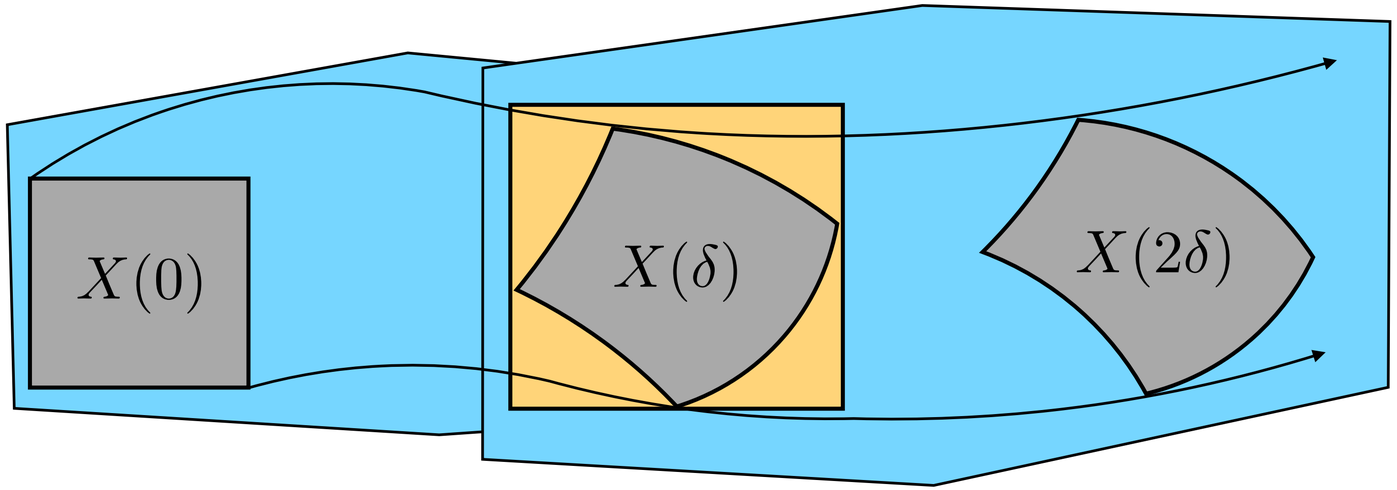
Functional Overapproximation for System Dynamics using ***Taylor Models***

Xin Chen
Assistant Professor, University of Dayton, USA.

The 18th International Summer School on Trustworthy Software. 2022.

Overestimation in Set Propagation



- Reachable set segments (flowpipes) are computed iteratively.
- The overestimation in a flowpipe is propagated to the next one.
- Overapproximations are often represented by convex sets, such as intervals, polytopes or zonotopes, although the actually reachable sets are nonconvex.
- It is hard to compute and represent nonconvex overapproximations.

Questions

- Is it possible to direct compute nonconvex overapproximation sets?
- If so, how can we represent the overapproximation sets?
- How easy can they be computed?
- How accurate can they be?
- How easy can we handle them in other computation and analysis tasks?

Flowmap of an ODE

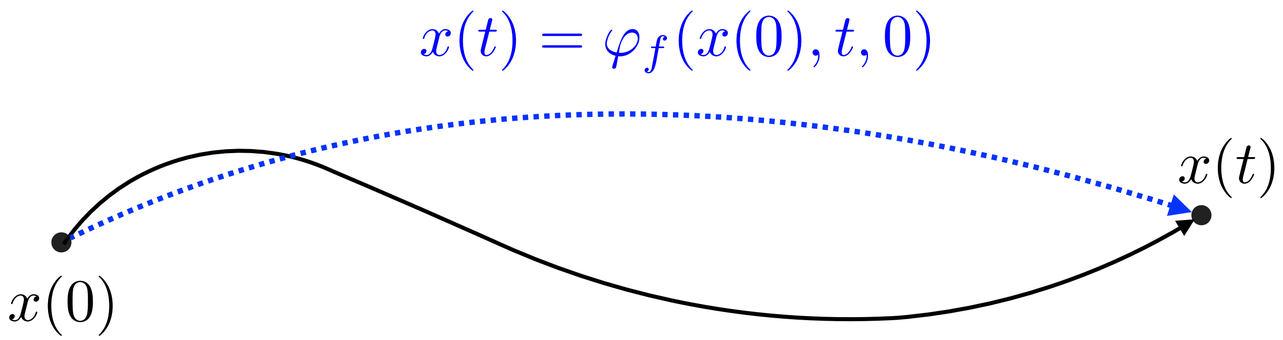
General solution form of the ODE $\dot{x} = f(x, t)$ regarding to the initial condition $x(t_0) = x_0$:

$$\varphi_f(x_0, t, t_0) = x_0 + \int_{t_0}^{t} f(\varphi_f(x_0, \tau, t_0), \tau) d\tau$$

The function φ_f defines a mapping from an initial configuration to the reachable state at a time, i.e., $x(t) = \varphi_f(x_0, t, t_0)$.

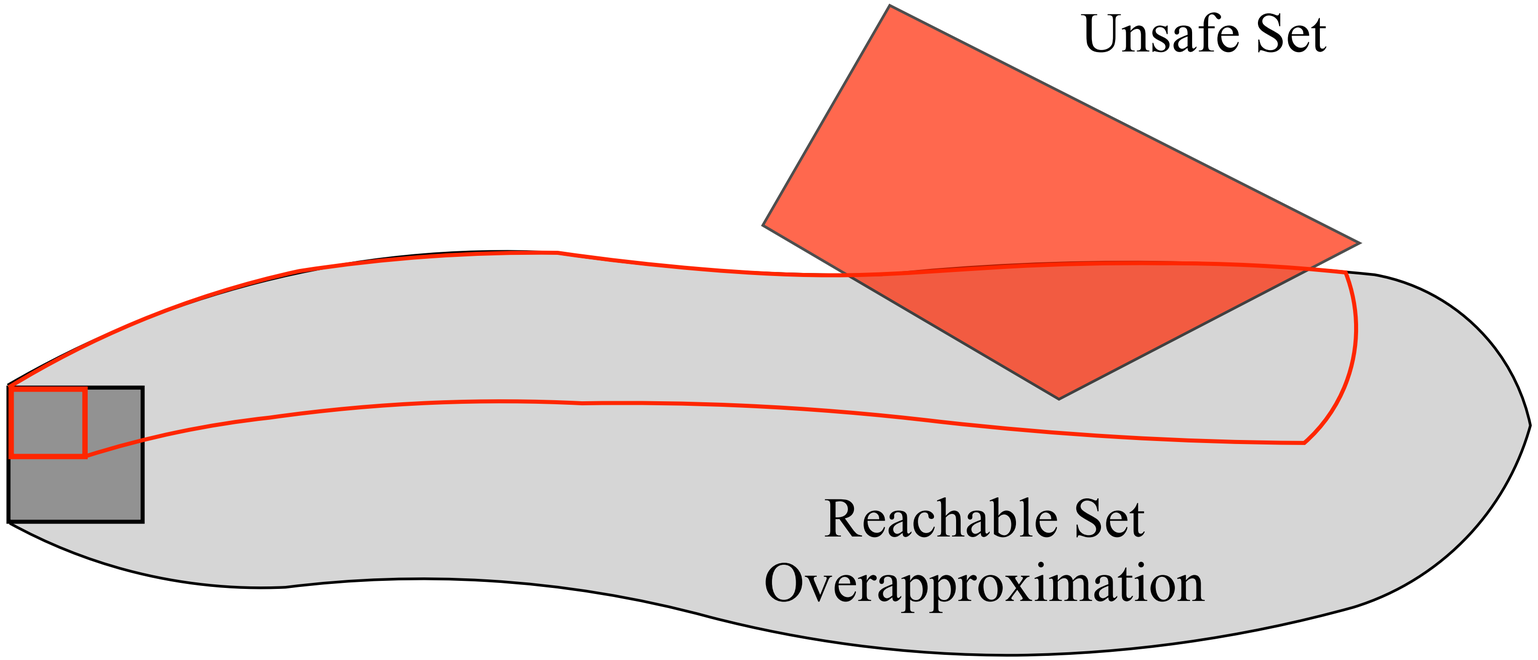
The set-propagation techniques compute *convex overapproximate ranges* of φ_f over small time intervals iteratively. **However, is it possible to overapproximate the function itself?**

Functional Overapproximation



- The actual flowmap function cannot be computed exactly.
- An overapproximate function of the flowmap contains not only the **reachable set** but also the **relation** between the reachable state and its initial state.
- Ideal overapproximation form: $\varphi_f(x(0), t, 0) \in \Phi(x(0), t, 0)$ for all $x(0) \in X$ and $t > 0$.

