Rigorous Computation with Function Enclosures in Chebyshev Basis

Tomáš Dzetkulič

Institute of Computer Science
Academy of Sciences of the Czech Republic

6th of June 2012

Rigorous Computation and Initial Value Problem in ODE

Lorentz system:

$\dot{x} = 10(y - x)$
$\dot{y} = x(28 - z) - y$
$\dot{z} = xy - 8z/3$

$x(0) = 15; y(0) = 15; z(0) = 36$

Rigorous Computation and Initial Value Problem in ODE

Lorentz system:

$\dot{x} = 10(y - x)$
$\dot{y} = x(28 - z) - y$
$\dot{z} = xy - 8z/3$

$x(0) = 15; y(0) = 15; z(0) = 36$

Using Matlab `ode45`, we get numerical solution:

	ATol=RTol=10^{-10}	ATol=RTol=10^{-16}
x(50)	6.85806551	4.89309707
y(50)	-1.82131145	7.51188676
z(50)	34.13364729	16.64840204

Rigorous Computation and Initial Value Problem in ODE

Lorentz system:

$\dot{x} = 10(y - x)$
$\dot{y} = x(28 - z) - y$
$\dot{z} = xy - 8z/3$

$x(0) = 15; y(0) = 15; z(0) = 36$

Using Matlab `ode45`, we get numerical solution:

	ATol=RTol=10^{-10}	ATol=RTol=10^{-16}
x(50)	6.85806551	4.89309707
y(50)	-1.82131145	7.51188676
z(50)	34.13364729	16.64840204

Rigorous solution:

$x(50) \in [-0.4737, -0.4738]$
$y(50) \in [-5.13, -5.14]$
$z(50) \in [26.93, 26.94]$

The Overview of the Contribution

We extend the work of Makino and Berz on Taylor Models

We replace the Taylor polynomial approximation in the Taylor Model with the Chebyshev polynomial approximation

The Overview of the Contribution

We extend the work of Makino and Berz on Taylor Models

We replace the Taylor polynomial approximation in the Taylor Model with the Chebyshev polynomial approximation

New rigorous methods for operations with the Chebyshev function enclosure are constructed

The Overview of the Contribution

We extend the work of Makino and Berz on Taylor Models

We replace the Taylor polynomial approximation in the Taylor Model with the Chebyshev polynomial approximation

New rigorous methods for operations with the Chebyshev function enclosure are constructed

Method is applied to the initial value problem of ordinary differential equations

Chebyshev Polynomials

Chebyshev polynomials of the first kind:

$T_0(x) = 1$
$T_1(x) = x$
$T_i(x) = 2x\ T_{i-1}(x) - T_{i-2}(x)$ for $i \in \{2..\infty\}$

$T_2(x) = 2x^2 - 1$; $T_3(x) = 4x^3 - 3x$; $T_4(x) = 8x^4 - 8x^2 + 1$

Chebyshev Polynomials

Chebyshev polynomials of the first kind:

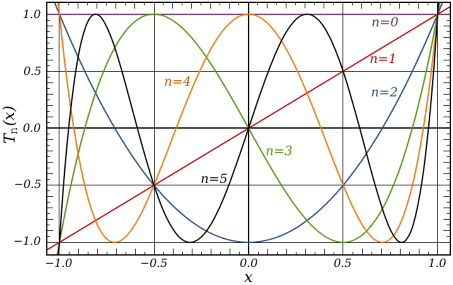
$T_0(x) = 1$

$T_1(x) = x$

$T_i(x) = 2x \ T_{i-1}(x) - T_{i-2}(x)$ for $i \in \{2..\infty\}$

$T_2(x) = 2x^2 - 1$; $T_3(x) = 4x^3 - 3x$; $T_4(x) = 8x^4 - 8x^2 + 1$

Alternative form: $T_i(x) = \cos(i \arccos(x))$



Polynomial Function Enclosure

Taylor Model by Makino and Berz [1]:

For the function $f(x_1, \ldots, x_n)$ on the domain $[-1, 1]^n$:
$f(x_1, \ldots, x_n) \in (\sum_{(i_1,\ldots,i_n)} a_{(i_1,\ldots,i_n)} \prod_j x_j^{i_j}) + [-e_{lo}, e_{hi}]$

Polynomial Function Enclosure

Taylor Model by Makino and Berz [1]:

For the function $f(x_1, \ldots, x_n)$ on the domain $[-1, 1]^n$:
$f(x_1, \ldots, x_n) \in (\sum_{(i_1,\ldots,i_n)} a_{(i_1,\ldots,i_n)} \prod_j x_j^{i_j}) + [-e_{lo}, e_{hi}]$

