Approximate reachability computation for polynomial systems

Thao Dang

VERIMAG, CNRS (France)

Introduction

Reachable set computation for differential equations

- often requires symbolic computation (to handle non-determinism)

We propose to use 'set integration' which combines 'traditional' numerical integration schemes with set computations

- Numerical integration can be applied to general differential equations and provides efficient error control mechanisms
- Convenient to approximate single solutions, but we need to characterize sets of all possible solutions

Reachable set computation using numerical integration

$$\dot{\mathbf{x}}(t) = g(\mathbf{x}(t)); \quad \mathbf{x}(0) = \mathbf{x}_0$$

Main idea

- A typical numerical scheme:

$$\mathbf{x}_{k+1} = \mathcal{Y}_k(g, h, \mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_k)$$

  where h is the step size.

- Set integration: computing such schemes with sets, that is $\mathbf{x}_k$ is not a single point in $\mathbb{R}^n$ but a subset of $\mathbb{R}^n$

In this presentation, we apply this idea to polynomial differential equations

Polynomial differential equations

We consider a polynomial differential equation:

$$\dot{\mathbf{x}}(t) = g(\mathbf{x}(t)) = A\mathbf{x}(t) + f(\mathbf{x}(t))$$

f is the non-linear part
Considering $f(\mathbf{x}(t))$ as independent input, we can write:

$$\mathbf{x}k + 1 = e^{Ah}\mathbf{x}_k + \int_0^h e^{A(h-\tau)} f(\mathbf{x}(t_k + \tau))\, d\tau.$$

Approximate $\mathbf{x}(t_k + \tau)$ by $\alpha(t_k + \tau) = \mathbf{x}_k + g(\mathbf{x}_k)\tau$. Then,

$$\bar{\mathbf{x}}_{k+1} = e^{Ah}\mathbf{x}_k + \int_0^h e^{A(h-\tau)} g(\alpha(t_k + \tau))\, d\tau = e^{Ah}\mathbf{x}_k + Q(\mathbf{x}_k)$$

The map $Q(\mathbf{x}_k)$ is a polynomial in $\mathbf{x}_k$. We shall compute this map using Bézier techniques

Second order approximation scheme

Definitions

A multi-index $\mathbf{i} = (\mathbf{i}[1], \ldots, \mathbf{i}[n+1])$ is a vector of non-negative integers. The norm $||\mathbf{i}|| = \sum_{j=1}^{n+1} \mathbf{i}[j]$

Δ simplex with vertices $\{\mathbf{v}_1, \ldots, \mathbf{v}_{n+1}\}$.
$\mathbf{x} \in \Delta$, $\lambda(\mathbf{x})$: barycentric coordinates of $\mathbf{x}$ w.r.t Δ, that is, $\mathbf{x} = \sum_k \lambda_k(\mathbf{x})\mathbf{v}_k$ and $\sum_k \lambda_k(\mathbf{x}) = 1$.

For $\mathbf{x} \in \Delta$, polynomial $\boldsymbol{\pi}(\mathbf{x})$ (of degree d) can be expressed as a Bézier simplex:

$$\boldsymbol{\pi}(\mathbf{x}) = \sum_{||\mathbf{i}||=d} \mathbf{p}_\mathbf{i} B_{\mathbf{i},d}(\lambda_1(\mathbf{x}), \ldots, \lambda_{n+1}(\mathbf{x}))$$

where $B_{\mathbf{i},d}(y_1, \ldots, y_{n+1}) = \binom{d}{\mathbf{i}} y_1^{\mathbf{i}[1]} y_2^{\mathbf{i}[2]} \ldots y_{n+1}^{\mathbf{i}[n+1]}$ (Bernstein polynomials), with the multimonial coefficient $\binom{d}{\mathbf{i}} = \frac{d!}{\mathbf{i}[1]!\mathbf{i}[2]!\ldots\mathbf{i}[n+1]!}$.