Counterexample Generation



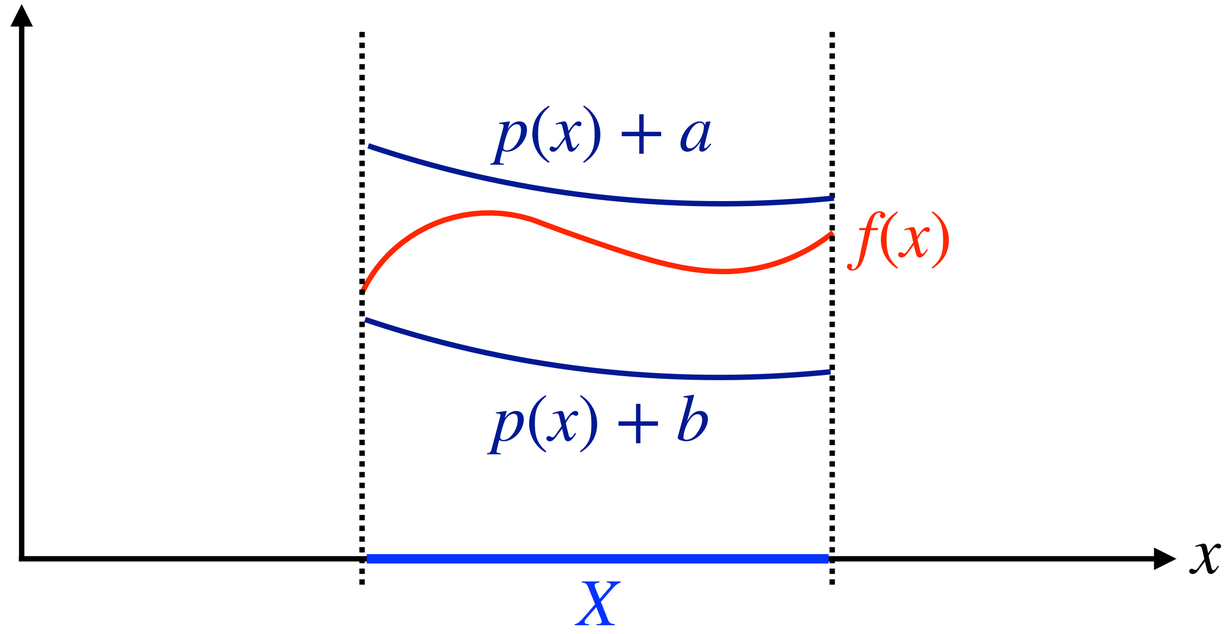
- When a pure range overapproximation intersect the unsafe set, we can only conclude that the system **might be unsafe**.
- However, if a functional overapproximation detects a potential unsafe intersection, it may backward find **an overapproximation of the unsafe initial states**.

How can we compute functional overapproximations?

Outline

- Taylor Models as Functional Overapproximations.
- Taylor Model Arithmetic.
- Taylor Model Flowpipe Construction for ODEs.
- Time-Triggered Switches of Dynamics.
- Flow* Toolbox.

Taylor Models



Overapproximation property: $\forall x \in X.(f(x) \in p(x) + [a, b])$

Kyoko Makino and Martin Berz. *Taylor Models and Other Validated Functional Inclusion Methods.* International Journal of Pure and Applied Mathematics 4 (4), 2003.

Basic Arithmetic of Taylor Models

Addition: $$(p_f(x), I_f) + (p_g(x), I_g) = (p_f + p_g, I_f + I_g)$$

Multiplication:

$$(p_f(x), I_f) \cdot (p_g(x), I_g) = (p_f \cdot p_g - r_k, B(p_f)I_g + B(p_g)I_f + I_f I_g + B(r_k))$$

Integration: $$\int_a^b (p_f(s), I_f)ds = (\int_a^b p_f(s)ds - r_k, B(r_k) + I_f \cdot [a, b])$$

$(p_f, I_f), (p_g, I_g)$ are assumed to be the TMs of some functions f, g respectively. $B(\cdot)$ denotes the interval enclosure, and r_k in a polynomial $p - r_k$ consists of the terms in p of degrees $\geq k$.

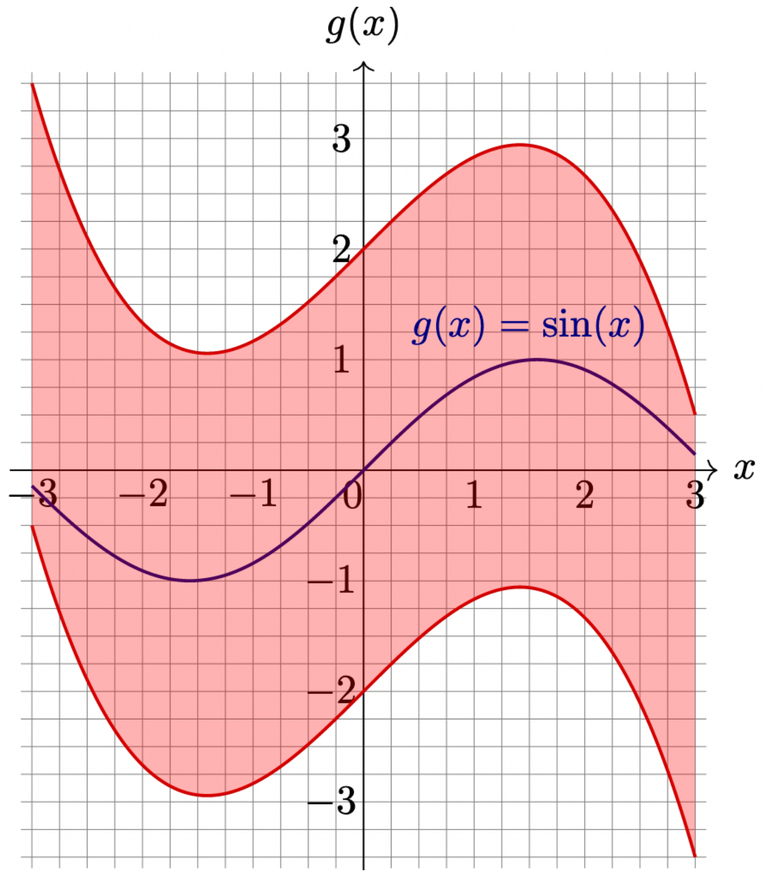
Overapproximating a Function

Steps of computing a TM $(p(x), I)$ for a function $g(x)$:

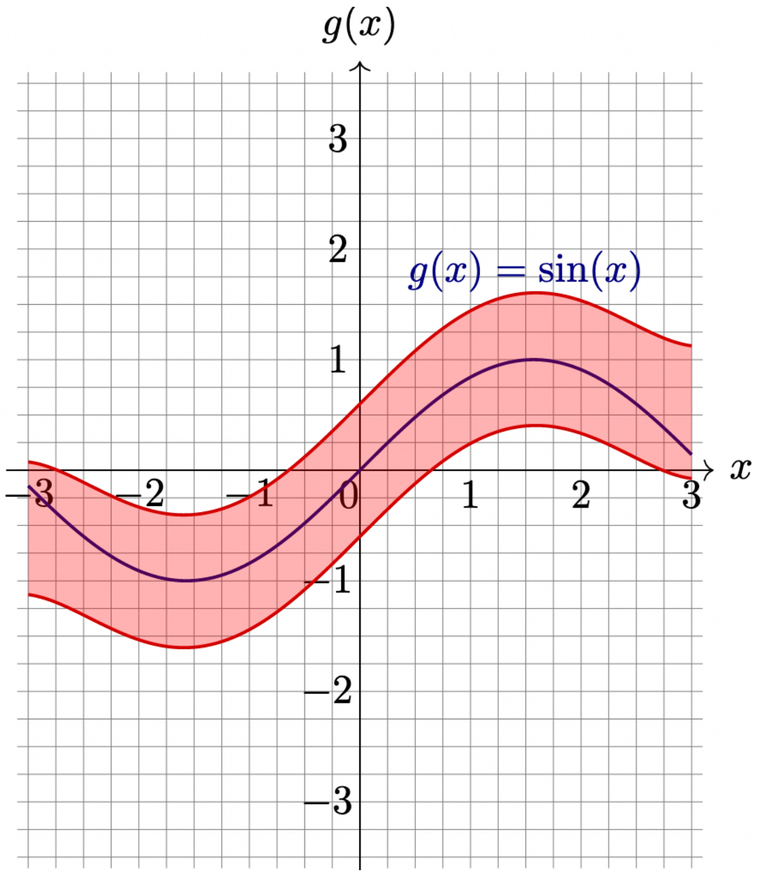
1. *Computing a* **polynomial approximation** $p(x)$ *of* $g(x)$ *w.r.t. a bounded domain* D.

2. *Evaluating a remainder interval* I *based on Lagrange form such that*

$$g(x) \in p(x) + I \text{ for all } x \in D$$

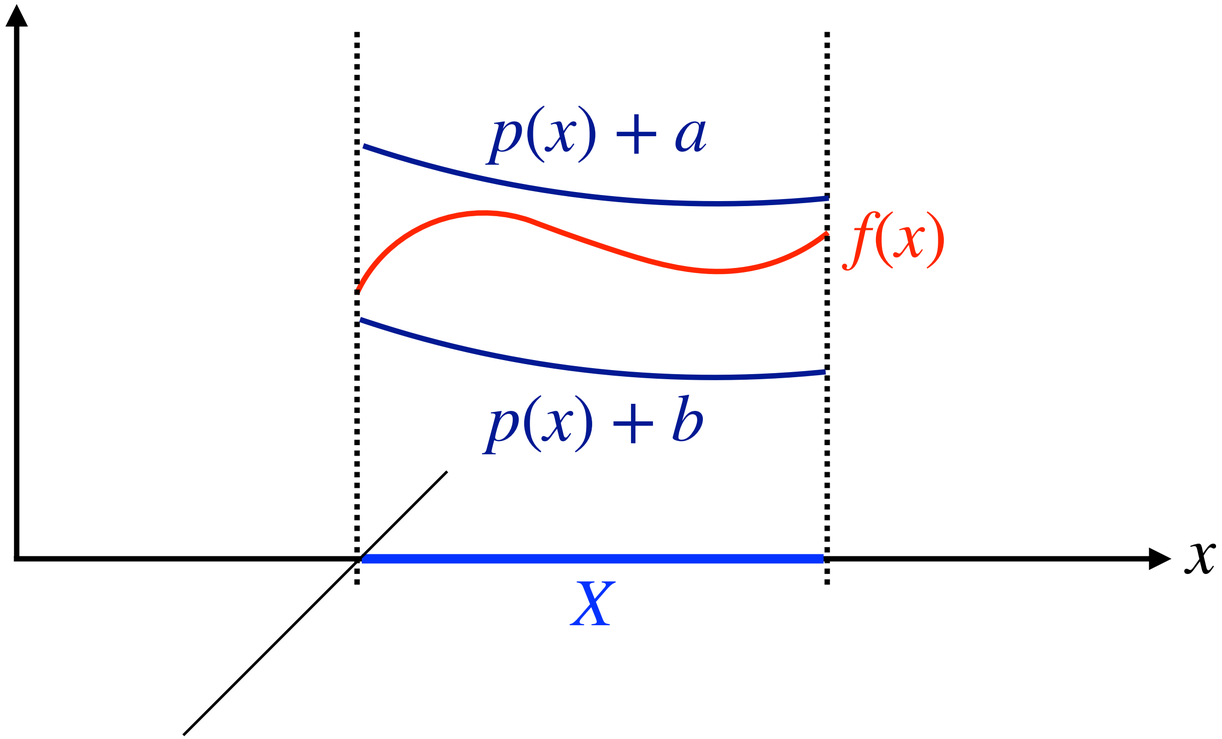


order 3



order 5

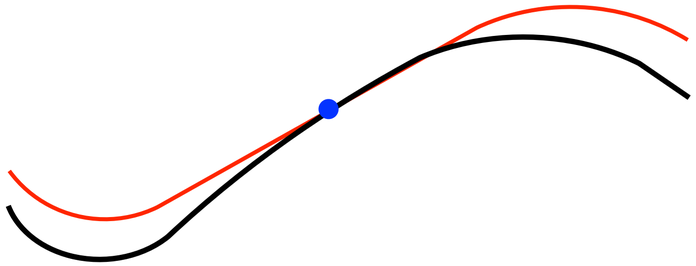
Taylor Models ≠ Taylor Overapproximations



The polynomial p could be any polynomial approximation of f, and more accurate approximation requires a smaller remainder interval.