Example: $f(x, y) = y \sin(x)$ on $[-1, 1]^2$:

$$xy + 0.1666x^3y + [-0.01, 0.01]$$

Polynomial Function Enclosure

Taylor Model by Makino and Berz [1]:

For the function $f(x_1, \ldots, x_n)$ on the domain $[-1, 1]^n$:
$f(x_1, \ldots, x_n) \in (\sum_{(i_1,\ldots,i_n)} a_{(i_1,\ldots,i_n)} \prod_j x_j^{i_j}) + [-e_{lo}, e_{hi}]$

Example: $f(x, y) = y \sin(x)$ on $[-1, 1]^2$:

$$xy + 0.1666x^3y + [-0.01, 0.01]$$

Makino and Berz in [1] claim, that:

- Magnitude of the Chebyshev series coefficients is higher compared to the Taylor series
- Chebyshev polynomial multiplication sub-optimal

$\rightarrow$ Chebyshev polynomials not suitable for rigorous computation

Chebyshev Function Enclosure

We show that:

- The Chebyshev series never lead to an increase in the magnitude of the coefficients
- Although sub-optimal, the Chebyshev polynomial multiplication gives better approximation compared to the Taylor multiplication

Chebyshev Function Enclosure

We show that:

- The Chebyshev series never lead to an increase in the magnitude of the coefficients
- Although sub-optimal, the Chebyshev polynomial multiplication gives better approximation compared to the Taylor multiplication

We introduce the Chebyshev function enclosure of the form:

For the function $f(x_1, \ldots, x_n)$ on the domain $[-1, 1]^n$:
$f(x_1, \ldots, x_n) \in (\sum_{(i_1,\ldots,i_n)} a_{(i_1,\ldots,i_n)} \prod_j T_{i_j}(x_j)) + [-e, e]$

Function Enclosure Operations

The operations with function enclosures:

- Addition and substraction - simple term based algorithm
- Multiplication - new recursive algorithm

Function Enclosure Operations

The operations with function enclosures:

- Addition and substraction - simple term based algorithm
- Multiplication - new recursive algorithm
- Composition - Clenshaw algorithm
- Division, square root, exp - application of composition

Function Enclosure Operations

The operations with function enclosures:

- Addition and substraction - simple term based algorithm
- Multiplication - new recursive algorithm
- Composition - Clenshaw algorithm
- Division, square root, exp - application of composition
- Integration and derivative - reordering of coefficients

Problems with Chebyshev Polynomials Multiplication

$$T_i(x) \times T_j(x) = (T_{i+j}(x) + T_{|i-j|}(x))/2$$

Low order terms of the result depend on high order terms

Problems with Chebyshev Polynomials Multiplication

$$T_i(x) \times T_j(x) = (T_{i+j}(x) + T_{|i-j|}(x))/2$$

Low order terms of the result depend on high order terms

Given the truncated Chebyshev series of $f_1(x)$ and $f_2(x)$:
- impossible to compute the truncated series for $f_1(x) \times f_2(x)$

Problems with Chebyshev Polynomials Multiplication

$$T_i(x) \times T_j(x) = (T_{i+j}(x) + T_{|i-j|}(x))/2$$

Low order terms of the result depend on high order terms

Given the truncated Chebyshev series of $f_1(x)$ and $f_2(x)$:
- impossible to compute the truncated series for $f_1(x) \times f_2(x)$

Multiplication of two n-variate terms gives 2^n term result:

$$(T_1(x)T_1(y)) \times (T_2(x)T_3(y)) =$$
$$(T_1(x)T_2(y) + T_1(x)T_4(y) + T_3(x)T_2(y) + T_3(x)T_4(y))/4$$

Recursive Polynomial Multiplication Algorithm

Task: Compute $F(x_1, \ldots, x_n) \times G(x_1, \ldots, x_n)$ with degree deg polynomials

Recursive Polynomial Multiplication Algorithm

Task: Compute $F(x_1, \ldots, x_n) \times G(x_1, \ldots, x_n)$ with degree deg polynomials

1. Separate variable x_1:
$F(x_1, \ldots, x_n) = \sum_{i=0}^{\deg} F_i(x_2, \ldots, x_n) T_i(x_1)$
$G(x_1, \ldots, x_n) = \sum_{i=0}^{\deg} G_i(x_2, \ldots, x_n) T_i(x_1)$

Recursive Polynomial Multiplication Algorithm

Task: Compute $F(x_1, \ldots, x_n) \times G(x_1, \ldots, x_n)$ with degree deg polynomials

