties
- verification
  - proofs of properties
  - machine-checked

Formalization of Mathematics

In Isabelle/HOL

- $\mathbb{N} \rightsquigarrow \mathbb{Z} \rightsquigarrow \mathbb{Q} \rightsquigarrow \mathbb{R}$

Formalization of Mathematics

In Isabelle/HOL

- $\mathbb{N} \rightsquigarrow \mathbb{Z} \rightsquigarrow \mathbb{Q} \rightsquigarrow \mathbb{R}$
- (Harrison's) multivariate analysis $\mathbb{R}^n$: e.g.,

Formalization of Mathematics

In Isabelle/HOL

- $\mathbb{N} \rightsquigarrow \mathbb{Z} \rightsquigarrow \mathbb{Q} \rightsquigarrow \mathbb{R}$
- (Harrison's) multivariate analysis $\mathbb{R}^n$: e.g.,
  - Taylor series expansions

Formalization of Mathematics

In Isabelle/HOL

- $\mathbb{N} \rightsquigarrow \mathbb{Z} \rightsquigarrow \mathbb{Q} \rightsquigarrow \mathbb{R}$
- (Harrison's) multivariate analysis $\mathbb{R}^n$: e.g.,
  - Taylor series expansions
  - Banach fixed point theorem

Formalization of Mathematics

In Isabelle/HOL

- $\mathbb{N} \rightsquigarrow \mathbb{Z} \rightsquigarrow \mathbb{Q} \rightsquigarrow \mathbb{R}$
- (Harrison's) multivariate analysis $\mathbb{R}^n$: e.g.,
  - Taylor series expansions
  - Banach fixed point theorem
- based on axiomatic type classes

Formalization of Mathematics

In Isabelle/HOL

- $\mathbb{N} \rightsquigarrow \mathbb{Z} \rightsquigarrow \mathbb{Q} \rightsquigarrow \mathbb{R}$
- (Harrison's) multivariate analysis $\mathbb{R}^n$: e.g.,
  - Taylor series expansions
  - Banach fixed point theorem
- based on axiomatic type classes: e.g.,

```
class metric_space =
  fixes dist::"'a ⇒ 'a ⇒ real"
  assumes "dist x y = 0 ⟷ x = y"
  assumes "dist x y ≤ dist x z + dist y z"

instance real::metric_space
sorry

instance complex::metric_space
sorry
```

Formalization of ODEs

$$\dot{\psi}(t) = f(\psi(t)); \psi(t_0) = x_0$$

Formalization of ODEs

$$\dot{\psi}(t) = f(\psi(t)); \psi(t_0) = x_0$$

- existence of unique solution
  (Picard-Lindelöf theorem)

Formalization of ODEs

$$\dot{\psi}(t) = f(\psi(t)); \psi(t_0) = x_0$$

- existence of unique solution (Picard-Lindelöf theorem)
  - $P : \mathcal{C}^{[[t_0;t_1],\mathbb{R}^n]} \to \mathcal{C}^{[[t_0;t_1],\mathbb{R}^n]}$

Formalization of ODEs

$$\dot{\psi}(t) = f(\psi(t)); \psi(t_0) = x_0$$

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

Formalization of ODEs

$$\dot{\psi}(t) = f(\psi(t)); \psi(t_0) = x_0$$

- existence of unique solution (Picard-Lindelöf theorem)
  - $P : \mathcal{C}^{[[t_0;t_1],\mathbb{R}^n]} \to \mathcal{C}^{[[t_0;t_1],\mathbb{R}^n]}$
  - $P(\psi) = (t \mapsto x_0 + \int_{t_0}^{t} f(\psi(\tau))\mathrm{d}\tau)$
  - no dependent types

Formalization of ODEs

$$\dot{\psi}(t) = f(\psi(t)); \psi(t_0) = x_0$$

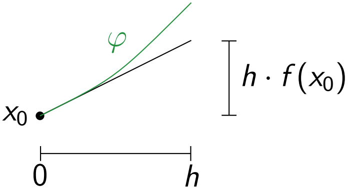
- existence of unique solution (Picard-Lindelöf theorem)
  - $P : \mathcal{C}^{[[t_0;t_1],\mathbb{R}^n]} \to \mathcal{C}^{[[t_0;t_1],\mathbb{R}^n]}$
  - $P(\psi) = (t \mapsto x_0 + \int_{t_0}^{t} f(\psi(\tau))\mathrm{d}\tau)$
  - no dependent types
  - type of bounded continuous functions $\overline{\mathcal{C}}^{[\mathbb{R},\mathbb{R}^n]}$

Formalization of ODEs

$$\dot{\psi}(t) = f(\psi(t)); \psi(t_0) = x_0$$

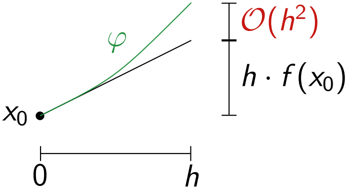
- existence of unique solution (Picard-Lindelöf theorem)
  - $P : \mathcal{C}^{[[t_0;t_1],\mathbb{R}^n]} \to \mathcal{C}^{[[t_0;t_1],\mathbb{R}^n]}$
  - $P(\psi) = (t \mapsto x_0 + \int_{t_0}^{t} f(\psi(\tau))\mathrm{d}\tau)$
  - no dependent types
  - type of bounded continuous functions $\overline{\mathcal{C}}^{[\mathbb{R},\mathbb{R}^n]}$
- flow $\varphi(x_0, t)$ (solution for initial value x_0 at time t)

Euler Method



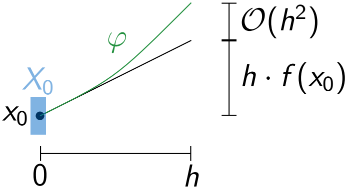
- Euler step:
  f ... slope given by ODE
  $\varphi(x_0, h) = x_0 + h \cdot f(x_0)$

Euler Method



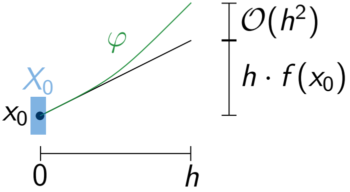
- Euler step:
  f ... slope given by ODE
  $\varphi(x_0, h) = x_0 + h \cdot f(x_0) + O(h^2)$

Euler Method



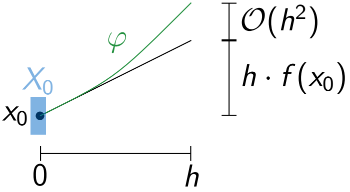
- Euler step:
  f ...slope given by ODE
  $\varphi(x_0, h) = x_0 + h \cdot f(x_0) + O(h^2)$
- set-based Euler step
  - $x_0 \in X_0$

Euler Method



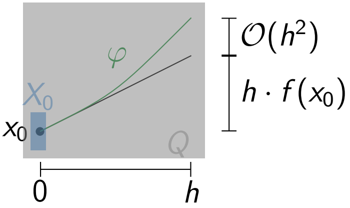
- Euler step:
  f ... slope given by ODE
  $\varphi(x_0, h) = x_0 + h \cdot f(x_0) + O(h^2)$
- set-based Euler step
  - $x_0 \in X_0$
  - f enclosed by F, i.e. $f(X) \subseteq F(X)$

Euler Method



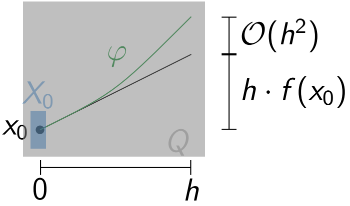
- Euler step:
  f ... slope given by ODE
  $\varphi(x_0, h) = x_0 + h \cdot f(x_0) + O(h^2)$
- set-based Euler step
  - $x_0 \in X_0$
  - f enclosed by F, i.e. $f(X) \subseteq F(X)$
  - Df enclosed by DF

Euler Method



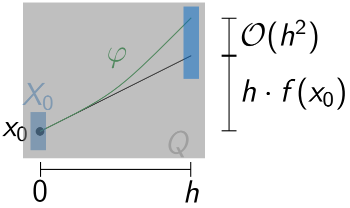
- Euler step:
  f ... slope given by ODE
  $\varphi(x_0, h) = x_0 + h \cdot f(x_0) + O(h^2)$
- set-based Euler step
  - $x_0 \in X_0$
  - f enclosed by F, i.e. $f(X) \subseteq F(X)$
  - $\mathrm{D}f$ enclosed by $\mathrm{D}F$
  - $\varphi(X_0, [0; h]) \subseteq Q$

Euler Method



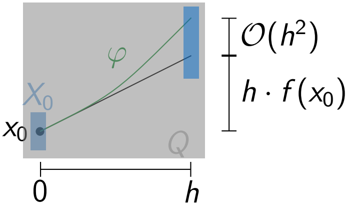
- Euler step:
  f ...slope given by ODE
  $\varphi(x_0, h) = x_0 + h \cdot f(x_0) + O(h^2)$
- set-based Euler step
  - $x_0 \in X_0$
  - f enclosed by F, i.e. $f(X) \subseteq F(X)$
  - $\mathrm{D}f$ enclosed by $\mathrm{D}F$
  - $\varphi(X_0, [0; h]) \subseteq Q = \textit{certify-stepsize}(X_0)$

Euler Method



- Euler step:
  f ...slope given by ODE
  $\varphi(x_0, h) = x_0 + h \cdot f(x_0) + O(h^2)$
- set-based Euler step
  - $x_0 \in X_0$
  - f enclosed by F, i.e. $f(X) \subseteq F(X)$
  - $\mathrm{D}f$ enclosed by $\mathrm{D}F$
  - $\varphi(X_0, [0; h]) \subseteq Q = \mathit{certify\text{-}stepsize}(X_0)$
  - $\mathrm{Euler}_h(X_0) =$
    $X_0 + h \cdot F(X_0) + \frac{1}{2}h^2 \cdot \mathit{box}\,(\mathrm{D}F(Q)(F(Q)))$