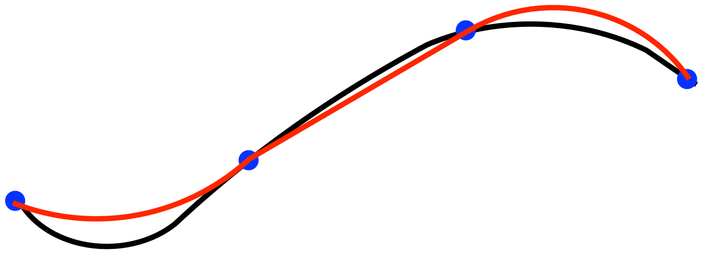
Taylor Approximation vs. Polynomial Interpolation

Taylor approximation



- Original function should be **differentiable**.
- Approximation is guaranteed to only “touch” the **expansion point**.
- Approximation quality depends on the distance to the expanding point.

Polynomial interpolation

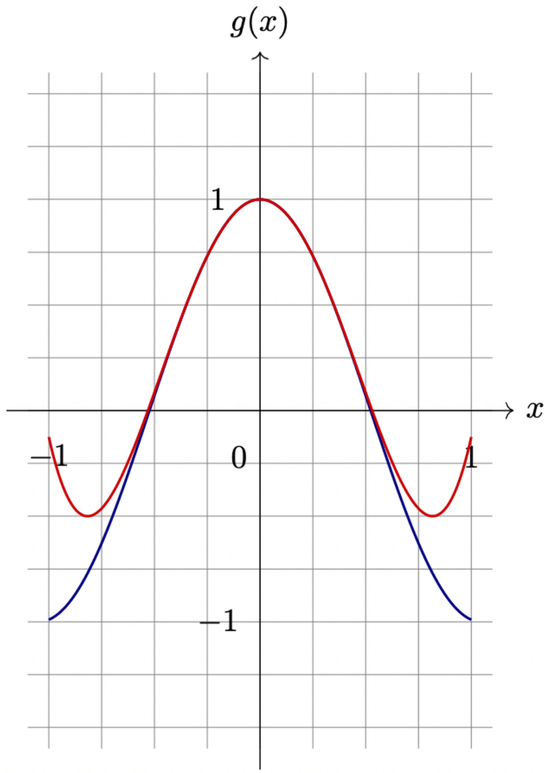


- Original function is **not required** to be differentiable.
- Approximation is guaranteed to “touch” the **interpolation points**.
- Approximation quality is overall good.

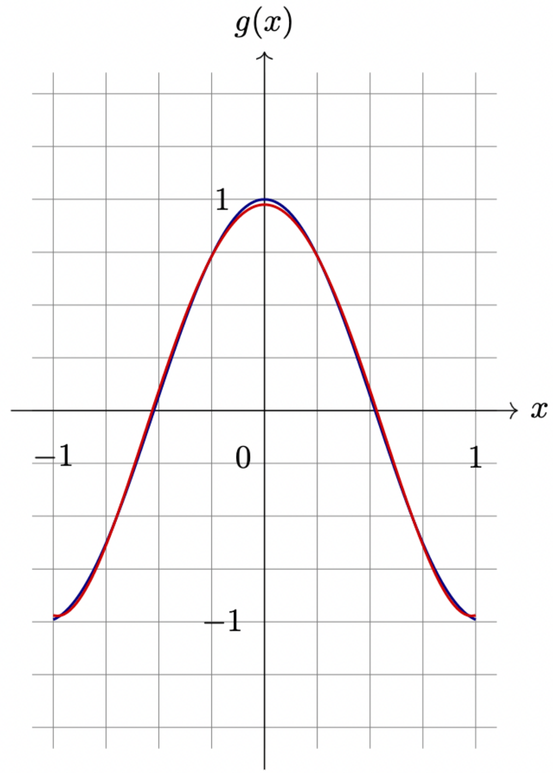
Taylor Approximation vs. Polynomial Interpolation

Interesting questions:

- Given the same order, is a polynomial interpolation always better than a Taylor approximation?
- Is is easy to evaluate a remainder for a polynomial interpolation? If so, is it tight enough?
- Can we use polynomial interpolations in Taylor model arithmetic?



(a) Order 4 Taylor approximation (in red)



(b) Order 4 Chebyshev interpolation (in red)

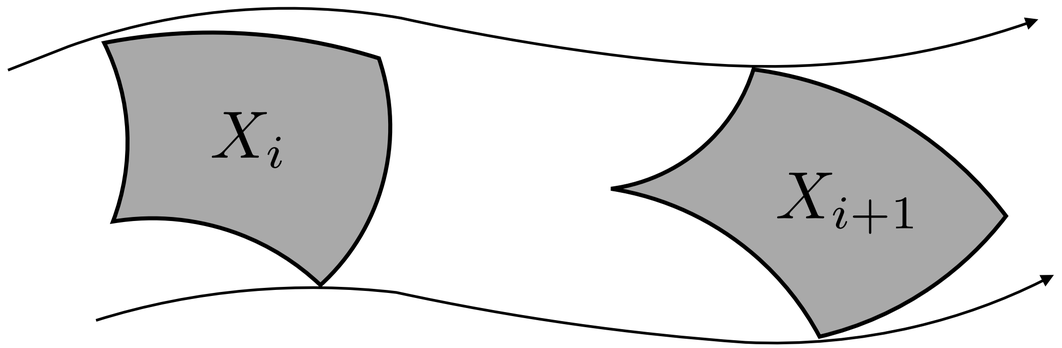
Facts

- If the original function has n variables, both of the Taylor approximation and the polynomial interpolation have n variables.
- The size of an order k polynomial which has n variables may have at most $N_k = \binom{n+k}{k}$ terms (different monomials).
- The computation of a **Taylor expansion** consists of a series of derivations at the expansion point, which often does not require to compute the zero coefficients.
- The computation of a **polynomial interpolation** often requires to compute all of the coefficients in N_k terms.

- If p is an order k **Taylor approximation** of a function, then the corresponding order $k-1$ approximation can be obtained by simply discard (truncate) the terms in p of degree k.
- If p is an order k **polynomial interpolation** of a function, then the terms in the order $k-1$ interpolation have to be re-computed.

Computing Taylor Model Flowpipes

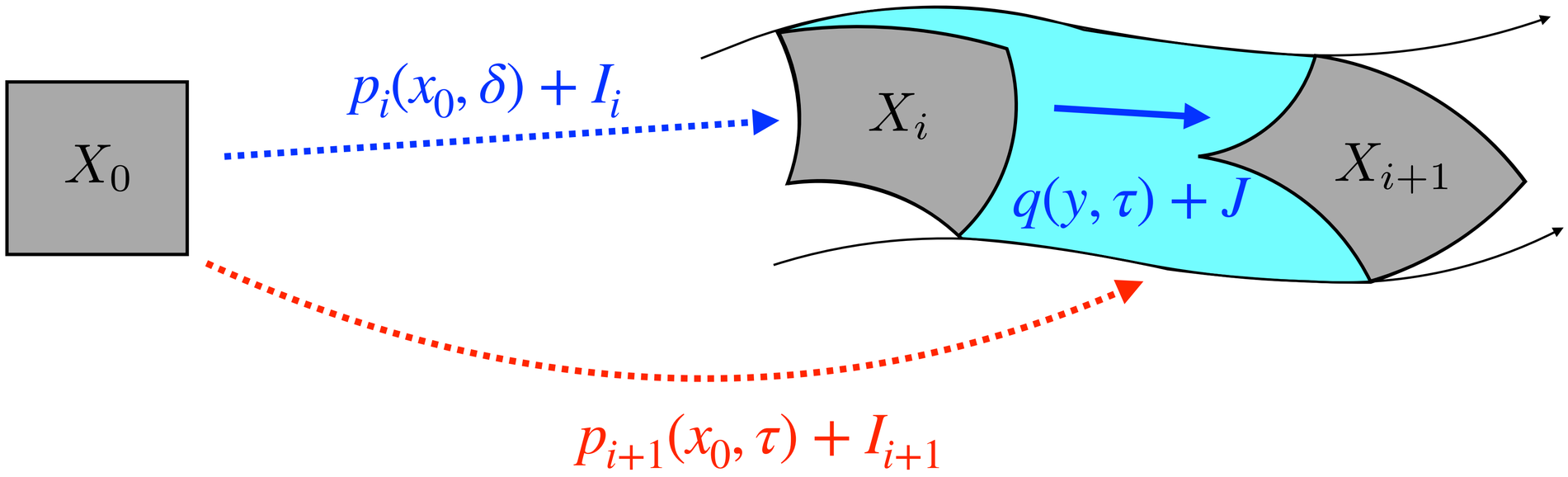
Main Framework



We perform the following steps for $i = 0,\ldots,N-1$ from the TM $(x_0,0)$ where $x_0 \in X_0$:

1. Compute a Taylor approximation $q(y,\tau)$ for the flowmap function $\varphi_f(y, i\delta+\tau, i\delta)$.
2. Evaluate a remainder J such that $\varphi_f(y, i\delta+\tau, i\delta) \in q(y,\tau)+J$ for all $y \in X_i, \tau \in [0,\delta]$.
3. Computing $p_i(x_0,\tau) = q(X_i,\tau)+J$.