1. Separate variable x_1:
$F(x_1, \ldots, x_n) = \sum_{i=0}^{\deg} F_i(x_2, \ldots, x_n) T_i(x_1)$
$G(x_1, \ldots, x_n) = \sum_{i=0}^{\deg} G_i(x_2, \ldots, x_n) T_i(x_1)$

2. For each pair F_i, G_j recursively compute $P_{(i,j)} = F_i \times G_j$

Recursive Polynomial Multiplication Algorithm

Task: Compute $F(x_1, \ldots, x_n) \times G(x_1, \ldots, x_n)$ with degree deg polynomials

1. Separate variable x_1:
$F(x_1, \ldots, x_n) = \sum_{i=0}^{\deg} F_i(x_2, \ldots, x_n) T_i(x_1)$
$G(x_1, \ldots, x_n) = \sum_{i=0}^{\deg} G_i(x_2, \ldots, x_n) T_i(x_1)$

2. For each pair F_i, G_j recursively compute $P_{(i,j)} = F_i \times G_j$

3. For all i set $R_i := 0$

4. Add $P_{(i,j)}/2$ to R_{i+j} and $R_{|i-j|}$

Recursive Polynomial Multiplication Algorithm

Task: Compute $F(x_1, \ldots, x_n) \times G(x_1, \ldots, x_n)$ with degree deg polynomials

1. Separate variable x_1:
$F(x_1, \ldots, x_n) = \sum_{i=0}^{\deg} F_i(x_2, \ldots, x_n) T_i(x_1)$
$G(x_1, \ldots, x_n) = \sum_{i=0}^{\deg} G_i(x_2, \ldots, x_n) T_i(x_1)$

2. For each pair F_i, G_j recursively compute $P_{(i,j)} = F_i \times G_j$

3. For all i set $R_i := 0$

4. Add $P_{(i,j)}/2$ to R_{i+j} and $R_{|i-j|}$

5. Construct the result: $\sum_{i=0}^{2\deg} R_i(x_2, \ldots, x_n) T_i(x_1)$

Recursive Polynomial Multiplication Algorithm

Task: Compute $F(x_1, \ldots, x_n) \times G(x_1, \ldots, x_n)$ with degree deg polynomials

1. Separate variable x_1:
$F(x_1, \ldots, x_n) = \sum_{i=0}^{\deg} F_i(x_2, \ldots, x_n) T_i(x_1)$
$G(x_1, \ldots, x_n) = \sum_{i=0}^{\deg} G_i(x_2, \ldots, x_n) T_i(x_1)$

2. For each pair F_i, G_j recursively compute $P_{(i,j)} = F_i \times G_j$

3. For all i set $R_i := 0$

4. Add $P_{(i,j)}/2$ to R_{i+j} and $R_{|i-j|}$

5. Construct the result: $\sum_{i=0}^{2\deg} R_i(x_2, \ldots, x_n) T_i(x_1)$

$2\deg^2$ function enclosures $P_{(i,j)}/2$, but only $2\deg$ R_i
→ huge cancellation in step 4 of the algorithm

Initial Value Problem

Given the input:

- system of n differential equations $\dot{\mathbf{x}} = f(\mathbf{x})$
- initial values over m free variables
  $\{\mathbf{x} \mid \exists \mathbf{a} \in [-1, 1]^m : g(\mathbf{a}) = \mathbf{x}\}$
- time bound t_{max}

Initial Value Problem

Given the input:

- system of n differential equations $\dot{\mathbf{x}} = f(\mathbf{x})$
- initial values over m free variables
  $\{\mathbf{x} \mid \exists \mathbf{a} \in [-1, 1]^m : g(\mathbf{a}) = \mathbf{x}\}$
- time bound t_{max}

Compute the function $h(t, \mathbf{a})$:

- it is the solution to the ODE: $dh(t, \mathbf{a})/dt = f(h(t, \mathbf{a}))$
- $h(0, \mathbf{a}) = g(\mathbf{a})$

Initial Value Problem

Given the input:

- system of n differential equations $\dot{\mathbf{x}} = f(\mathbf{x})$
- initial values over m free variables
  $\{\mathbf{x} \mid \exists \mathbf{a} \in [-1, 1]^m : g(\mathbf{a}) = \mathbf{x}\}$
- time bound t_{max}

Compute the function $h(t, \mathbf{a})$:

- it is the solution to the ODE: $dh(t, \mathbf{a})/dt = f(h(t, \mathbf{a}))$
- $h(0, \mathbf{a}) = g(\mathbf{a})$