Euler Method

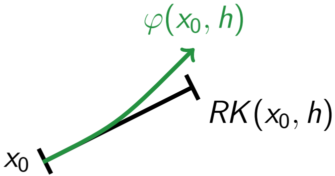


- Euler step:
  f ...slope given by ODE
  $\varphi(x_0, h) = x_0 + h \cdot f(x_0) + O(h^2)$
- set-based Euler step
  - $x_0 \in X_0$
  - f enclosed by F, i.e. $f(X) \subseteq F(X)$
  - $\mathrm{D}f$ enclosed by $\mathrm{D}F$
  - $\varphi(X_0, [0; h]) \subseteq Q = \mathit{certify\text{-}stepsize}(X_0)$
  - $\mathrm{Euler}_h(X_0) =$
    $X_0 + h \cdot F(X_0) + \frac{1}{2}h^2 \cdot \mathit{box}\,(\mathrm{D}F(Q)(F(Q)))$

Theorem

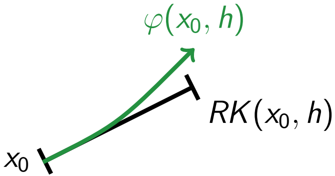
$\varphi(X_0, h) \subseteq \mathrm{Euler}_h(X_0)$

Runge-Kutta methods



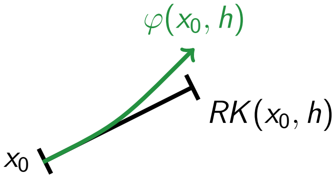
- nested evaluations of f

Runge-Kutta methods



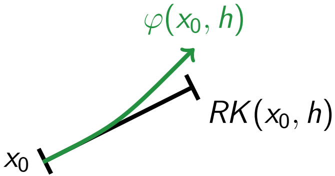
- nested evaluations of f
- higher order approximations

Runge-Kutta methods



- nested evaluations of f
- higher order approximations
- e.g., method of Heun

Runge-Kutta methods



- nested evaluations of f
- higher order approximations
- e.g., method of Heun
- $\varphi(x, h) = x + h \cdot (\frac{1}{2} f(x) + \frac{1}{2} f(x + h \cdot f(x)) + O(h^3)$

Set-Based Arithmetic

Motivation

- enclose errors (algorithm/finite precision)

Set-Based Arithmetic

Motivation

- enclose errors (algorithm/finite precision)

e.g., intervals / interval arithmetic

Set-Based Arithmetic

Motivation

- enclose errors (algorithm/finite precision)

e.g., intervals / interval arithmetic

Problems

Set-Based Arithmetic

Motivation

- enclose errors (algorithm/finite precision)

e.g., intervals / interval arithmetic

Problems

- dependency problem:

Set-Based Arithmetic

Motivation

- enclose errors (algorithm/finite precision)

e.g., intervals / interval arithmetic

Problems

- dependency problem:
  $x \in [0;1] \Longrightarrow x - x \in [0;1] - [0;1] = [-1;1]$

Set-Based Arithmetic

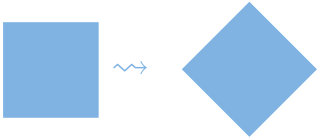
Motivation

- enclose errors (algorithm/finite precision)

e.g., intervals / interval arithmetic

Problems

- dependency problem:
  $x \in [0; 1] \Longrightarrow x - x \in [0; 1] - [0; 1] = [-1; 1]$
- wrapping effect:



Set-Based Arithmetic

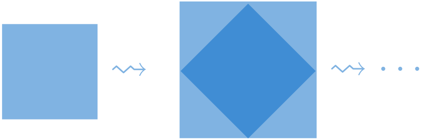
Motivation

- enclose errors (algorithm/finite precision)

e.g., intervals / interval arithmetic

Problems

- dependency problem:
  $x \in [0;1] \Longrightarrow x - x \in [0;1] - [0;1] = [-1;1]$
- wrapping effect:



Affine Arithmetic

Affine Form

$\langle a_0, \ldots, a_k \rangle = a_0 + \sum_{i=1}^{k} \varepsilon_i \cdot a_i$

Affine Arithmetic

Affine Form

$\langle a_0, \ldots, a_k \rangle = a_0 + \sum_{i=1}^{k} \varepsilon_i \cdot a_i$

Linear Operations

$A : \mathbb{R}^n \to \mathbb{R}^m$

- $A\langle a_0, \ldots, a_k \rangle = \langle Aa_0, \ldots, Aa_k \rangle$

Affine Arithmetic

Affine Form

$\langle a_0, \ldots, a_k \rangle = a_0 + \sum_{i=1}^{k} \varepsilon_i \cdot a_i$

Linear Operations

$A : \mathbb{R}^n \to \mathbb{R}^m$

- $A\langle a_0, \ldots, a_k \rangle = \langle Aa_0, \ldots, Aa_k \rangle$

Nonlinear Operations (e.g., $*$, $/$)

approximation with quadratic error

Affine Arithmetic

Affine Form

$\langle a_0, \ldots, a_k \rangle = a_0 + \sum_{i=1}^{k} \varepsilon_i \cdot a_i$

Linear Operations

$A : \mathbb{R}^n \to \mathbb{R}^m$

- $A\langle a_0, \ldots, a_k \rangle = \langle Aa_0, \ldots, Aa_k \rangle$

Nonlinear Operations (e.g., $*$, $/$)

approximation with quadratic error

Explicit Round-Off Operation

round every generator
collect errors in fresh noise symbols ε_i

Refinement

- abstract specification of ODEs/algorithms

Refinement

- abstract specification of ODEs/algorithms
- refinement towards executable structures

Refinement

- abstract specification of ODEs/algorithms
- refinement towards executable structures
  - real numbers $\rightsquigarrow$
    software floating point numbers:

Refinement

- abstract specification of ODEs/algorithms
- refinement towards executable structures
  - real numbers $\rightsquigarrow$
    software floating point numbers:
    - $\alpha(\mathit{Float}(m, e)) = m \cdot 2^e$ for $m, e \in \mathbb{Z}$

Refinement

- abstract specification of ODEs/algorithms
- refinement towards executable structures
  - real numbers $\rightsquigarrow$
    software floating point numbers:
    - $\alpha(Float(m, e)) = m \cdot 2^e$ for $m, e \in \mathbb{Z}$
    - $\alpha(Float(m_1, e_1)) \cdot \alpha(Float(m_2, e_2)) =$
      $\alpha(Float(m_1 \cdot m_2, e_1 + e_2)$

Refinement

- abstract specification of ODEs/algorithms
- refinement towards executable structures
  - real numbers $\rightsquigarrow$ software floating point numbers:
    - $\alpha(\mathit{Float}(m, e)) = m \cdot 2^e$ for $m, e \in \mathbb{Z}$
    - $\alpha(\mathit{Float}(m_1, e_1)) \cdot \alpha(\mathit{Float}(m_2, e_2)) = \alpha(\mathit{Float}(m_1 \cdot m_2, e_1 + e_2)$
  - set $\rightsquigarrow$ list

Refinement

- abstract specification of ODEs/algorithms
- refinement towards executable structures
  - real numbers $\rightsquigarrow$
    software floating point numbers:
    - $\alpha(Float(m, e)) = m \cdot 2^e$ for $m, e \in \mathbb{Z}$
    - $\alpha(Float(m_1, e_1)) \cdot \alpha(Float(m_2, e_2)) =$
      $\alpha(Float(m_1 \cdot m_2, e_1 + e_2)$
  - set $\rightsquigarrow$ list
  - set of $\mathbb{R}^n \rightsquigarrow$ list of affine forms

Refinement

- abstract specification of ODEs/algorithms
- refinement towards executable structures
  - real numbers $\rightsquigarrow$
    software floating point numbers:
    - $\alpha(\mathit{Float}(m, e)) = m \cdot 2^e$ for $m, e \in \mathbb{Z}$
    - $\alpha(\mathit{Float}(m_1, e_1)) \cdot \alpha(\mathit{Float}(m_2, e_2)) =$
      $\alpha(\mathit{Float}(m_1 \cdot m_2, e_1 + e_2)$
  - set $\rightsquigarrow$ list
  - set of $\mathbb{R}^n \rightsquigarrow$ list of affine forms
  - enclosure of solution $\rightsquigarrow$ Euler/RK2

Refinement

- abstract specification of ODEs/algorithms
- refinement towards executable structures
  - real numbers $\rightsquigarrow$
    software floating point numbers:
    - $\alpha(\mathit{Float}(m, e)) = m \cdot 2^e$ for $m, e \in \mathbb{Z}$
    - $\alpha(\mathit{Float}(m_1, e_1)) \cdot \alpha(\mathit{Float}(m_2, e_2)) =$
      $\alpha(\mathit{Float}(m_1 \cdot m_2, e_1 + e_2)$
  - set $\rightsquigarrow$ list
  - set of $\mathbb{R}^n \rightsquigarrow$ list of affine forms
  - enclosure of solution $\rightsquigarrow$ Euler/RK2
- code generation (Standard ML)

Refinement

- abstract specification of ODEs/algorithms
- refinement towards executable structures
  - real numbers $\rightsquigarrow$ software floating point numbers:
    - $\alpha(\mathit{Float}(m, e)) = m \cdot 2^e$ for $m, e \in \mathbb{Z}$
    - $\alpha(\mathit{Float}(m_1, e_1)) \cdot \alpha(\mathit{Float}(m_2, e_2)) = \alpha(\mathit{Float}(m_1 \cdot m_2, e_1 + e_2)$
  - set $\rightsquigarrow$ list
  - set of $\mathbb{R}^n \rightsquigarrow$ list of affine forms
  - enclosure of solution $\rightsquigarrow$ Euler/RK2
- code generation (Standard ML)
- in principle generic!