Main Framework



Theorem.
For any $i = 0, \ldots, N-1$, the reachable state at a time $t \in [i\delta, (i+1)\delta]$ from any initial state $x_0 \in X_0$ is contained in the interval $p_i(x_0, t - i\delta) + I_i$.

Computing an Order k Taylor Approximation

1. Set $p(x_0, \tau) = x_0$.

2. Repeatedly compute $p(x_0, \tau) = x_0 + \int_0^\tau f(p(x_0, s), s)ds$ and only keep the terms of low degrees.

Example: $\dot{x} = 2 + x^2$

$p = x_0$ **(initial)**

$p = x_0 + \int_0^\tau (2 + x_0^2)ds \approx x_0 + 2\tau$ **(1st iteration, order 1)**

$p = x_0 + \int_0^\tau (2 + (x_0 + 2s)^2)ds \approx x_0 + 2\tau$ **(2nd iteration, order 2)**

$p = x_0 + \int_0^\tau (2 + (x_0 + 2s)^2)ds \approx x_0 + 2\tau + x_0^2\tau + 2x_0\tau^2 + \frac{4}{3}\tau^3$ **(3rd iteration, order 3)**

Evaluating a Conservative Remainder

Assume that $p(x_0, \tau)$ is already a polynomial approximation of the flowmap, then for any remainder I,

if the TM of $x_0 + \int_0^\tau f(p(x_0, s) + I, s)ds$ is contained in $p(x_0, s) + I$, then $p(x_0, s) + I$ is a functional overapproximation of the flowmap.

The Picard operation contracts the TM $p(x_0, s) + I$.

We may repeatedly perform the Picard operation to refine a valid remainder.

Roundoff Errors

Example: the decimal number $(9.4)_{10} = (1001.\overline{0110})_2$ can be rearranged into

$$+1.\underbrace{00101100110011001100110011001100110011001100110011}_{52\text{ bits}}10\ldots \times 2^3$$

[Wikipedia: Round-off error]

- Many non-integer numbers can not be represented exactly in the floating-point format.
- We need to modify the standard interval arithmetic to ensure the conservativeness. E.g., $[a, b] + [c, d] = [\underline{a + c}, \overline{b + d}]$. Interval multiplication is more complex.
- Non-interval values may be rounded to their closest floating-point numbers, and the conservativeness of the result could be verified afterwards.

Conservative Interval Multiplication

Computing a conservative interval for $[a, b] \cdot [c, d]$:

Case 1: $a \geq 0$ and $c \geq 0$. The result is $[\underline{ac}, \overline{bd}]$.

Case 2: $a \geq 0$ and $d \leq 0$. The result is $[\underline{bc}, \overline{ad}]$.

Case 3: $a \geq 0$ and $0 \in (c, d)$. The result is $[\underline{bc}, \overline{bd}]$.

Case 4: $b \leq 0$ and $c \geq 0$. The result is $[\underline{ad}, \overline{bc}]$.

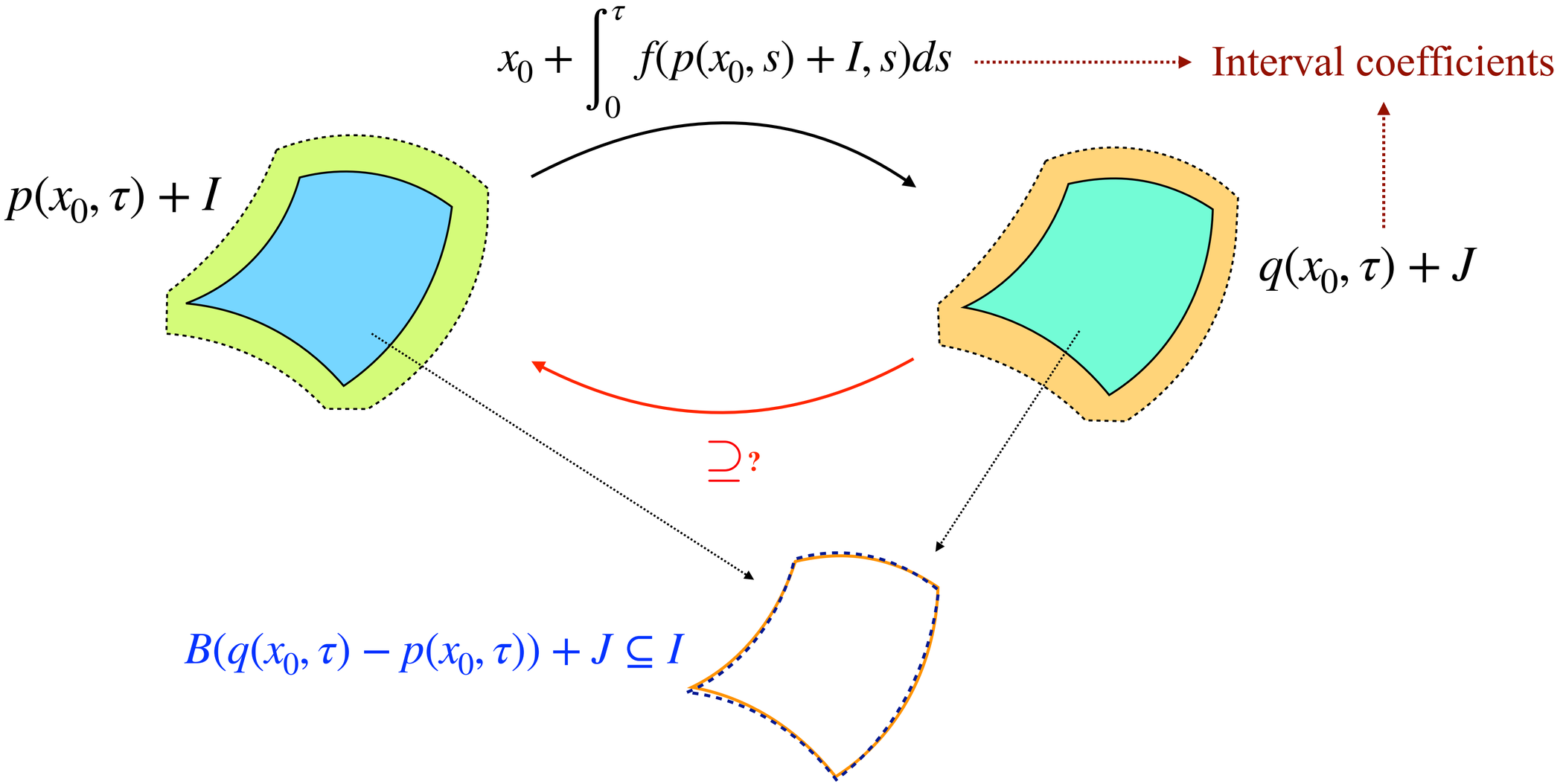
Case 5: $b \leq 0$ and $d \leq 0$. The result is $[\underline{bd}, \overline{ac}]$.

Case 6: $b \geq 0$ and $0 \in (c, d)$. The result is $[\underline{ad}, \overline{ac}]$.

…

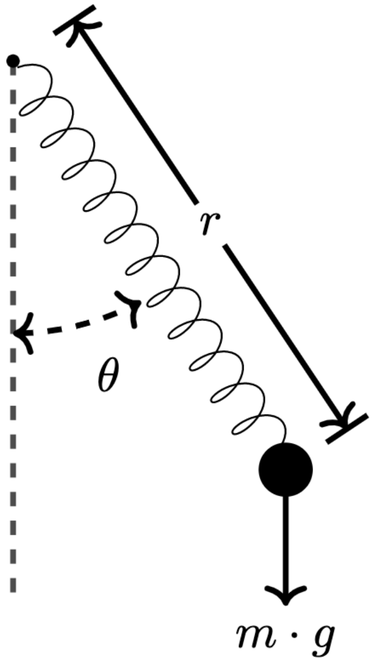
It is expensive to take roundoff errors into account, but necessary!

Conservativeness of the Remainder Evaluation



If $p(x_0, \tau)$ is the order k Taylor approximation of the flowmap, then $q(x_0, \tau) - p(x_0, \tau)$ only contains roundoff errors.