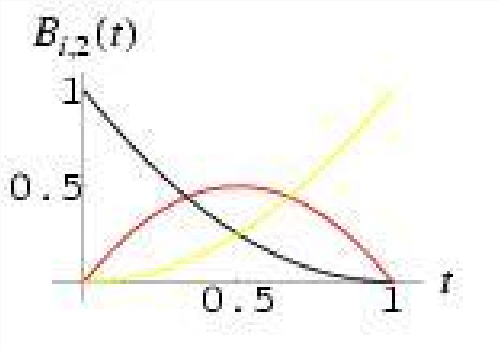
Each vector $\mathbf{p}_\mathbf{i} \in \mathbb{R}^n$: Bézier control point. All such $\mathbf{p}_\mathbf{i}$ form the Bézier control net.

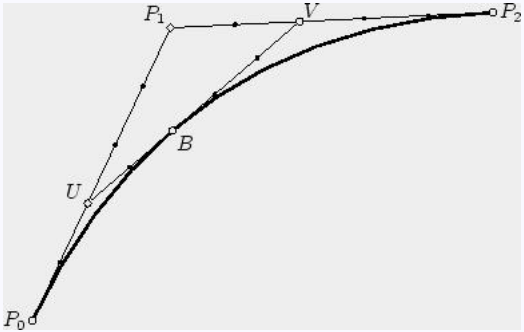
Example - Bézier curves

Dimension $n = 1$. With $d = 2$, multi-indices with $||\mathbf{i}|| = 2$: $\{(0,2),(1,1),(2,0)\}$.

A quadratic Bézier curve is: $\boldsymbol{\pi}(\mathbf{x}) = \sum_{\mathbf{i}} \mathbf{p}_{\mathbf{i}} B_{\mathbf{i},2}(\lambda_1(\mathbf{x}), \lambda_2(\mathbf{x}))$, for $\mathbf{x} \in [\mathbf{v}_1, \mathbf{v}_2]$.

There are 3 control points $\mathbf{p}_{\mathbf{i}}$.