Intermediate Summary: Formalization and Verification

- flow of ODE: $\varphi :: \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$

Intermediate Summary: Formalization and Verification

- flow of ODE: $\varphi :: \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$
- set based Runge-Kutta methods
  [Bouissou et al. 2013]

Intermediate Summary: Formalization and Verification

- flow of ODE: $\varphi :: \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$
- set based Runge-Kutta methods
  [Bouissou et al. 2013]
  - $RK :: \mathcal{P}(\mathbb{R}^n) \times \mathbb{R} \to \mathcal{P}(\mathbb{R}^n)$

Intermediate Summary: Formalization and Verification

- flow of ODE: $\varphi :: \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$
- set based Runge-Kutta methods
  [Bouissou et al. 2013]
  - $RK :: \mathcal{P}(\mathbb{R}^n) \times \mathbb{R} \to \mathcal{P}(\mathbb{R}^n)$
  - $RK(X, t) \supseteq \varphi(X, t)$

Intermediate Summary: Formalization and Verification

- flow of ODE: $\varphi :: \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$
- set based Runge-Kutta methods
  [Bouissou et al. 2013]
  - $RK :: \mathcal{P}(\mathbb{R}^n) \times \mathbb{R} \to \mathcal{P}(\mathbb{R}^n)$
  - $RK(X, t) \supseteq \varphi(X, t)$
- refinement

Intermediate Summary: Formalization and Verification

- flow of ODE: $\varphi :: \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$
- set based Runge-Kutta methods
  [Bouissou et al. 2013]
  - $RK :: \mathcal{P}(\mathbb{R}^n) \times \mathbb{R} \to \mathcal{P}(\mathbb{R}^n)$
  - $RK(X,t) \supseteq \varphi(X,t)$
- refinement
  - executable $\widetilde{RK}$

Intermediate Summary: Formalization and Verification

- flow of ODE: $\varphi :: \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$
- set based Runge-Kutta methods
  [Bouissou et al. 2013]
  - $RK :: \mathcal{P}(\mathbb{R}^n) \times \mathbb{R} \to \mathcal{P}(\mathbb{R}^n)$
  - $RK(X, t) \supseteq \varphi(X, t)$
- refinement
  - executable $\widetilde{RK}$
  - $\alpha(\widetilde{RK}(\tilde{X}, \tilde{t})) = RK(\alpha(\tilde{X}), \alpha(\tilde{t}))$

Intermediate Summary: Formalization and Verification

- flow of ODE: $\varphi :: \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$
- set based Runge-Kutta methods [Bouissou et al. 2013]
  - $RK :: \mathcal{P}(\mathbb{R}^n) \times \mathbb{R} \to \mathcal{P}(\mathbb{R}^n)$
  - $RK(X, t) \supseteq \varphi(X, t)$
- refinement
  - executable $\widetilde{RK}$
  - $\alpha(\widetilde{RK}(\tilde{X}, \tilde{t})) = RK(\alpha(\tilde{X}), \alpha(\tilde{t}))$

Up Next

applications/optimizations

Oil Reservoir Problem

Oil reservoir problem

$y'(t) = z(t)$

$z'(t) = z(t)^2 - \frac{1}{10^{-3}+y(t)^2}$

$y(0) = 10; z(0) = 0$

- stiff: small step sizes required
- performance: $\approx$ 20 times slower than [Bouissou et al., 2013]
- nontrivial: VNODE fails to maintain precision

Oil Reservoir Problem

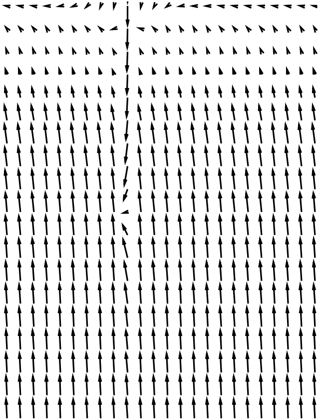
Oil reservoir problem

$y'(t) = z(t)$

$z'(t) = z(t)^2 - \frac{1}{10^{-3}+y(t)^2}$

$y(0) = 10; z(0) = 0$

- stiff: small step sizes required
- performance: $\approx$ 20 times slower than [Bouissou et al., 2013]
- nontrivial: VNODE fails to maintain precision



Oil Reservoir Problem

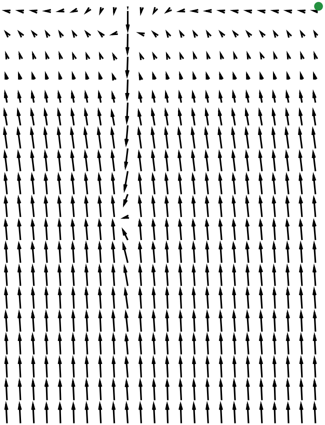
Oil reservoir problem

$y'(t) = z(t)$

$z'(t) = z(t)^2 - \frac{1}{10^{-3}+y(t)^2}$

$y(0) = 10; z(0) = 0$

- stiff: small step sizes required
- performance: $\approx$ 20 times slower than [Bouissou et al., 2013]
- nontrivial: VNODE fails to maintain precision



Oil Reservoir Problem

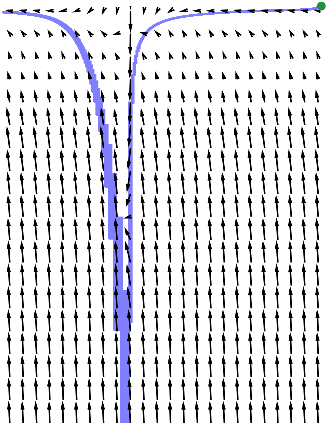
Oil reservoir problem

$y'(t) = z(t)$

$z'(t) = z(t)^2 - \frac{1}{10^{-3}+y(t)^2}$

$y(0) = 10; z(0) = 0$

- stiff: small step sizes required
- performance: $\approx$ 20 times slower than [Bouissou et al., 2013]
- nontrivial: VNODE fails to maintain precision



Content

Problems

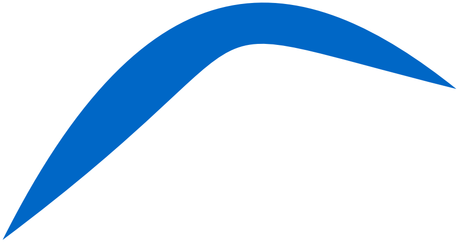
- zonotopes: convex

Problems

- zonotopes: convex
- wrapping non-convex sets

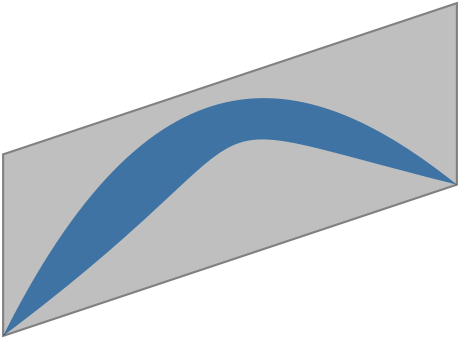
Problems

- zonotopes: convex
- wrapping non-convex sets

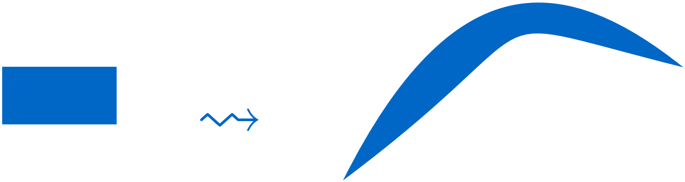


Problems

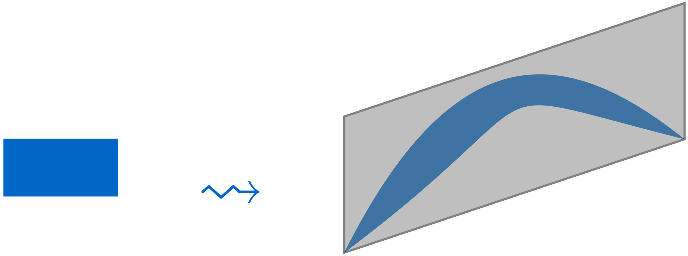
- zonotopes: convex
- wrapping non-convex sets