Example: Spring Pendulum

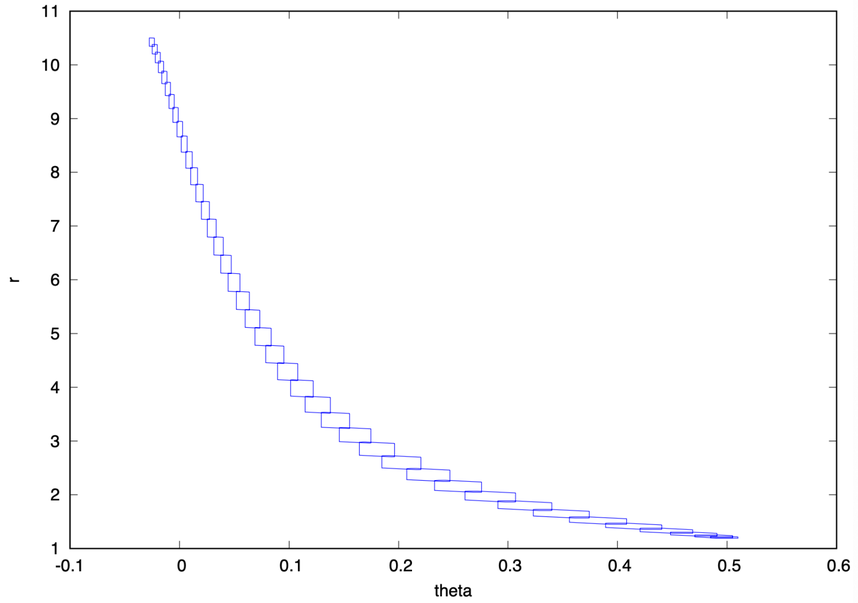


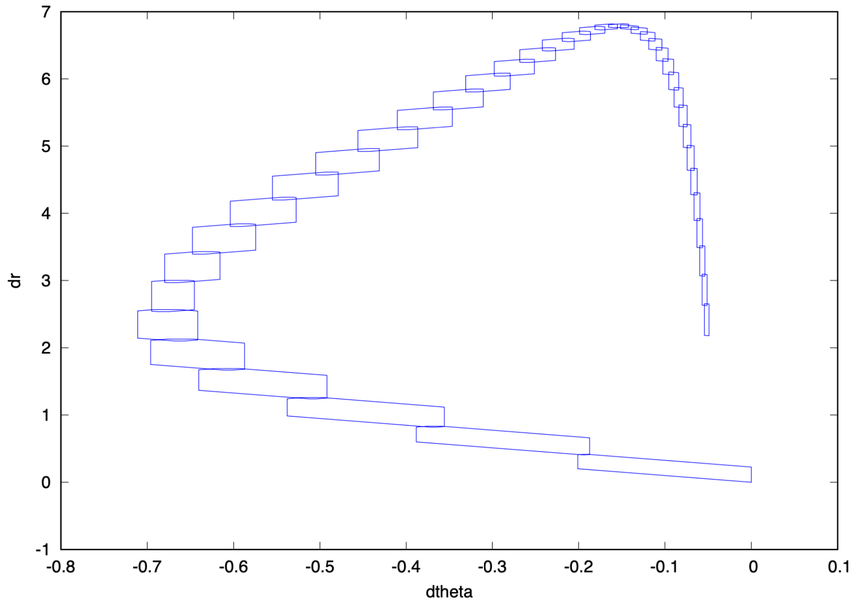
$$
\begin{aligned}
\dot{r} &= v_r \\
\dot{\theta} &= v_\theta \\
\dot{v}_r &= r \cdot v_\theta^2 + g \cdot \cos(\theta) - k \cdot (r - L) \\
\dot{v}_\theta &= -\frac{(2 \cdot v_r \cdot v_\theta + g \cdot \sin(\theta))}{r}
\end{aligned}
$$

Right Figures:

Taylor model flowpipes (wrapped by octagons) computed from the initial set $r(0) \in [1.19,1.21], \theta(0) \in [0.49,0.51]$, $v_r(0) = 0, v_\theta(0) = 0$.

Stepsize: 0.05, Order: 5.





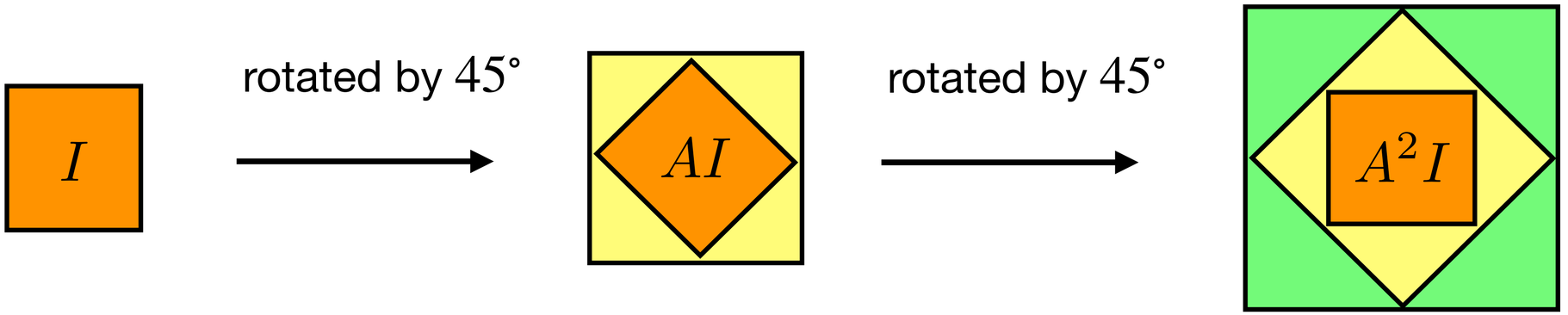
Performance Improvement

- **Adaptive stepsize for the TM flowpipes** - For each TM, finding the largest stepsize according to a given remainder.
- **Adaptive approximation order** - For each TM, finding the smallest order according to a given remainder.
- **Cutting off the "small" terms regularly** - The polynomial of a TM may have at most $\binom{n+k}{k}$ terms. We may regularly remove the terms whose range are smaller than a threshold and add their interval enclosures to the remainder.

However, the above methods do not improve the overall performance.

Accumulation of Overestimation

$$A = \begin{pmatrix} \cos(\frac{\pi}{4}) & -\sin(\frac{\pi}{4}) \\ \sin(\frac{\pi}{4}) & \cos(\frac{\pi}{4}) \end{pmatrix} \qquad I = \begin{pmatrix} [-1,1] \\ [-1,1] \end{pmatrix}$$

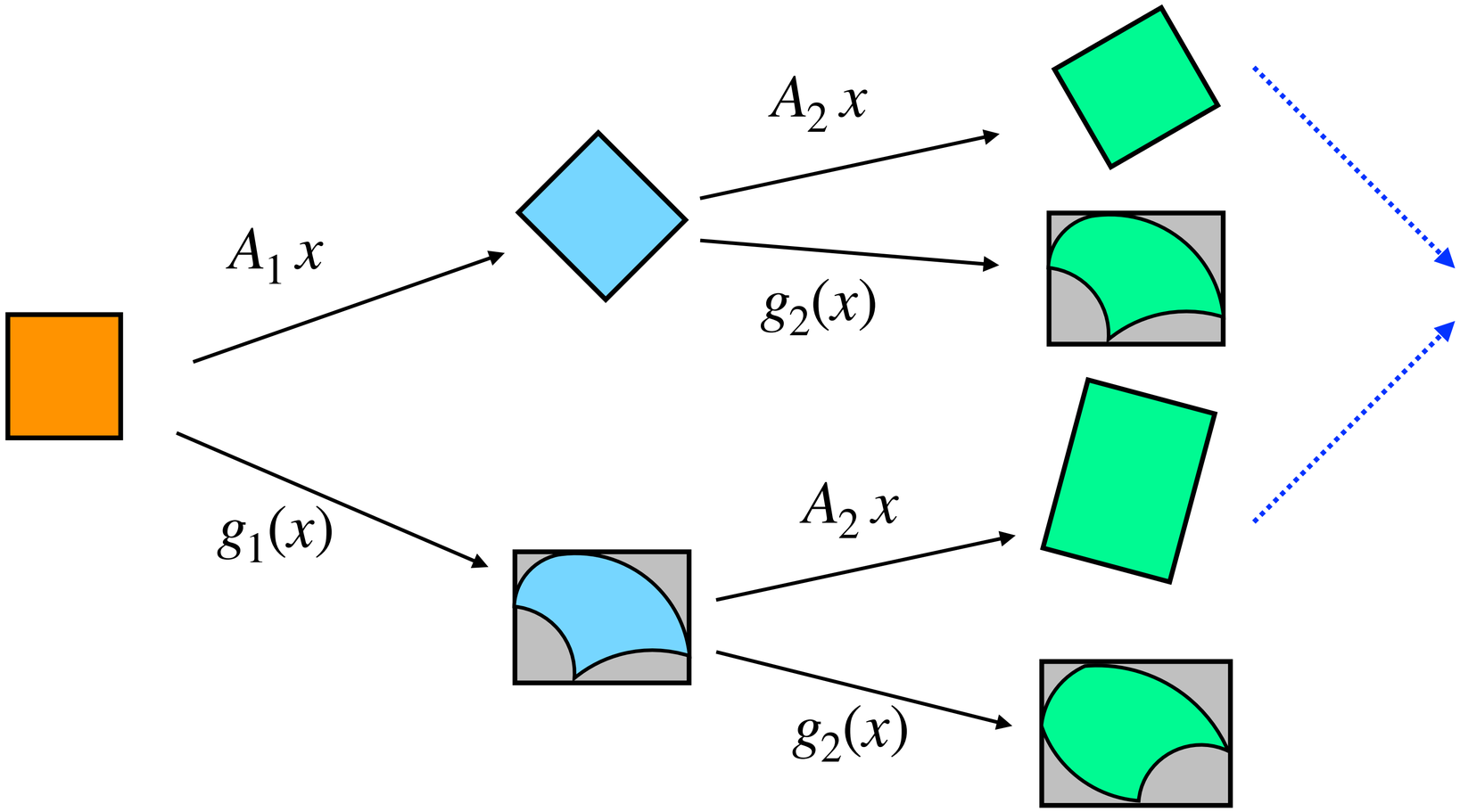


Taylor model computation may also have wrapping effect:

- Remainders are always represented as intervals.
- Non-polynomial parts are always evaluated by interval arithmetic.