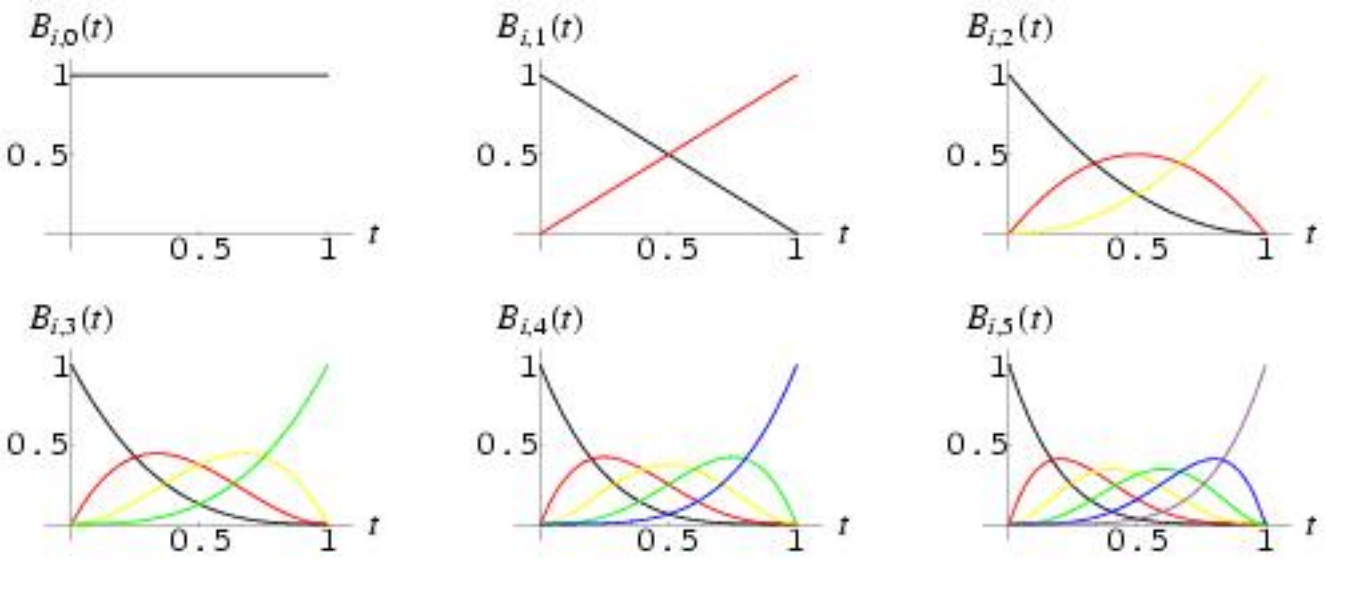


Bernstein polynomials

$$B_{\mathbf{i},2}(t) = B_{\mathbf{i},2}(y_1, y_2) = \binom{2}{\mathbf{i}} y_1^{\mathbf{i}[1]} y_2^{\mathbf{i}[2]}, y_1 = t \in [0,1], y_2 = 1 - t$$

Example - Bézier curves (cont'd)

Dimension $n = 1$.



Bernstein polynomials

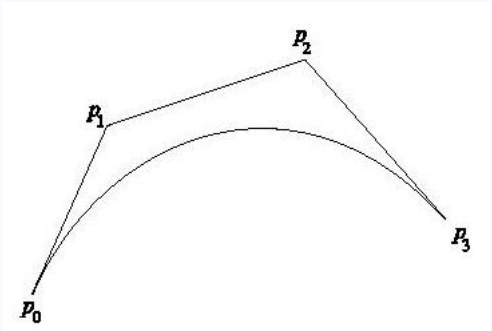
$$B_{\mathbf{i},d}(t) = B_{\mathbf{i},d}(y_1, y_2) = \binom{d}{\mathbf{i}} y_1^{\mathbf{i}[1]} y_2^{\mathbf{i}[2]}, y_1 = t \in [0,1], y_2 = 1 - t$$

Bézier simplices - Shape Properties

Any polynomial can be written in form of a Bézier simplex. Given an arbitrary point $\mathbf{x} \in \Delta$,

1. **Convex hull property**: the point $\boldsymbol{\pi}(\mathbf{x})$ lies inside the convex hull of the control net
2. **End-point interpolation property**: the polynomial $\boldsymbol{\pi}$ interpolates the control net at the corner control points.

Number of control points $\mathbf{p_i}$ is $\binom{d+n}{n} = \frac{(d+n)!}{d!\,n!}$.



$\Rightarrow$ Using the convex hull of the Bézier control net as a **tight over-approximation** of $\boldsymbol{\pi}(\Delta)$.

Computing the Bézier control points

Problem: For a polynomial $\boldsymbol{\pi}$ given in monomial form and a simplex $\Delta = conv\{\mathbf{v}_1, \ldots, \mathbf{v}_{n+1}\}$, we want to compute the control net of $\boldsymbol{\pi}$ with respect to Δ.

Blossoming principe For any polynomial $\boldsymbol{\pi} : \mathbb{R}^n \to \mathbb{R}^n$ of degree d, there is a unique symmetric d-affine map $\beta : \mathbb{R}^{nd} \to \mathbb{R}^n$ such that for all $\mathbf{x} \in \mathbb{R}^n$: $\beta(\mathbf{x}, \ldots, \mathbf{x}) = \boldsymbol{\pi}(\mathbf{x})$.
Recall: $q(\mathbf{x}_1, \ldots, \mathbf{x}_d)$ is *d-affine* if it is affine when all but one of its arguments are kept fixed; symmetric if its value does not depend on the ordering of the arguments.

Connection with the Bézier control net:

$$\mathbf{p_i} = \beta(\underbrace{\mathbf{v}_1, \ldots, \mathbf{v}_1}_{i[1]}, \underbrace{\mathbf{v}_2, \ldots, \mathbf{v}_2}_{i[2]}, \ \ldots, \ \underbrace{\mathbf{v}_{n+1}, \ldots, \mathbf{v}_{n+1}}_{i[n+1]})$$

$\Rightarrow$ To compute $\mathbf{p_i}$ we can evaluate the blossom with some particular arguments