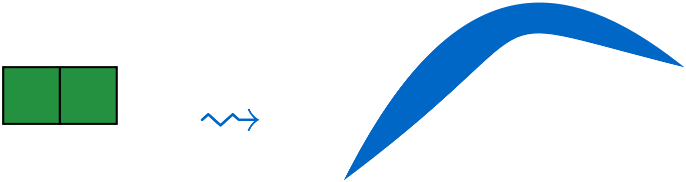
Splitting



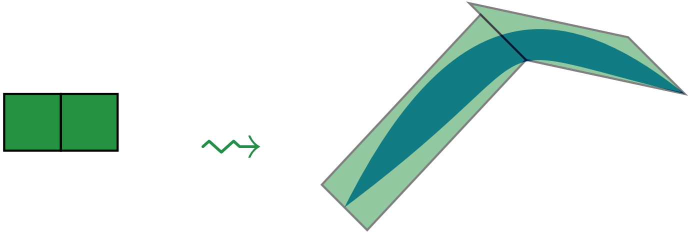
Splitting



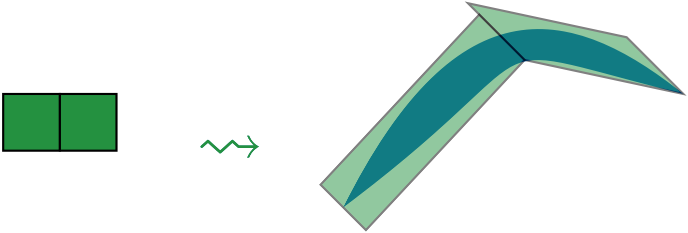
Splitting



Splitting



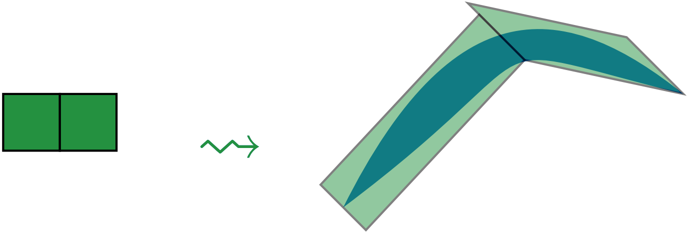
Splitting



Heuristics

- split largest generator of affine form

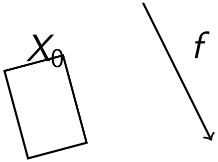
Splitting



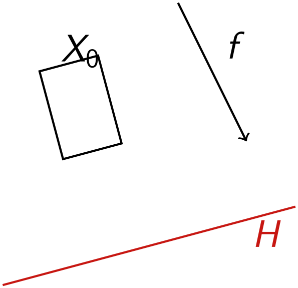
Heuristics

- split largest generator of affine form
- split when diameter exceeds threshold

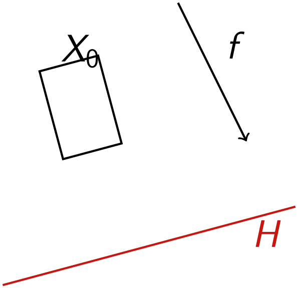
Reduction



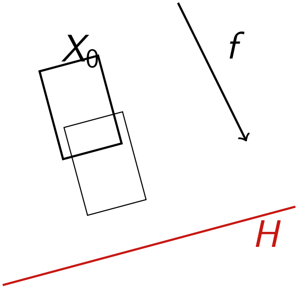
Reduction



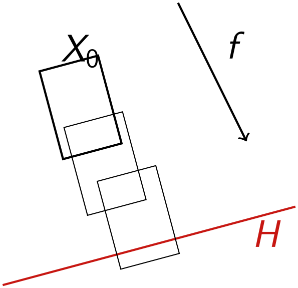
Reduction



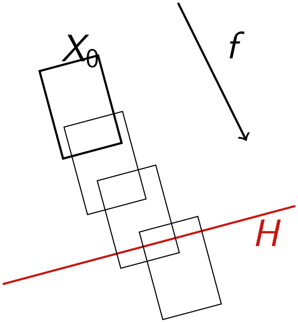
Reduction



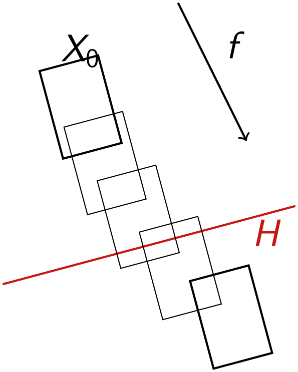
Reduction



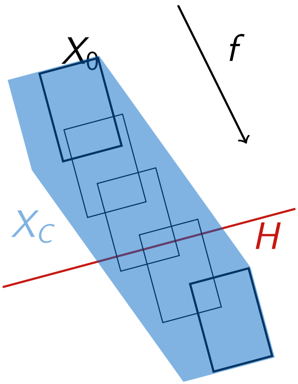
Reduction



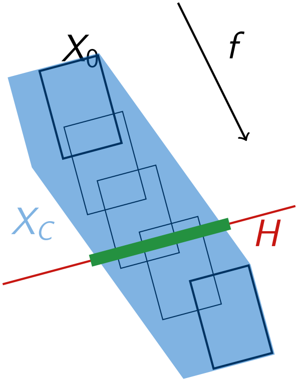
Reduction



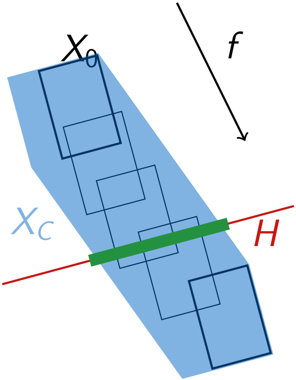
Reduction



Reduction

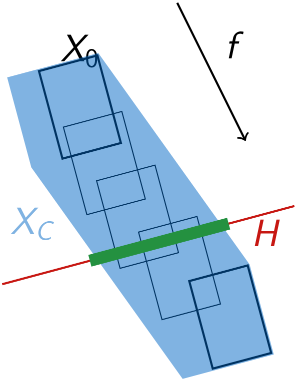


Reduction



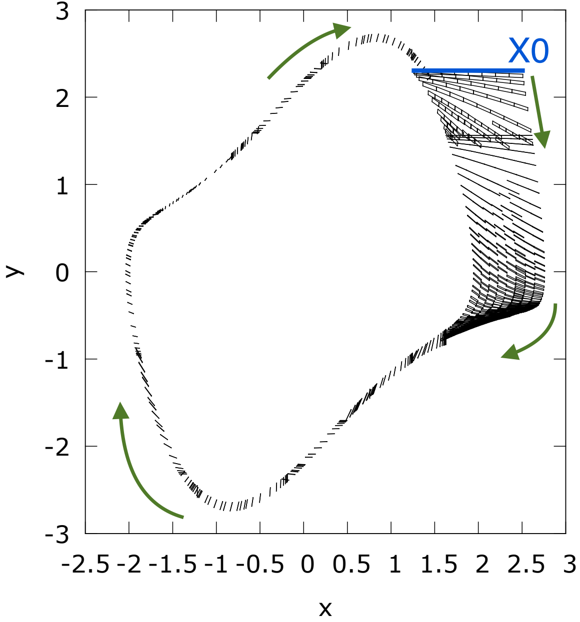
- $X_C \cap H$ can be smaller

Reduction

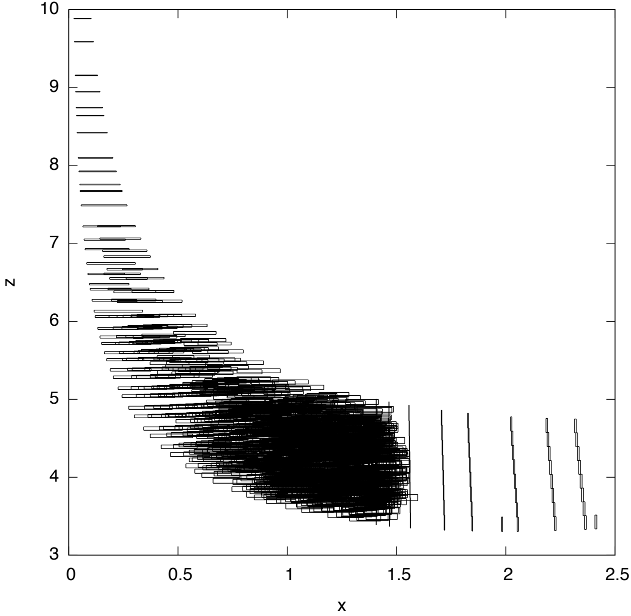


- $X_C \cap H$ can be smaller
- geometric zonotope/hyperplane intersection

Van-der-Pol Oscillator

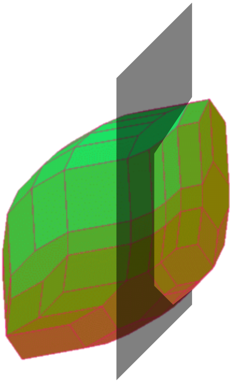


Lorenz attractor (reduction)



Verification of Geometric Algorithms

$\Bigg($



[Boldo, MAP 2016]

Enclosures in Affine Arithmetic

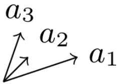
Zonotope

$$\gamma\langle a_0, \ldots, a_k\rangle = \left\{ a_0 + \sum_{i=1}^{k} \varepsilon_i \cdot a_i \middle| \varepsilon_i \in [-1; 1], a_i \in \mathbb{R}^n \right\}$$

Enclosures in Affine Arithmetic

Zonotope

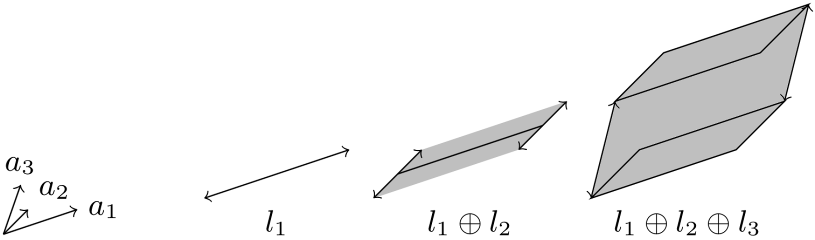
$$\gamma\langle a_0, \ldots, a_k \rangle = \left\{ a_0 + \sum_{i=1}^{k} \varepsilon_i \cdot a_i \middle| \varepsilon_i \in [-1; 1], a_i \in \mathbb{R}^n \right\}$$



Enclosures in Affine Arithmetic

Zonotope

$$\gamma\langle a_0, \ldots, a_k\rangle = \left\{ a_0 + \sum_{i=1}^{k} \varepsilon_i \cdot a_i \middle| \varepsilon_i \in [-1; 1], a_i \in \mathbb{R}^n \right\}$$