Using Symbolic Representations

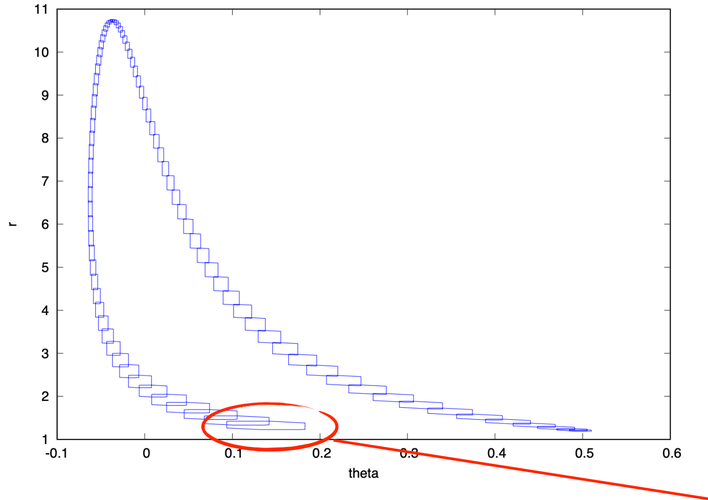
$$f_1(x) = A_1 x + g_1(x) \qquad f_2(x) = A_2 x + g_2(x)$$

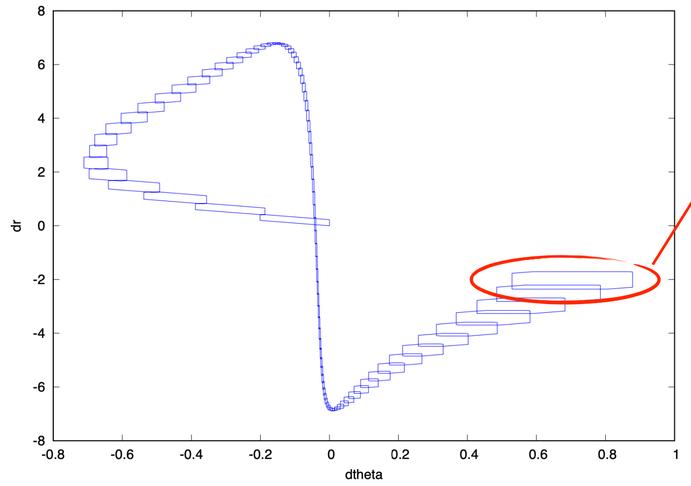


We do not compute the the intervals out, but track their linear transformations.

Xin Chen and Sriram Sankaranarayanan. *Decomposed Reachability Analysis for Nonlinear Systems.* In Real-Time Systems Symposium (RTSS'16), pp. 13–24. IEEE, 2016.

Example: Spring Pendulum



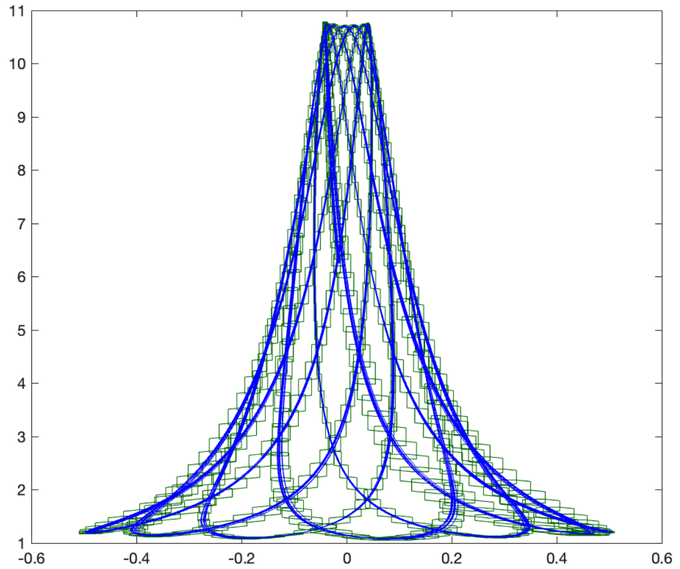


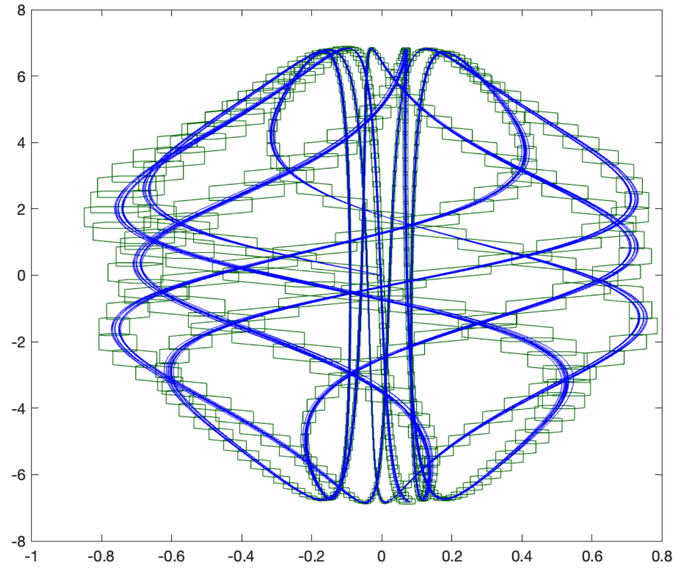
Using Interval Remainders

Stepsize: 0.05, Order: 5,
T = 30.

Cannot compute the next TM due to the too large overestimation.

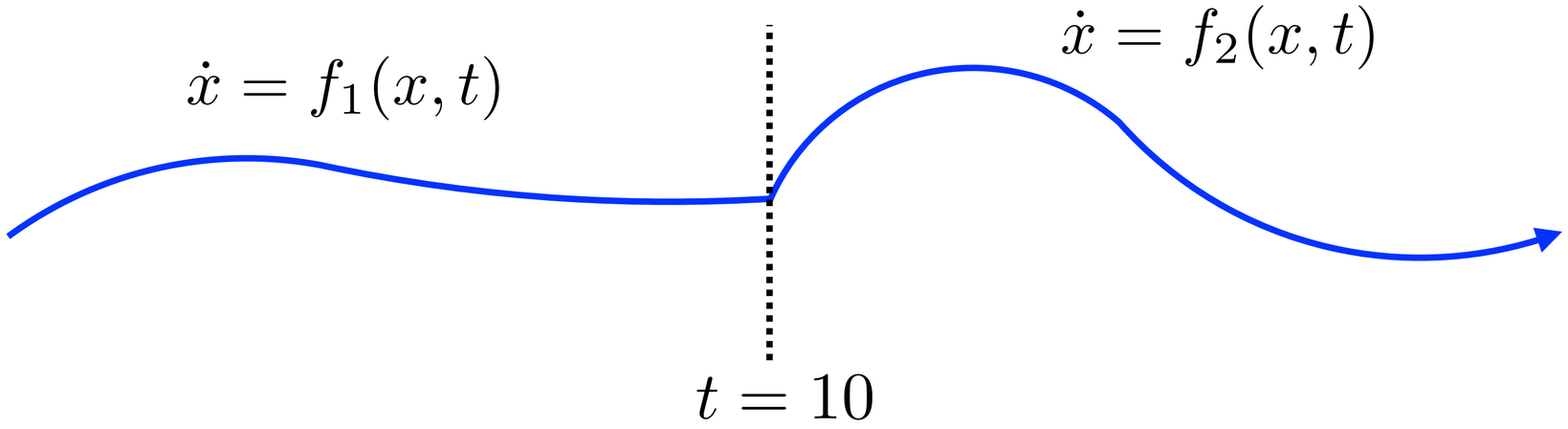
Time cost: 18 seconds
(M1 Mac)