$f(\mathbf{x})$ and $g(\mathbf{a})$ given as Chebyshev function enclosures

Picard Iteration

We set $h_0(t, \mathbf{a}) := 0$

Application of Picard operator:
Compute a sequence of enclosures for the recurrence
$h_{i+1}(t, \mathbf{a}) := g(\mathbf{a}) + t_{max} \int_0^t f(h_i(t, \mathbf{a}))dt$

Picard Iteration

We set $h_0(t, \mathbf{a}) := 0$

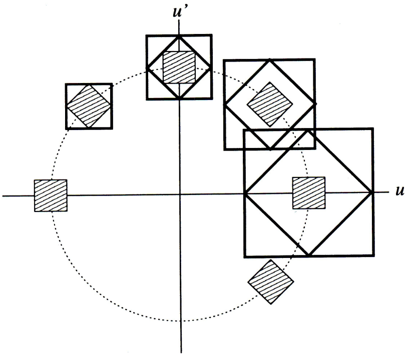
Application of Picard operator:
Compute a sequence of enclosures for the recurrence
$h_{i+1}(t, \mathbf{a}) := g(\mathbf{a}) + t_{max} \int_0^t f(h_i(t, \mathbf{a}))dt$

In case of convergence, the sequence of h_i converges to $h(t, \mathbf{a})$

In case of non-convergent sequence: t_{max} can be reduced and muli-step method can be applied

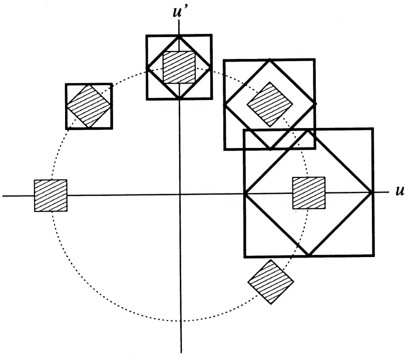
Wrapping Effect

The exponential growth of the error due to re-packaging of the solution in each step



Wrapping Effect

The exponential growth of the error due to re-packaging of the solution in each step



The wrapped object may be of complex non-convex shape

New Method for Wrapping Effect Suppression

In rigorous Picard operator:
$H_{i+1}(t,\mathbf{a}) := G(\mathbf{a}) + t_{max}\int_0^t F(H_i(t,\mathbf{a}))dt$

if $G(\mathbf{a})$ contains non-empty error interval:
$\rightarrow$ same error interval is added in each operator application

New Method for Wrapping Effect Suppression

In rigorous Picard operator:
$H_{i+1}(t, \mathbf{a}) := G(\mathbf{a}) + t_{max} \int_0^t F(H_i(t, \mathbf{a}))dt$

if $G(\mathbf{a})$ contains non-empty error interval:
$\rightarrow$ same error interval is added in each operator application

The dependency problem is introduced

New Method for Wrapping Effect Suppression

In rigorous Picard operator:
$H_{i+1}(t, \mathbf{a}) := G(\mathbf{a}) + t_{max} \int_0^t F(H_i(t, \mathbf{a}))dt$

if $G(\mathbf{a})$ contains non-empty error interval:
$\rightarrow$ same error interval is added in each operator application

The dependency problem is introduced

Solution: Make the error interval disappear

New Method for Wrapping Effect Suppression

In rigorous Picard operator:
$H_{i+1}(t, \mathbf{a}) := G(\mathbf{a}) + t_{max} \int_0^t F(H_i(t, \mathbf{a}))dt$

if $G(\mathbf{a})$ contains non-empty error interval:
$\rightarrow$ same error interval is added in each operator application

The dependency problem is introduced

Solution: Make the error interval disappear

Additional variables $\mathbf{p}$ are used to parametrize the error interval

We construct error-free $G'(\mathbf{a}, \mathbf{p})$ that describes the same initial set

New Method for Wrapping Effect Suppression

In rigorous Picard operator:
$H_{i+1}(t, \mathbf{a}) := G(\mathbf{a}) + t_{max} \int_0^t F(H_i(t, \mathbf{a}))dt$

if $G(\mathbf{a})$ contains non-empty error interval:
$\rightarrow$ same error interval is added in each operator application

The dependency problem is introduced

Solution: Make the error interval disappear

Additional variables $\mathbf{p}$ are used to parametrize the error interval

We construct error-free $G'(\mathbf{a}, \mathbf{p})$ that describes the same initial set

$G'(\mathbf{a}, \mathbf{p})$ is used instead of $G(\mathbf{a})$ to compute $H'(t, \mathbf{a}, \mathbf{p})$