Zonotope ∩ Hyperplane

- approximate geometric algorithm [Girard/Le Guernic 2008]
- "proof" not at all formal!
- but similar to convex hull [Knuth: Axioms and Hulls, 1992]

Girard and le Guernic's Algorithm

- reduction to two-dimensional problem

Proposition 1. *Let G be a hyperplane, $G = \{x \in \mathbb{R}^d : x \cdot n = \gamma\}$, Z a set, and ℓ a vector. Let $\Pi_{n,\ell}$ be the following linear transformation:*

$$\begin{aligned} \Pi_{n,\ell} : \mathbb{R}^d &\to \mathbb{R}^2 \\ x &\mapsto (x \cdot n, x \cdot \ell) \end{aligned}$$

Then, we have the following equality

$$\{x \cdot \ell : x \in Z \cap G\} = \{y : (\gamma, y) \in \Pi_{n,\ell}(Z)\}$$

Proof. Let y belongs to $\{x \cdot \ell : x \in Z \cap G\}$, then there exists x in $Z \cap G$ such that $x \cdot \ell = y$. Since $x \in G$, we have $x \cdot n = \gamma$. Therefore $(\gamma, y) = \Pi_{n,\ell}(x) \in \Pi_{n,\ell}(Z)$ because $x \in Z$. Thus, $y \in \{y : (\gamma, y) \in \Pi_{n,\ell}(Z)\}$. Conversely, if $y \in \{y : (\gamma, y) \in \Pi_{n,\ell}(Z)\}$, then $(\gamma, y) \in \Pi_{n,\ell}(Z)$. It follows that there exists $x \in Z$ such that $x \cdot n = \gamma$ and $x \cdot \ell = y$. Since $x \cdot n = \gamma$, it follows that $x \in G$. Thus, $y = x \cdot \ell$ with $x \in Z \cap G$ and it follows that $y \in \{x \cdot \ell : x \in Z \cap G\}$. ∎

Girard and le Guernic's Algorithm

- reduction to two-dimensional problem

```
lemma inter_proj_eq:
  fixes n g l
  defines "G ≡ {x. x • n = g}"
  shows "(λx. x • l) ` (Z ∩ G) =
    {y. (g, y) ∈ (λx. (x • n, x • l)) ` Z}"
  by (auto simp: G_def)
```

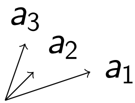
Girard and le Guernic's Algorithm

- reduction to two-dimensional problem
- 2D-zonotope $\cap$ line

```
lemma inter_proj_eq:
  fixes n g l
  defines "G ≡ {x. x • n = g}"
  shows "(λx. x • l) ` (Z ∩ G) =
    {y. (g, y) ∈ (λx. (x • n, x • l)) ` Z}"
  by (auto simp: G_def)
```

Girard and le Guernic's Algorithm

- reduction to two-dimensional problem
- 2D-zonotope $\cap$ line
- compute hull of 2D-zonotope: append sorted line segments

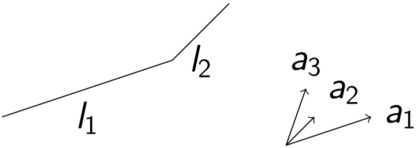


Girard and le Guernic's Algorithm

- reduction to two-dimensional problem
- 2D-zonotope $\cap$ line
- compute hull of 2D-zonotope: append sorted line segments

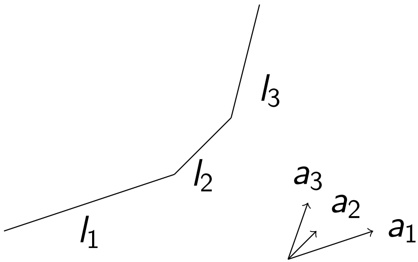
Girard and le Guernic's Algorithm

- reduction to two-dimensional problem
- 2D-zonotope $\cap$ line
- compute hull of 2D-zonotope: append sorted line segments



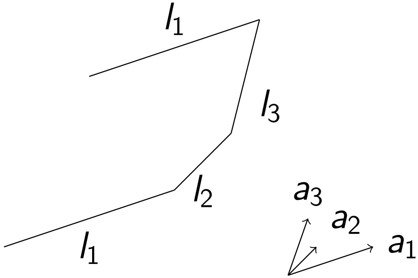
Girard and le Guernic's Algorithm

- reduction to two-dimensional problem
- 2D-zonotope $\cap$ line
- compute hull of 2D-zonotope: append sorted line segments



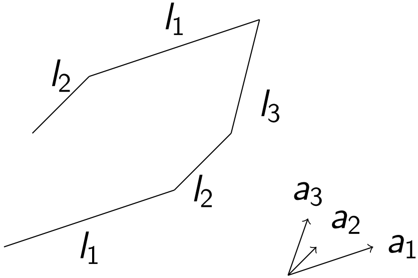
Girard and le Guernic's Algorithm

- reduction to two-dimensional problem
- 2D-zonotope $\cap$ line
- compute hull of 2D-zonotope: append sorted line segments



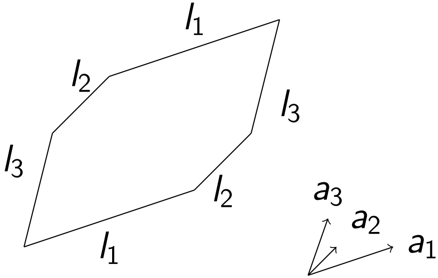
Girard and le Guernic's Algorithm

- reduction to two-dimensional problem
- 2D-zonotope $\cap$ line
- compute hull of 2D-zonotope: append sorted line segments



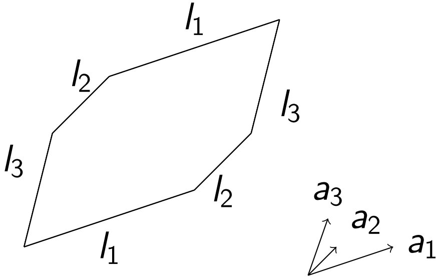
Girard and le Guernic's Algorithm

- reduction to two-dimensional problem
- 2D-zonotope $\cap$ line
- compute hull of 2D-zonotope: append sorted line segments



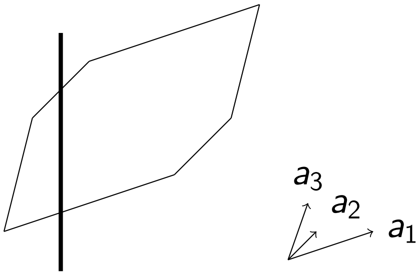
Girard and le Guernic's Algorithm

- reduction to two-dimensional problem
- 2D-zonotope $\cap$ line
- compute hull of 2D-zonotope: append sorted line segments



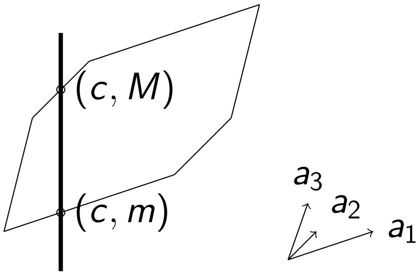
Girard and le Guernic's Algorithm

- reduction to two-dimensional problem
- 2D-zonotope $\cap$ line
- compute hull of 2D-zonotope: append sorted line segments
- intersection: minimum/maximum intersection of segment



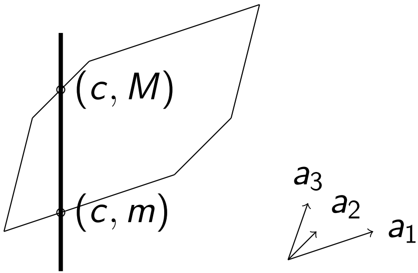
Girard and le Guernic's Algorithm

- reduction to two-dimensional problem
- 2D-zonotope $\cap$ line
- compute hull of 2D-zonotope: append sorted line segments
- intersection: minimum/maximum intersection of segment



Girard and le Guernic's Algorithm

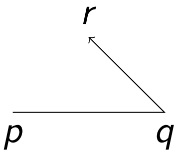
- reduction to two-dimensional problem
- 2D-zonotope $\cap$ line
- compute hull of 2D-zonotope: append sorted line segments
- intersection: minimum/maximum intersection of segment



Knuth's Theory of CCW Systems

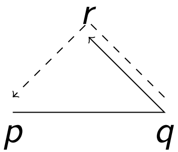
Knuth's Theory of CCW Systems

- cyclic symmetry:
  $pqr \implies qrp$



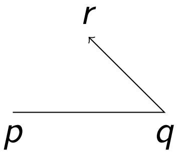
Knuth's Theory of CCW Systems

- cyclic symmetry:
  $pqr \implies qrp$



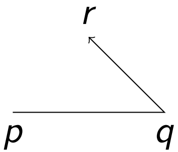
Knuth's Theory of CCW Systems

- cyclic symmetry:
  $pqr \implies qrp$
- antisymmetry:
  $pqr \implies \neg prq$



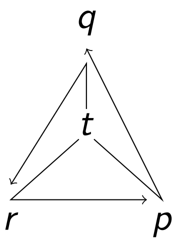
Knuth's Theory of CCW Systems

- cyclic symmetry:
  $pqr \implies qrp$
- antisymmetry:
  $pqr \implies \neg prq$
- nondegeneracy: $pqr \vee prq$



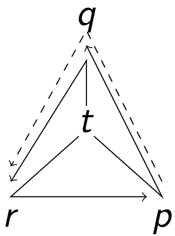
Knuth's Theory of CCW Systems

- cyclic symmetry:
  $pqr \implies qrp$
- antisymmetry:
  $pqr \implies \neg prq$
- nondegeneracy: $pqr \vee prq$
- interiority:
  $tpq \wedge tqr \wedge trp \implies pqr$