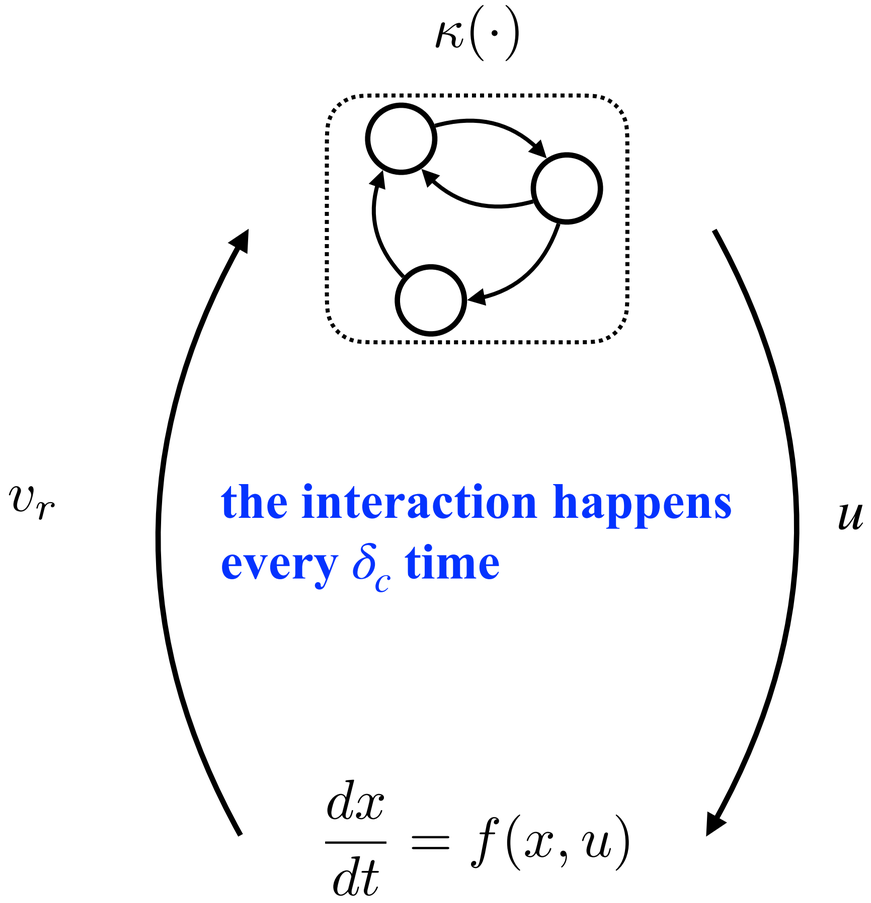


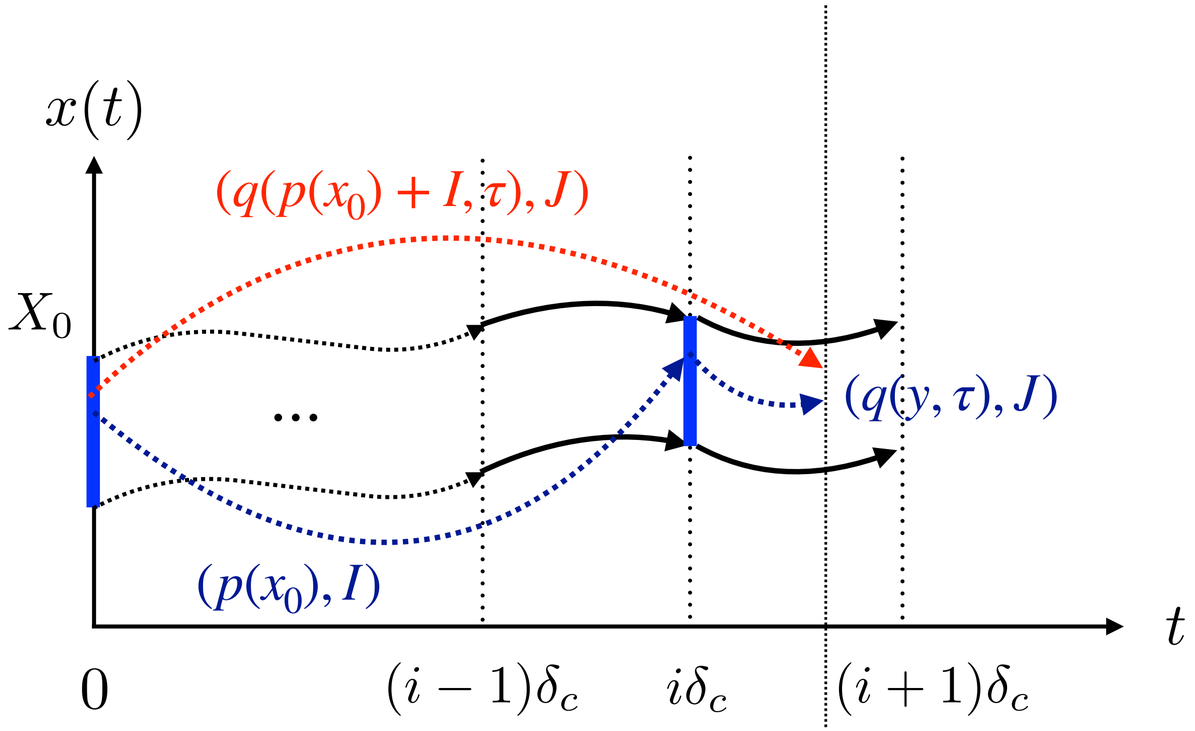
Using Symbolic Remainders

Time-Triggered Switches



State Feedback System





Deterministic flowmaps with time-triggered switches can be overapproximated by TMs using the standard flowpipe construction framework.

Example: Quadcopter

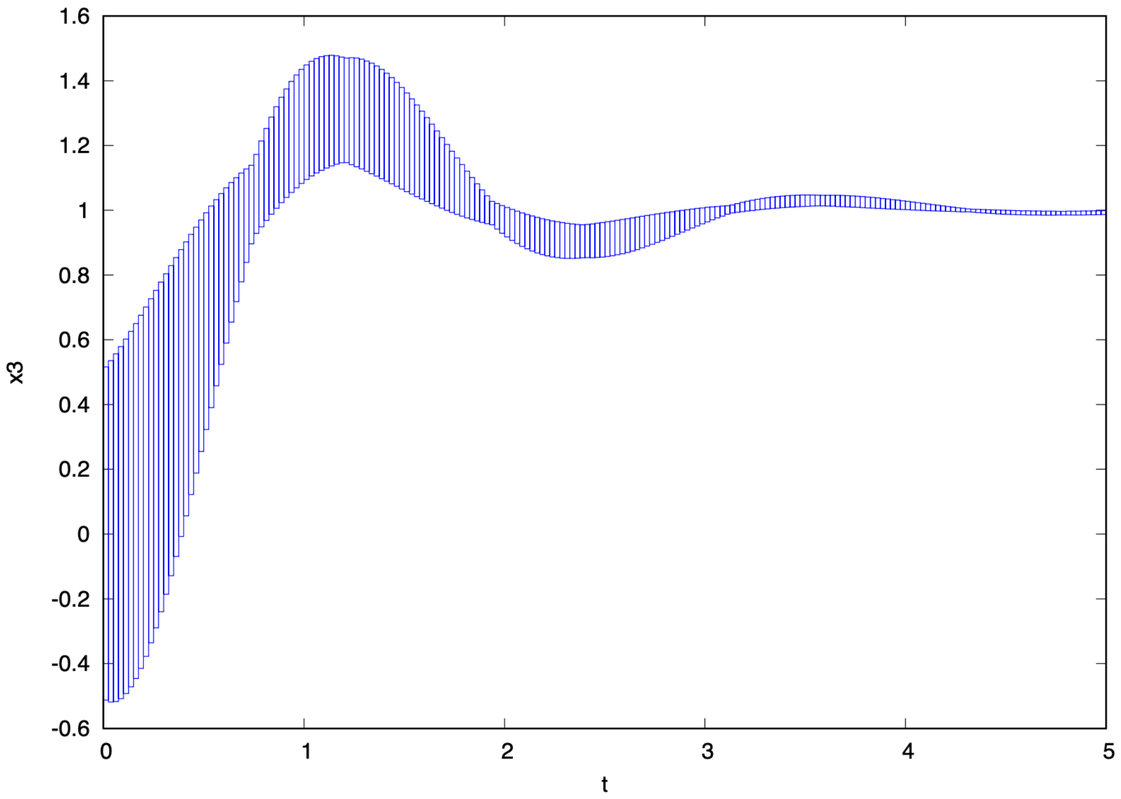
$$
\begin{cases}
\dot{x}_1 = \cos(x_8)\cos(x_9)x_4 + (\sin(x_7)\sin(x_8)\cos(x_9) - \cos(x_7)\sin(x_9))\,x_5 \\
\qquad + (\cos(x_7)\sin(x_8)\cos(x_9) + \sin(x_7)\sin(x_9))\,x_6 \\
\dot{x}_2 = \cos(x_8)\sin(x_9)x_4 + (\sin(x_7)\sin(x_8)\sin(x_9) + \cos(x_7)\cos(x_9))\,x_5 \\
\qquad + (\cos(x_7)\sin(x_8)\sin(x_9) - \sin(x_7)\cos(x_9))\,x_6 \\
\dot{x}_3 = \sin(x_8)x_4 - \sin(x_7)\cos(x_8)x_5 - \cos(x_7)\cos(x_8)x_6 \\
\dot{x}_4 = x_{12}x_5 - x_{11}x_6 - g\sin(x_8) \\
\dot{x}_5 = x_{10}x_6 - x_{12}x_4 + g\cos(x_8)\sin(x_7) \\
\dot{x}_6 = x_{11}x_4 - x_{10}x_5 + g\cos(x_8)\cos(x_7) - g - u_1/m \\
\dot{x}_7 = x_{10} + \sin(x_7)\tan(x_8)x_{11} + \cos(x_7)\tan(x_8)x_{12} \\
\dot{x}_8 = \cos(x_7)x_{11} - \sin(x_7)x_{12} \\
\dot{x}_9 = \dfrac{\sin(x_7)}{\cos(x_8)}x_{11} - \sin(x_7)x_{12} \\
\dot{x}_{10} = \dfrac{J_y - J_z}{J_x}x_{11}x_{12} + \dfrac{1}{J_x}u_2 \\
\dot{x}_{11} = \dfrac{J_z - J_x}{J_y}x_{10}x_{12} + \dfrac{1}{J_y}u_3 \\
\dot{x}_{12} = \dfrac{J_x - J_y}{J_z}x_{10}x_{11} + \dfrac{1}{J_z}\tau_\psi
\end{cases}
$$

Feedback Law (applied every 0.1 seconds):

$$u_1 = 7.14285714285714(x_3 - 1) - 2.14285714285714x_6$$

$$u_2 = -x_7 - x_{10}$$

$$u_3 = -x_8 - x_{11}$$



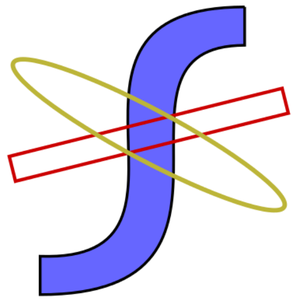
$x_1 \in [-0.4, 0.4], x_2 \in [-0.4, 0.4], x_3 \in [-0.4, 0.4], x_4 \in [-0.4, 0.4],$
$x_5 \in [-0.4, 0.4], x_6 \in [-0.4, 0.4], x_7 = 0, x_8 = 0, x_9 = 0, x_{10} = 0, x_{11} = 0, x_{12} = 0$

Time cost: 4.8 seconds (M1 Mac)

Other Operations on Taylor Models

- **Intersection.** When a condition is associated with a switch, we need to find out all TMs that satisfy the condition.
- **Union**. When several switches may be performed nondeterministically, we sometimes need to merge the TMs to reduce the time cost.

However we are not going to use them in this course.



Flow* Toolbox -

A Platform for Modeling and Analysis of Cyber-Physical Systems

Brief Introduction

- Formal verification tool for time-bounded reachability analysis and safety verification.
- Discrete, continuous and hybrid dynamical systems.
- Using Taylor models, intervals, zonotopes, and convex polyhedra as set representations.
- The **first version** was released in 2013 [RTSS'12, CAV'13].
- **Version 1.2.0** was released in 2015 (Performance improvement).
- **Version 2.1.0** was released in 2017 (Performance improvement and new features [RTSS'16, EMSOFT'17]).
- **Toolbox version** will be released soon. (Re-designed data structures, new features, significant performance improvement).