Evaluating blossom values

Illustrate the computation of blossom values of polynomial $(\mathbf{x}[i])^h(\mathbf{x}[j])^k$.

$$\beta^d_{h,k}(\mathbf{u}_1,\ldots,\mathbf{u}_d) = \frac{1}{\binom{d}{h}\binom{d-h}{k}} \sum_{\substack{I\cup J\subseteq\{1,\ldots,d\},\\ |I|=h,|J|=k,I\cap J=\emptyset}} \prod_{r\in I}\mathbf{u}_r[i]\prod_{s\in J}\mathbf{u}_s[j]$$

Denote $\sigma^d_{h,k} = \frac{1}{\binom{d}{h}\binom{d-h}{k}}\beta^d_{h,k}(\mathbf{u}_1,\mathbf{u}_2,\ldots,\mathbf{u}_d)$.

The symmetric function σ: choose h i^{th} coordinates of d argument points and k j^{th} coordinates and form their product, then sum these products over all possible choices.

$$\begin{cases} \sigma^d_{h,k} &= \sigma^{d-1}_{h,k} + \mathbf{u}_d[i]\sigma^{d-1}_{h-1,k} + \mathbf{u}_d[j]\sigma^{d-1}_{h,k-1} \text{ if } h,k\geq 0 \text{ and } h+k\geq 1,\\ \sigma^d_{0,0} &= 0 \end{cases}$$

Complexity $O(d^3)$

Error bound and subdivision

Error in the approximation of the polynomial map $\boldsymbol{\pi}$ by its Bézier control points is $\mathcal{O}(\rho^2)$ (ρ =max side length).

When Δ is large $\Rightarrow$ subdivide it into smaller simplices, which creates new Bézier bases and therefore new control points.

Computation of the new control nets that reuses the results obtained for the original simplex:

1. Partition the simplex Δ by adding a point $\mathbf{x} \in \Delta$ and forming new smaller simplices.
2. We can use *de Casteljau* algorithm to evaluate $\boldsymbol{\pi}(\mathbf{x})$ and this computation also produces the control net for the new simplices.

Reachability algorithm

$R_0 = X_0$, $k = 0$
REPEAT
  $S_\Delta = triangulation(vertices(R_k))$
  $C = \emptyset$
  FORALL$(\Delta \in S_\Delta)$
    $C = C \cup Bez(\Delta)$
  ENDFOR
  $R_{k+1} = conv(C)$
  $k = k + 1$
UNTIL$(R_{k+1} = R_k)$

Bez over-approximates $P(\Delta)$, P is the integration map

Bacteria Vibrio Fisheri

$$\begin{cases} \dot{x_1} &= k_2 x_2 - k_1 x_1 x_3 + u_1 \\ \dot{x_2} &= k_1 x_1 x_3 - k_2 x_2 \\ \dot{x_3} &= k_2 x_2 - k_1 x_1 x_3 - n x_3 + n u_2 \end{cases}$$

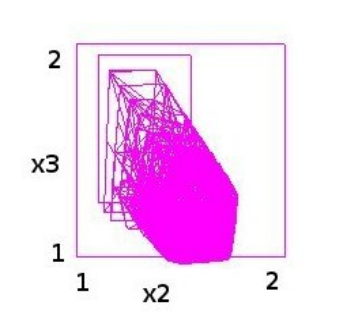
State variables (x_1, x_2, x_3): cellular concentrations of different species.

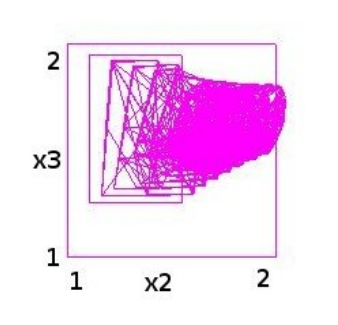
Parameters k_1, k_2, n: binding, dissociation and diffusion constants.

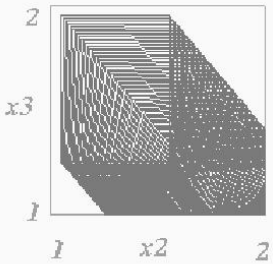
Control objective: steering the system to the face $x_2 = 2$ (activation of some genes).