Knuth's Theory of CCW Systems

- cyclic symmetry:
  $pqr \implies qrp$
- antisymmetry:
  $pqr \implies \neg prq$
- nondegeneracy: $pqr \vee prq$
- interiority:
  $tpq \wedge tqr \wedge trp \implies pqr$



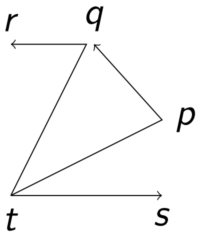
Knuth's Theory of CCW Systems

- cyclic symmetry:
  $pqr \implies qrp$
- antisymmetry:
  $pqr \implies \neg prq$
- nondegeneracy: $pqr \vee prq$
- interiority:
  $tpq \wedge tqr \wedge trp \implies pqr$
- transitivity:
  $tsp \wedge tsq \wedge tsr$

$$t \longrightarrow s$$

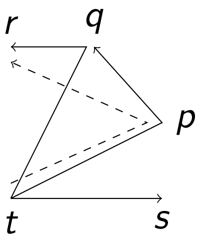
Knuth's Theory of CCW Systems

- cyclic symmetry:
  $pqr \implies qrp$
- antisymmetry:
  $pqr \implies \neg prq$
- nondegeneracy: $pqr \vee prq$
- interiority:
  $tpq \wedge tqr \wedge trp \implies pqr$
- transitivity:
  $tsp \wedge tsq \wedge tsr \wedge tpq \wedge tqr$



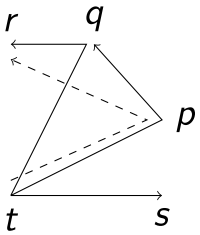
Knuth's Theory of CCW Systems

- cyclic symmetry:
  $pqr \implies qrp$
- antisymmetry:
  $pqr \implies \neg prq$
- nondegeneracy: $pqr \vee prq$
- interiority:
  $tpq \wedge tqr \wedge trp \implies pqr$
- transitivity:
  $tsp \wedge tsq \wedge tsr \wedge tpq \wedge tqr \implies tpr$



Knuth's Theory of CCW Systems

- cyclic symmetry:
  $pqr \implies qrp$
- antisymmetry:
  $pqr \implies \neg prq$
- nondegeneracy: $pqr \vee prq$
- interiority:
  $tpq \wedge tqr \wedge trp \implies pqr$
- transitivity:
  $tsp \wedge tsq \wedge tsr \wedge tpq \wedge tqr \implies tpr$



total order in halfplane left of ts

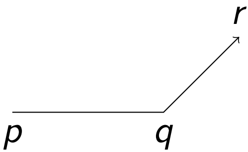
Knuth's Theory of CCW Systems

- cyclic symmetry:
  $pqr \implies qrp$
- antisymmetry:
  $pqr \implies \neg prq$
- nondegeneracy: $pqr \vee prq$
- interiority:
  $tpq \wedge tqr \wedge trp \implies pqr$
- transitivity:
  $tsp \wedge tsq \wedge tsr \wedge tpq \wedge tqr \implies tpr$

total order in halfplane left of ts
sorted$[p, q, r]$

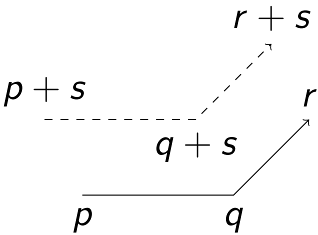
Extensions for Vector Spaces

- translation:
  $(p + s)(q + s)(r + s) \Leftrightarrow pqr$



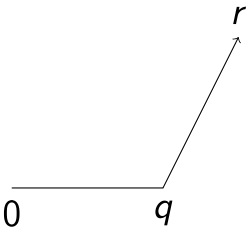
Extensions for Vector Spaces

- translation:
  $(p+s)(q+s)(r+s) \Leftrightarrow pqr$



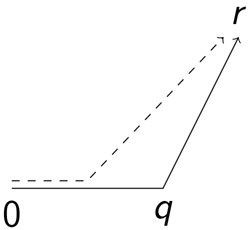
Extensions for Vector Spaces

- translation:
  $(p+s)(q+s)(r+s) \Leftrightarrow pqr$
- scaling:
  $\alpha > 0 \implies 0(\alpha \cdot q)r \Longleftrightarrow 0qr$



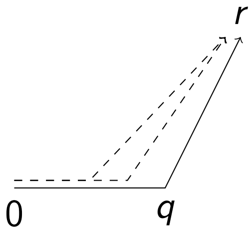
Extensions for Vector Spaces

- translation:
  $(p+s)(q+s)(r+s) \Leftrightarrow pqr$
- scaling:
  $\alpha > 0 \implies 0(\alpha{\cdot}q)r \iff 0qr$



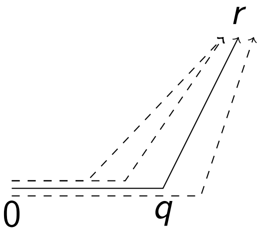
Extensions for Vector Spaces

- translation:
  $(p+s)(q+s)(r+s) \Leftrightarrow pqr$
- scaling:
  $\alpha > 0 \implies 0(\alpha{\cdot}q)r \iff 0qr$



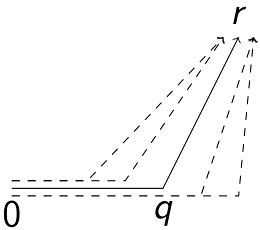
Extensions for Vector Spaces

- translation:
  $(p+s)(q+s)(r+s) \Leftrightarrow pqr$
- scaling:
  $\alpha > 0 \implies 0(\alpha{\cdot}q)r \iff 0qr$



Extensions for Vector Spaces

- translation:
  $(p+s)(q+s)(r+s) \Leftrightarrow pqr$
- scaling:
  $\alpha > 0 \implies 0(\alpha{\cdot}q)r \iff 0qr$



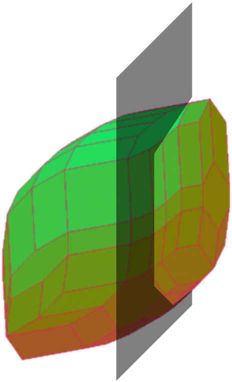
Extensions for Vector Spaces

- translation:
  $(p+s)(q+s)(r+s) \Leftrightarrow pqr$
- scaling:
  $\alpha > 0 \implies 0(\alpha \cdot q)r \iff 0qr$
- reflection: $0(-p)q \iff 0qp$

Extensions for Vector Spaces

- translation:
  $(p+s)(q+s)(r+s) \Leftrightarrow pqr$
- scaling:
  $\alpha > 0 \implies 0(\alpha \cdot q)r \iff 0qr$
- reflection: $0(-p)q \iff 0qp$
- addition
  $0pq \implies 0pr \implies 0p(q+r)$

Close Parenthesis



)

Content

The Lorenz Attractor

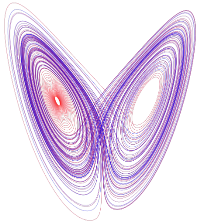
- Lorenz equations (1963)

$$\begin{aligned}\dot{x} &= 10(y - x)\\ \dot{y} &= x(28 - z) - y\\ \dot{z} &= xy - \frac{8}{3}z\end{aligned}$$

Edward N. Lorenz

The Lorenz Attractor

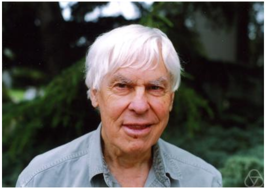
- Lorenz equations (1963)
- numerical simulations



Lorenz attractor

The Lorenz Attractor

- Lorenz equations (1963)
- numerical simulations
- conjecture: chaos
  (strange attractor)



Smale's 14th problem

The Lorenz Attractor

- Lorenz equations (1963)
- numerical simulations
- conjecture: chaos
  (strange attractor)
- proof: Tucker (1999), relying
  on C++-program

Warwick Tucker

The Lorenz Attractor

- Lorenz equations (1963)
- numerical simulations
- conjecture: chaos (strange attractor)
- proof: Tucker (1999), relying on C++-program
- correctness of program?

Sketch of Tucker's Proof

1. attracting set (numerically enclose ODE)

Sketch of Tucker's Proof

1. attracting set (numerically enclose ODE)
2. sensitive dependence on initial conditions (numerically enclose variational equation)

Sketch of Tucker's Proof

1. attracting set (numerically enclose ODE)
2. sensitive dependence on initial conditions (numerically enclose variational equation)
3. analytical reasoning where numerics cannot work

Sketch of Tucker's Proof

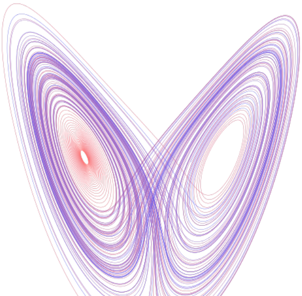
1. attracting set (numerically enclose ODE)
2. sensitive dependence on initial conditions (numerically enclose variational equation)
3. analytical reasoning where numerics cannot work

Contribution

part 1. using *verified* ODE solver

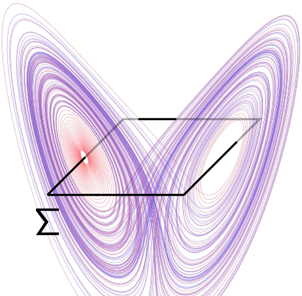
return map

- 3D continuous dynamics



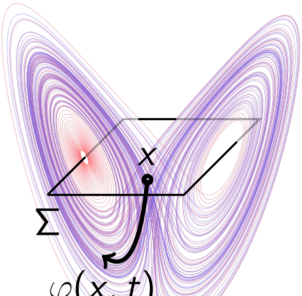
return map

- 3D continuous dynamics
- standard reduction: *return plane* Σ
  iteration of 2D *return map* $R : \Sigma \to \Sigma$