Website of the Tool

[https://github.com/chenxin415/flowstar]

Flow* Toolbox -

A Platform for Modeling and Analysis of Cyber-Physical Systems

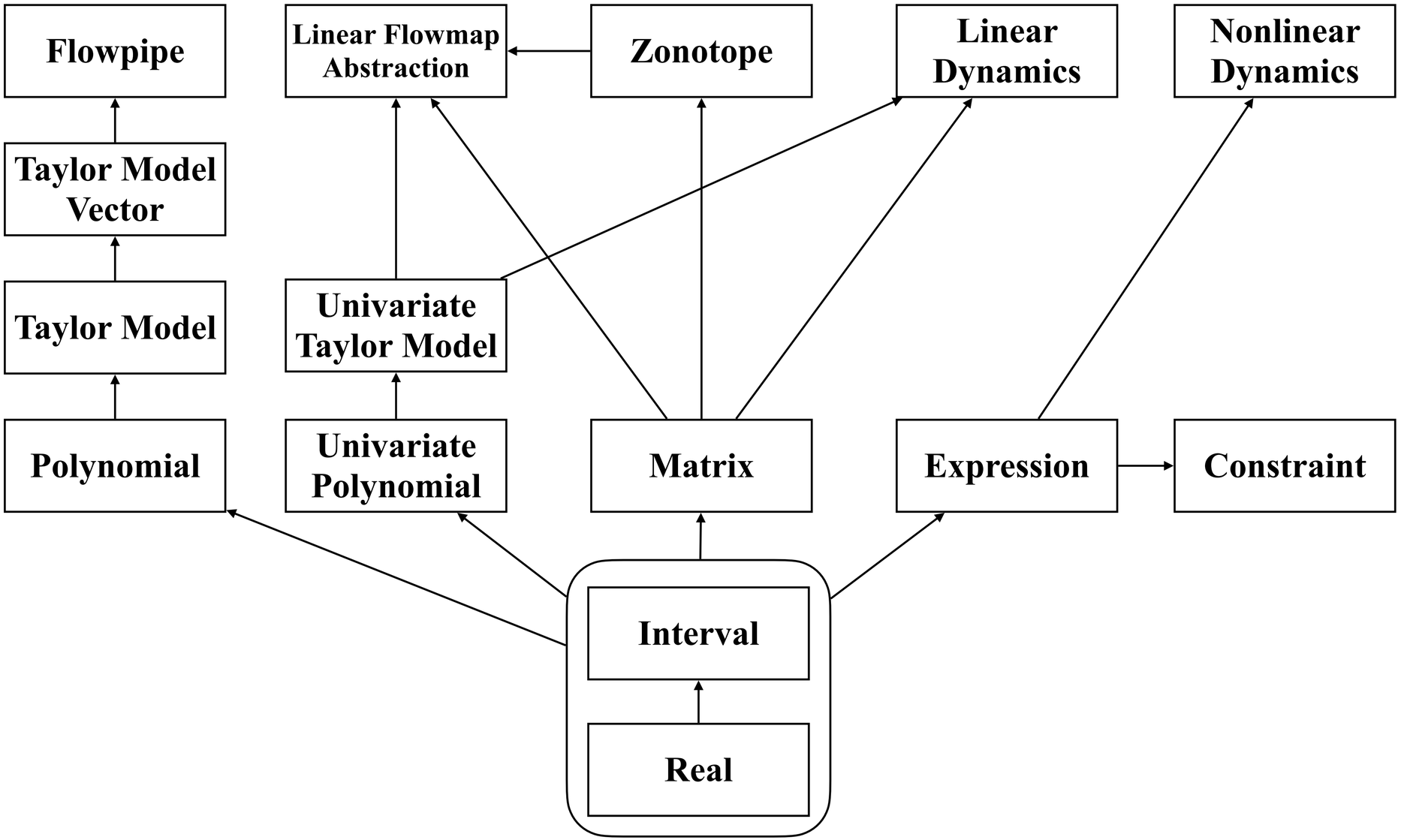
Introduction

This is the homepage of the toolbox version of Flow*. The first version of Flow* was released in the year of 2013, and improved in 2015 (version 1.2.0) and 2017 (version 2.1.0). The purpose of releasing a toolbox version is to provide a more flexible way to model and analyze cyber-physical systems (CPS), and expose the key functions to the tools for verifying more complex systems, such as the CPS with machine learning components. The main data structures in the toolbox version are completely re-designed and implemented such that the performance is **at least 10x faster** than the version 2.1.0.

How to Use the Toolbox?

Online Verification for a Racing Car. [IROS'20]

Data Structures (Toolbox Version)



Conservativeness

- **Polynomial coefficients** - It is unnecessary to represent all real numbers as intervals. We only need to ensure that a transformation from a TM $\pi(p(x) + I)$ regarding to $x \in D$ produces a TM $q(x) + J$ which is guaranteed to contain the actual result.

- In Flow*, the roundoff errors in **flowpipe construction** for nonlinear ODE is considered in the remainders. The only transformation is composing two TMs $q(p(x) + I, t) + J$ such that J is already guaranteed to contain the roundoff error when the contractiveness of the Picard operation on $p(x) + I$ is verified.

- In Flow*, roundoff errors are taken into account in a **range overapproximation** of TMs. Every resulting real value is calculated as an interval.

- In Flow*, roundoff errors are taken into account in **safety verification**.

- Whether or not considering roundoff errors in other operations can be decided by users.

Installation

- The following GNU open-source libraries should be pre-installed:

M4, GMP, MPFR, GSL, GLPK, BISON, FLEX

most of them are available at https://ftp.gnu.org/. All of them can be compiled and installed using the 3 steps: ./configure, make, and make install.

- Flow* requires GCC 8.0 or later versions.

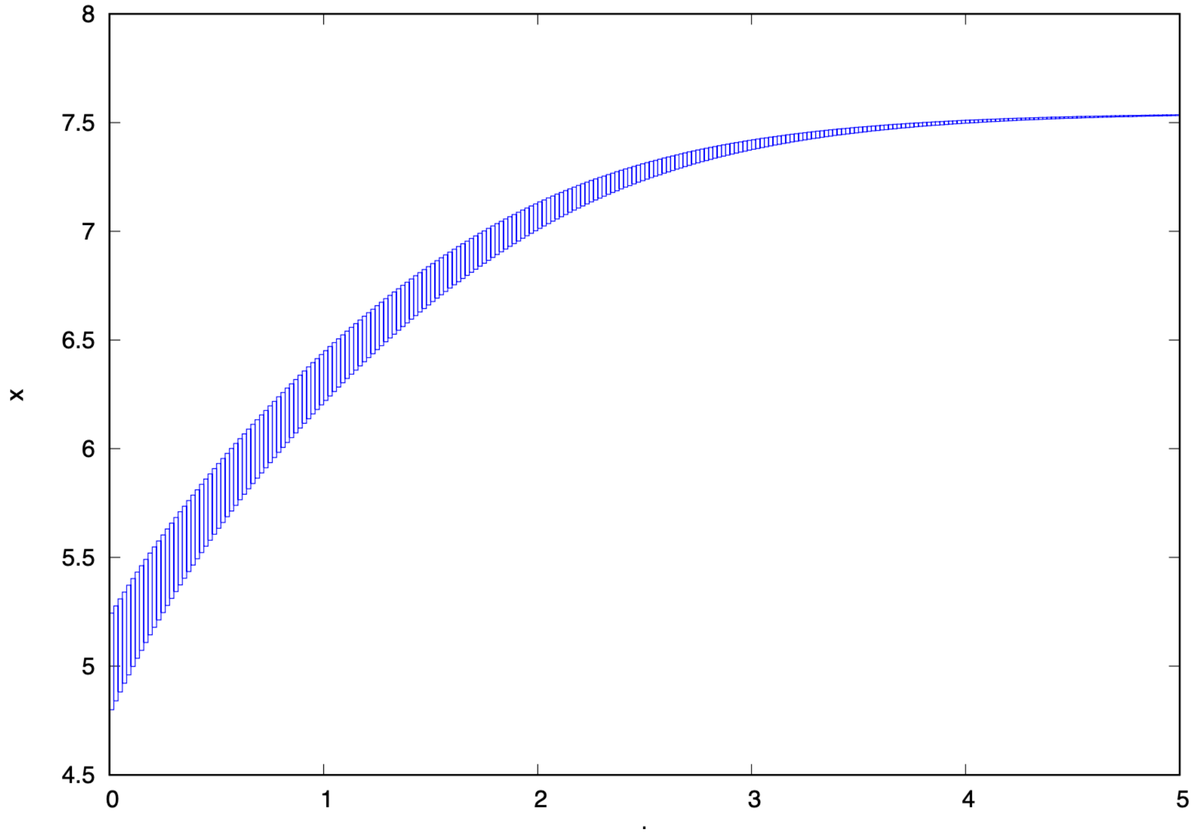
- The source code of Flow* can be compiled by simply running make. If it is successful, a file named libflowstar.a will be created.

Modeling a Simple Reachability Problem

ODE: $\dot{x} = 1 - \sin(x)\dfrac{\sqrt{\log(x)}}{e^{\cos(x)}}$

Initial Set: $x(0) \in [4.8, 5.2]$

Flow* Setting: **stepsize: 0.02, order: 5**
cutoff: 1e-10



Time cost: 1.8 seconds (M1 Mac)

Exercises

1. Compile and run the tool Flow*.
2. Use Flow* to compute the reachable set of the Van der Pol oscillator:

$$\dot{x} = y,\ \ \dot{y} = (1 - x^2)y - x$$

   from the initial set $x(0) \in [1.35,1.55], y(0) \in [2.35,2.45]$.

3. Try some different settings in Flow* and compare the results to CAPD.

Reading Assignments

- Polynomial Regression.
- Polynomial Interpolation.
- Bernstein Polynomial.
- Feedforward Neural Network.