New Method for Wrapping Effect Suppression

In rigorous Picard operator:
$H_{i+1}(t, \mathbf{a}) := G(\mathbf{a}) + t_{max} \int_0^t F(H_i(t, \mathbf{a}))dt$

if $G(\mathbf{a})$ contains non-empty error interval:
$\rightarrow$ same error interval is added in each operator application

The dependency problem is introduced

Solution: Make the error interval disappear

Additional variables $\mathbf{p}$ are used to parametrize the error interval

We construct error-free $G'(\mathbf{a}, \mathbf{p})$ that describes the same initial set

$G'(\mathbf{a}, \mathbf{p})$ is used instead of $G(\mathbf{a})$ to compute $H'(t, \mathbf{a}, \mathbf{p})$

Similar idea used for multi-step wrapping effect suppression

Implementation and Extensions

Rigorous IVP solver implemented in C++ with both Taylor and Chebyshev function enclosures

Implementation and Extensions

Rigorous IVP solver implemented in C++ with both Taylor and Chebyshev function enclosures

Multi-precision extension:

- `double` polynomial coefficients can be replaced with more precise data type
- High precision results can be used to verify numerical results and low-precision results

Implementation and Extensions

Rigorous IVP solver implemented in C++ with both Taylor and Chebyshev function enclosures

Multi-precision extension:

- `double` polynomial coefficients can be replaced with more precise data type
- High precision results can be used to verify numerical results and low-precision results

Multi-processor support:

- All data structures can be used in parallel execution
- Solving high dimension problems is executed in parallel

Computational Experiments

Comparison of Taylor and Chebyshev polynomial enclosures:

Problem (degree)	Taylor	Chebyshev
Volterra (10)	1.1E-6	5.7E-9
Volterra (12)	3.4E-8	5.2E-11
Volterra (14)	1.1E-9	9.8E-13
Roessler (12)	1.8E-6	1.4E-8
Roessler (14)	1.2E-7	2.7E-10
Roessler (16)	9E-9	5.7E-12
Roessler (18)	6.6E-10	5.3E-13

VERICOMP Computation Experiments

VERICOMP - A System for Comparing Verified IVP Solvers
`http://vericomp.inf.uni-due.de/`

		Best result in VERICOMP				Our tool	
#	N	VNODE_LP		RIOT		Width	Time
1	2	4.67079	0.01s	10.1	2s	4.67078	0.08s
2	3	0.232544	0.01s	0.235	0.7s	0.232544	0.03s
3	1	0.89	0.01s	0.44	40s	0.38	0.12s
4	2	0.073	0.02s	0.067569	38s	0.067561	0.4s
5	51	0.21527	2s	N/A		0.21527	18s
6	30	2.95E-5	3s	N/A		2.54E-5	160s
28	2	N/A		N/A		1.018	6s

In benchmarks 1 to 5, the results from our tool match optimal interval width in all displayed digits.

Conclusion

Method for rigorous computation with multivariate function enclosures in Chebyshev basis

Conclusion

Method for rigorous computation with multivariate function enclosures in Chebyshev basis

Provides verified enclosure for various polynomial operations

Conclusion

Method for rigorous computation with multivariate function enclosures in Chebyshev basis

Provides verified enclosure for various polynomial operations

Used in tool for rigorous solution of initial value problem for ODE

Conclusion

Method for rigorous computation with multivariate function enclosures in Chebyshev basis

Provides verified enclosure for various polynomial operations

Used in tool for rigorous solution of initial value problem for ODE

New wrapping effect suppression method proposed

Conclusion

Method for rigorous computation with multivariate function enclosures in Chebyshev basis

Provides verified enclosure for various polynomial operations

Used in tool for rigorous solution of initial value problem for ODE

New wrapping effect suppression method proposed

Implementation available (open source) from:
`http://odeintegrator.souceforge.net`

Ongoing, Future Work

Planned tool improvements:

- Implementation of time step control
- Allow trigonometric functions on input
- Improved handling of equations with many variables

Ongoing, Future Work

Planned tool improvements:

- Implementation of time step control
- Allow trigonometric functions on input
- Improved handling of equations with many variables

Use the method for reachability analysis in hybrid system verification

Ongoing, Future Work

Planned tool improvements:

- Implementation of time step control
- Allow trigonometric functions on input
- Improved handling of equations with many variables

Use the method for reachability analysis in hybrid system verification

Thank you for you attention.

[1] K. Makino and M. Berz. Taylor models and other validated functional inclusion methods. *International Journal of Pure and Applied Mathematics*, 4(4):379–456, 2003.