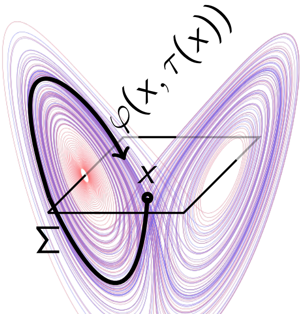
return map

- 3D continuous dynamics
- standard reduction: *return plane* Σ
  iteration of 2D *return map* $R : \Sigma \to \Sigma$
  - $\varphi(x, t)$ ...solution with initial value x after time t



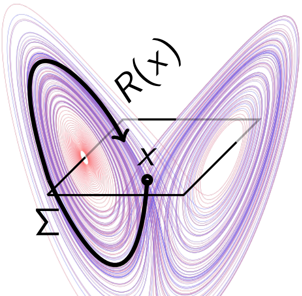
return map

- 3D continuous dynamics
- standard reduction: *return plane* Σ
  iteration of 2D *return map* $R : \Sigma \to \Sigma$
  - $\varphi(x, t)$ ... solution with initial value x after time t
  - $\tau(x)$ ... "first return time"



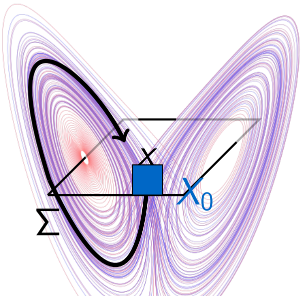
return map

- 3D continuous dynamics
- standard reduction: *return plane* Σ
  iteration of 2D *return map* $R : \Sigma \to \Sigma$
  - $\varphi(x, t)$ ...solution with initial value x after time t
  - $\tau(x)$ ... "first return time"
  - $R(x) := \varphi(x, \tau(x))$ ... "return map"



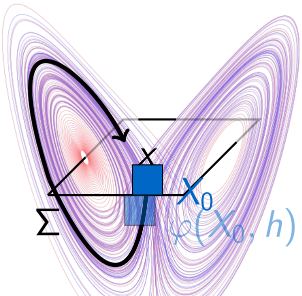
return map

- 3D continuous dynamics
- standard reduction: *return plane* Σ
  iteration of 2D *return map* $R : \Sigma \to \Sigma$
  - $\varphi(x, t)$ ...solution with initial value x after time t
  - $\tau(x)$ ... "first return time"
  - $R(x) := \varphi(x, \tau(x))$ ... "return map"
- Tucker: numerical enclosures for φ, τ



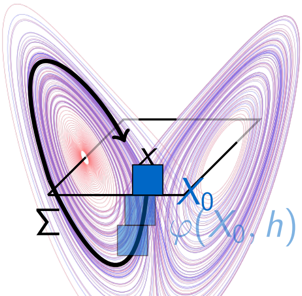
return map

- 3D continuous dynamics
- standard reduction: *return plane* Σ
  iteration of 2D *return map* $R : \Sigma \to \Sigma$
  - $\varphi(x, t)$ ...solution with initial value x after time t
  - $\tau(x)$ ... "first return time"
  - $R(x) := \varphi(x, \tau(x))$ ... "return map"
- Tucker: numerical enclosures for φ, τ



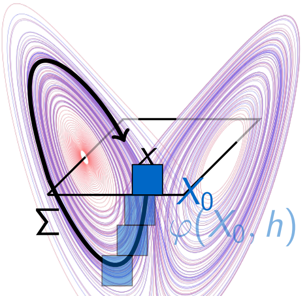
return map

- 3D continuous dynamics
- standard reduction: *return plane* Σ
  iteration of 2D *return map* $R : \Sigma \to \Sigma$
  - $\varphi(x, t)$ ...solution with initial value x after time t
  - $\tau(x)$ ... "first return time"
  - $R(x) := \varphi(x, \tau(x))$ ... "return map"
- Tucker: numerical enclosures for φ, τ



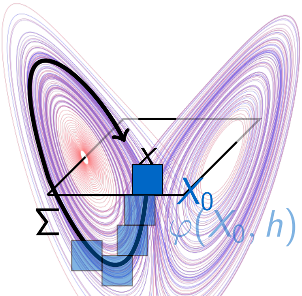
return map

- 3D continuous dynamics
- standard reduction: *return plane* Σ
  iteration of 2D *return map* $R : \Sigma \to \Sigma$
  - $\varphi(x, t)$ ...solution with initial value x after time t
  - $\tau(x)$ ... "first return time"
  - $R(x) := \varphi(x, \tau(x))$ ... "return map"
- Tucker: numerical enclosures for φ, τ



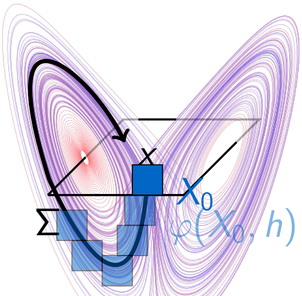
return map

- 3D continuous dynamics
- standard reduction: *return plane* Σ
  iteration of 2D *return map* $R : \Sigma \to \Sigma$
  - $\varphi(x, t)$ ...solution with initial value x after time t
  - $\tau(x)$ ... "first return time"
  - $R(x) := \varphi(x, \tau(x))$ ... "return map"
- Tucker: numerical enclosures for φ, τ



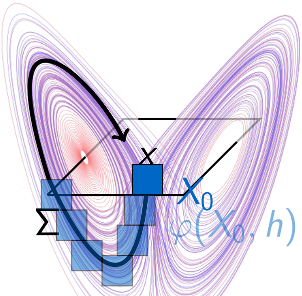
return map

- 3D continuous dynamics
- standard reduction: *return plane* Σ
  iteration of 2D *return map* $R : \Sigma \to \Sigma$
  - $\varphi(x,t)$ ...solution with initial value x after time t
  - $\tau(x)$ ... "first return time"
  - $R(x) := \varphi(x, \tau(x))$ ... "return map"
- Tucker: numerical enclosures for φ, τ



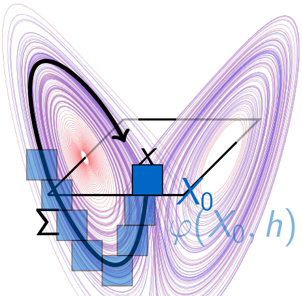
return map

- 3D continuous dynamics
- standard reduction: *return plane* Σ
  iteration of 2D *return map* $R : \Sigma \to \Sigma$
  - $\varphi(x, t)$ ...solution with initial value x after time t
  - $\tau(x)$ ... "first return time"
  - $R(x) := \varphi(x, \tau(x))$ ... "return map"
- Tucker: numerical enclosures for φ, τ



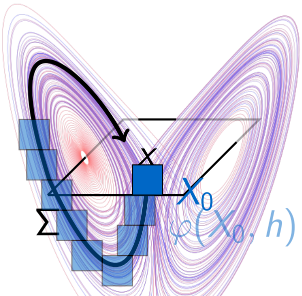
return map

- 3D continuous dynamics
- standard reduction: *return plane* Σ
  iteration of 2D *return map* $R : \Sigma \to \Sigma$
  - $\varphi(x, t)$ …solution with initial value x after time t
  - $\tau(x)$ … "first return time"
  - $R(x) := \varphi(x, \tau(x))$ … "return map"
- Tucker: numerical enclosures for φ, τ



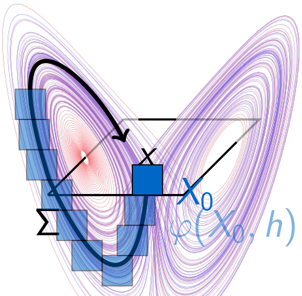
return map

- 3D continuous dynamics
- standard reduction: *return plane* Σ
  iteration of 2D *return map* $R : \Sigma \to \Sigma$
  - $\varphi(x, t)$ ...solution with initial value x after time t
  - $\tau(x)$ ... "first return time"
  - $R(x) := \varphi(x, \tau(x))$ ... "return map"
- Tucker: numerical enclosures for φ, τ



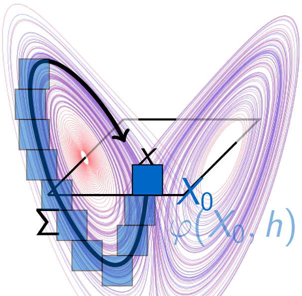
return map

- 3D continuous dynamics
- standard reduction: *return plane* Σ
  iteration of 2D *return map* $R : \Sigma \to \Sigma$
  - $\varphi(x, t)$ . . . solution with initial value x after time t
  - $\tau(x)$ . . . "first return time"
  - $R(x) := \varphi(x, \tau(x))$ . . . "return map"
- Tucker: numerical enclosures for φ, τ



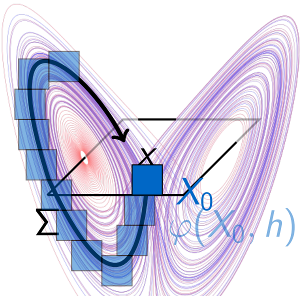
return map

- 3D continuous dynamics
- standard reduction: *return plane* Σ
  iteration of 2D *return map* $R : \Sigma \to \Sigma$
  - $\varphi(x, t)$ ...solution with initial value x after time t
  - $\tau(x)$ ... "first return time"
  - $R(x) := \varphi(x, \tau(x))$ ... "return map"
- Tucker: numerical enclosures for φ, τ