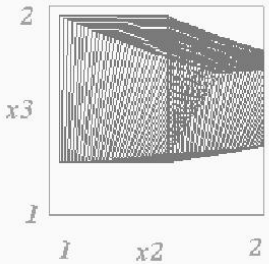
Control laws: $u_1(\mathbf{x}) = -10(x_2 + x_1(-1+3) - 4x_3)$ and $u_2(\mathbf{x}) = x_1(3 + x_2(-1+x_3)) - (-2+x_2)x_3$.

Bacteria Vibrio Fisheri









The results are more precise than those obtained using abstraction by projection [Asarin Dang 03].

Bacteria Vibrio Fisheri (cont'd)

Hybrid model with two modes and one additional continuous variable x_4. The continuous dynamics is $\dot{\mathbf{x}} = A\mathbf{x} + g(\mathbf{x}) + b_{ij}$ where b_{01} and b_{10} correspond respectively to the non-luminescent and luminescent modes

$$A = \begin{pmatrix} \frac{-1}{H_{sp}} & 0 & 0 & r_{Co} \\ 0 & 0 & 0 & \frac{-1}{H_{sp}} - r_{Co} \\ 0 & x_0 r_{AII} & \frac{-1}{H_{AI}} & x_0 r_{Co} \\ 0 & \frac{-1}{H_{sp}} & 0 & 0 \end{pmatrix} ; \; g(\mathbf{x}) = \begin{pmatrix} -1 \\ 1 \\ -x_0 \\ 0 \end{pmatrix} r_{AIR} x_1 x_3$$

Question: determine the sets of states from which the system can reach the luminescent equilibrium. The condition for switching between the two modes is $x_2 = x_{2sw}$.

Qualitative results compatible with the previous study using d/dt (on a linearized model by fixing x_1 constant),

Larger set of states that can reach the equilibrium.

Concluding remarks

Combining the idea of set integration with techniques from CAGD

We achieved a second order approximation method

Higher order methods can be derived (but the resulting polynomials are more complex, i.e. more monomial terms)

Future directions: using splines??

Questions?? and thank you.

Subdivision

$$\mathbf{p}_{\mathbf{i}}^{l} = \beta(\underbrace{\mathbf{v}_1, \ldots, \mathbf{v}_1}_{\mathbf{i}[1]}, \ldots, \underbrace{\mathbf{v}_{n+1}, \ldots, \mathbf{v}_{n+1}}_{\mathbf{i}[n+1]}, \underbrace{\mathbf{x}_1, \ldots, \mathbf{x}_l}_{l})$$

$$\mathbf{p}_{\mathbf{i}}^{l} = \lambda_1(\mathbf{x}_l)\mathbf{p}_{\mathbf{i}+\mathbf{e}_1}^{l-1} + \ldots + \lambda_n(\mathbf{x}_l)\mathbf{p}_{\mathbf{i}+\mathbf{e}_n}^{l-1}, \quad \mathbf{i}[1] + \ldots + \mathbf{i}[n+1] + l = d$$

where $\mathbf{e}_1 = (1, 0 \ldots, 0)$, $\mathbf{o} = (0, 0 \ldots, 0)$. Note that $\mathbf{p}_{\mathbf{i}}^0$ are the Bézier control points, and $\mathbf{p}_{\mathbf{o}}^d = \beta(\mathbf{x}_1, \ldots, \mathbf{x}_d)$.
We run the algorithm from $l = 0$ to $l = d$ with all $\mathbf{x}_l = \mathbf{x}$ to obtain $\mathbf{p}_{\mathbf{o}}^d = \beta(\mathbf{x}, \ldots, \mathbf{x})$. In the following example, $\mathbf{p}_{(2,0,0)}^1 = \lambda_1(\mathbf{x})\mathbf{p}_{(3,0,0)}^0 + \lambda_2(\mathbf{x})\mathbf{p}_{(2,1,0)}^0 + \lambda_3(\mathbf{x})\mathbf{p}_{(2,0,1)}^0$