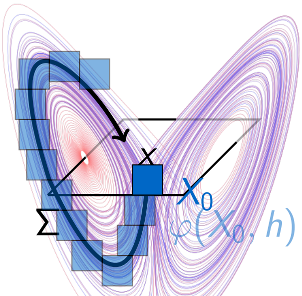
return map

- 3D continuous dynamics
- standard reduction: *return plane* Σ
  iteration of 2D *return map* $R : \Sigma \to \Sigma$
  - $\varphi(x, t)$ ...solution with initial value x after time t
  - $\tau(x)$ ... "first return time"
  - $R(x) := \varphi(x, \tau(x))$ ... "return map"
- Tucker: numerical enclosures for φ, τ



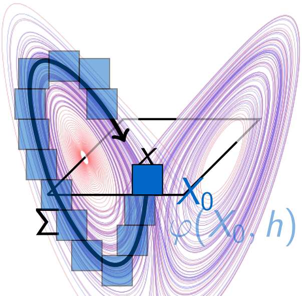
return map

- 3D continuous dynamics
- standard reduction: *return plane* Σ
  iteration of 2D *return map* $R : \Sigma \to \Sigma$
  - $\varphi(x, t)$ ...solution with initial value x after time t
  - $\tau(x)$ ... "first return time"
  - $R(x) := \varphi(x, \tau(x))$ ... "return map"
- Tucker: numerical enclosures for φ, τ



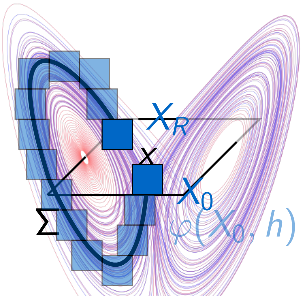
return map

- 3D continuous dynamics
- standard reduction: *return plane* Σ
  iteration of 2D *return map* $R : \Sigma \to \Sigma$
  - $\varphi(x, t)$ ...solution with initial value x after time t
  - $\tau(x)$ ... "first return time"
  - $R(x) := \varphi(x, \tau(x))$ ... "return map"
- Tucker: numerical enclosures for φ, τ



return map

- 3D continuous dynamics
- standard reduction: *return plane* Σ
  iteration of 2D *return map* $R : \Sigma \to \Sigma$
  - $\varphi(x, t)$ …solution with initial value x after time t
  - $\tau(x)$ … "first return time"
  - $R(x) := \varphi(x, \tau(x))$ … "return map"
- Tucker: numerical enclosures for φ, τ



Certify attracting set N

Goal

- given N
- show: $R(N) \subseteq N$

Certify attracting set N

Goal

- given N
- show: $R(N) \subseteq N$

Parallelization

Certify attracting set N

Goal

- given N
- show: $R(N) \subseteq N$

Parallelization

- subdivision: $N = \bigcup_{i=0}^{14000} N_i$

Certify attracting set N

Goal

- given N
- show: $R(N) \subseteq N$

Parallelization

- subdivision: $N = \bigcup_{i=0}^{14000} N_i$
- compute independently for each i: $R(N_i) \subseteq N$

Certify attracting set N

Goal

- given N
- show: $R(N) \subseteq N$

Parallelization

- subdivision: $N = \bigcup_{i=0}^{14000} N_i$
- compute independently for each i: $R(N_i) \subseteq N$
- time: $\approx 1000 * 5\text{h}$ (Tucker: $\approx 2000\text{h}$)

Certify attracting set N

Goal

- given N
- show: $R(N) \subseteq N$

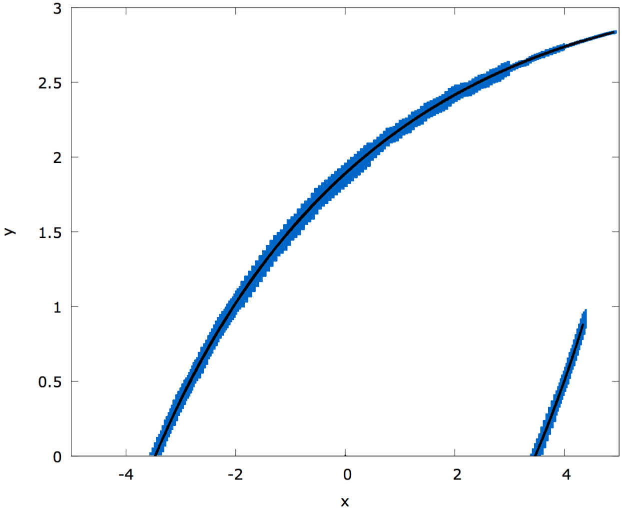
Parallelization

- subdivision: $N = \bigcup_{i=0}^{14000} N_i$
- compute independently for each i: $R(N_i) \subseteq N$
- time: $\approx 1000 * 5\text{h}$ (Tucker: $\approx 2000\text{h}$)

Result

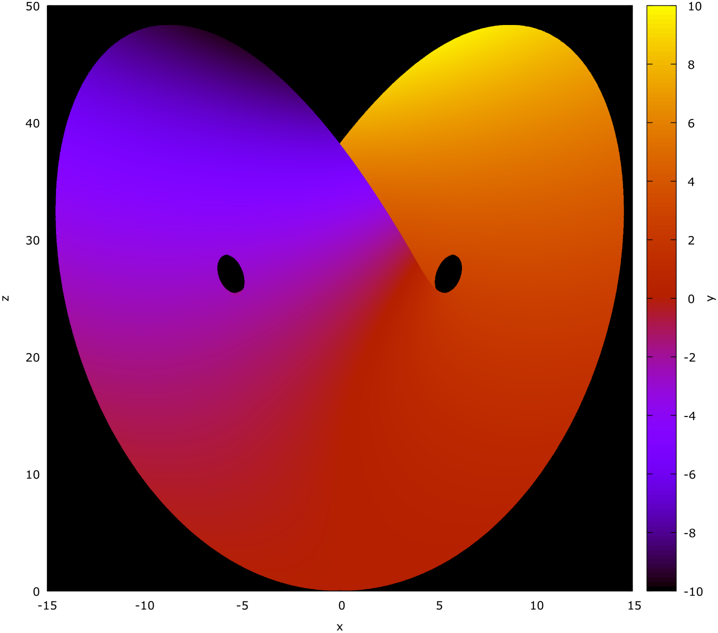
verified a sufficiently precise N

Bound on N

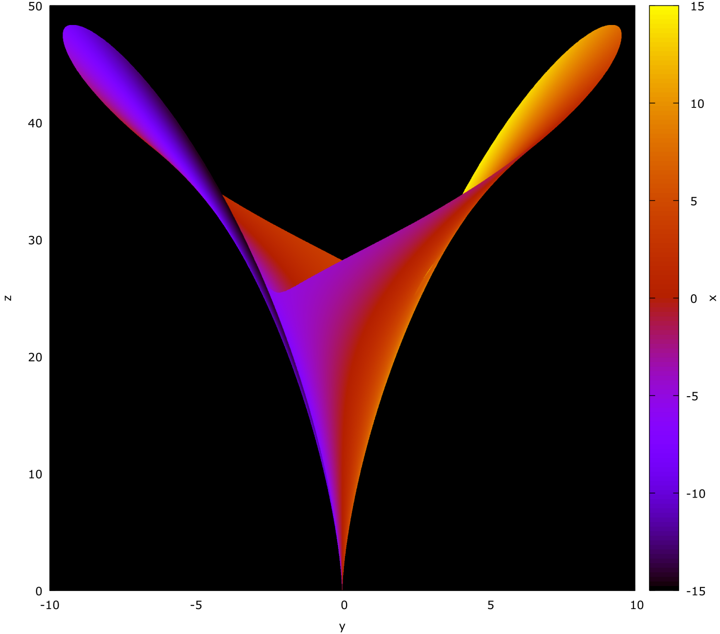


- blue: N_{Tucker}, black: $N_{Isabelle}$

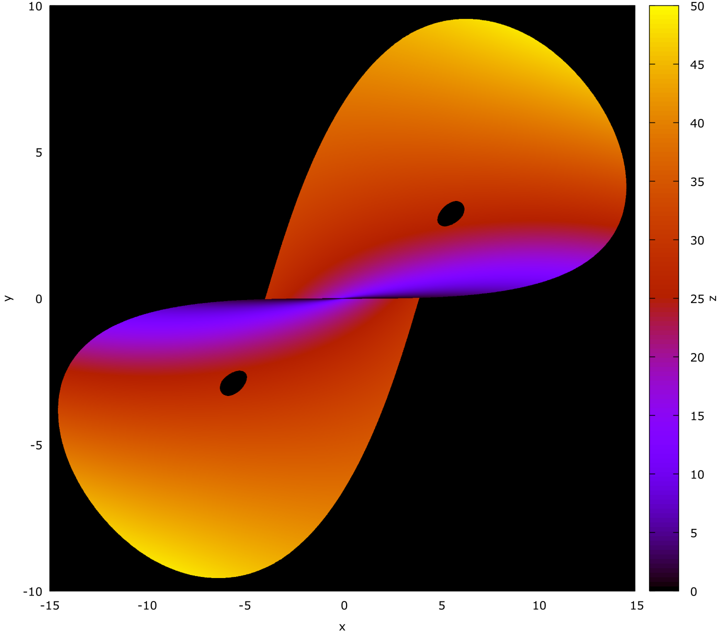
Lorenz Attractor (Front)



Lorenz Attractor (Left)



Lorenz Attractor (Bottom)



Conclusion

verification is feasible and useful:

- verified computation as part of proof

Conclusion

verification is feasible and useful:

- verified computation as part of proof
- novel combination of affine arithmetic/Runge-Kutta methods/reduction

Verified Numerics for ODEs in Isabelle/HOL

Fabian Immler

MAP 2016

Technische Universität München