MANCHESTER
1824

The University of Manchester

CICADA Collection

Chahlaoui, Younes and Korovina, Margarita

2014

MIMS EPrint: **2014.24**

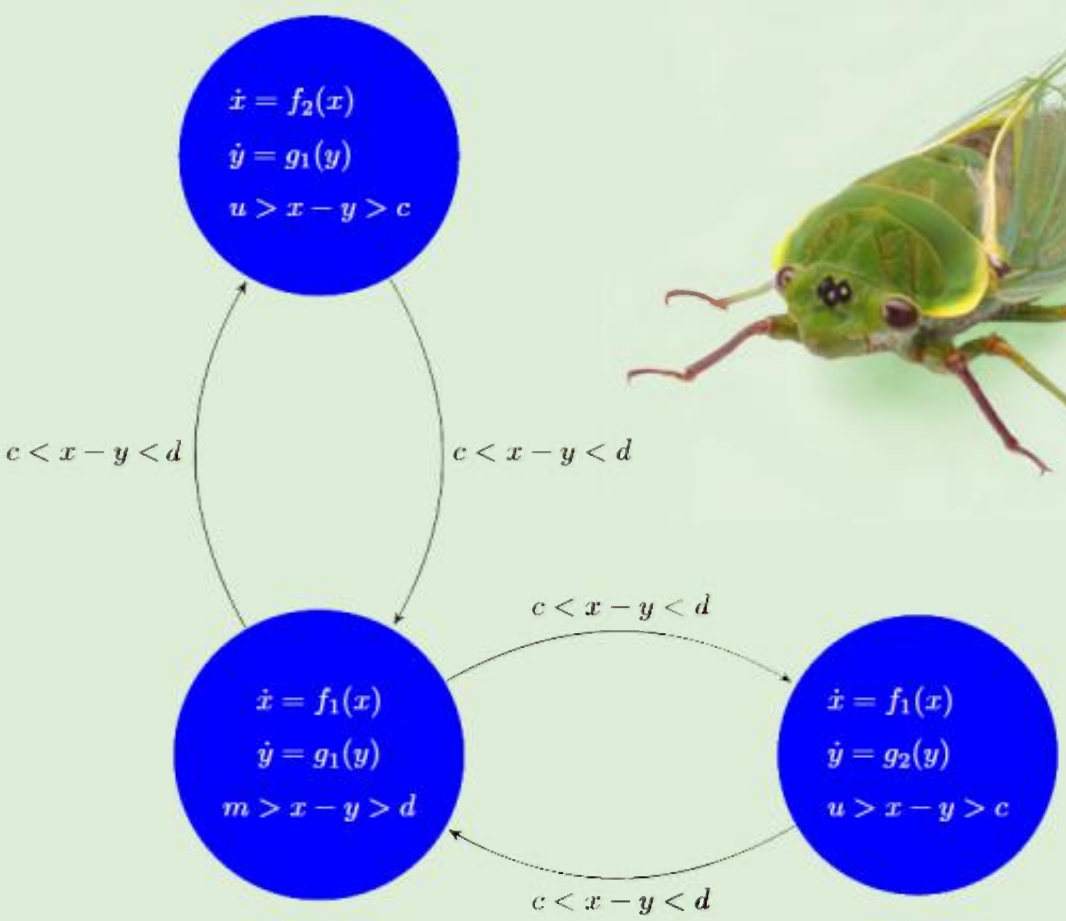
Manchester Institute for Mathematical Sciences
School of Mathematics

The University of Manchester

Reports available from: http://eprints.maths.manchester.ac.uk/
And by contacting: The MIMS Secretary
School of Mathematics
The University of Manchester
Manchester, M13 9PL, UK

ISSN 1749-9097

CICADA COLLECTION



CICADA Collection*

Edited by Y. Chahlaoui and M. Korovina

*Centre for Interdisciplinary Computational and Dynamical Analysis (CICADA) School of Mathematics, The University of Manchester, Manchester, M13 9PL, UK. This work was supported by Engineering and Physical Sciences Research Council grant EP/E050441/1

Preface

This volume is CICADA Collection which contains contributions developed by researches working on CICADA Project, The University of Manchester. CICADA Project creates a warm and fruitful atmosphere for research collaboration in many aries of Mathematics, Computer Science, Engineering including hybrid and dynamical systems, verification of safety critical systems, human robotics, model reduction and high dimensional systems, max-pus algebra, stochastic hybrid systems, analysis of adaptive systems and control. This volume presents examples and software which have been developed and used by CICADA community.

Manchester, 2012

Younes Chahlaoui & Margarita Korovina

This book is dedicated to David Broomhead, an outstanding person and researcher.

Contents

1

Introduction

1 Hybrid automaton

Hybrid automata were proposed as a formal model for Hybrid Systems in [1]. A Hybrid system is a dynamical system with both discrete and continuous components.

A hybrid automaton $HA = (Q, C, Init, D, E, G, R)$ consists of

- A finite discrete state space: Q;
- A continuous state space: $C \subseteq \mathbb{R}^n$;
- The hybrid automaton state space: $X = Q \times C$;
- The set of initial states: $X_0 \subseteq X$;
- Continuous flow: $\dot{x} = F(l,x)$ for each discrete state l;
- Invariants: $Inv(l) \subseteq \mathbb{R}^n$ for each discrete state l;
- Discontinuous changes $R \subseteq X \times X$

We define discrete transitions, guards, resets as follows:

$$
\begin{aligned}
&E = \{(l,\acute{l}) | \exists x \in Inv(l) \exists \acute{x}\ ((l,x),(\acute{l},\acute{x})) \in R\};\\
&Init(l) = \{x \in Inv(x) | (l,x) \in X_0\};\\
&Guard(e) = \{x \in Inv(l) | \exists \acute{x} \in Inv(\acute{l})((l,x),(\acute{l},\acute{x}) \in R\};\\
&Reset(e,x) = \{\acute{x} \in Inv(\acute{l}) | ((l,x),(\acute{l},\acute{x})) \in R\}.
\end{aligned}
$$

Properties of hybrid automata.

- A hybrid automaton it is non-blocking if from any initial condition there exists at least one state trajectory (solution));
- An invariant of the hybrid automaton is a property that is always true.
- A hybrid automaton is Zeno if takes an infinite number of discrete transitions in finite time.
- A hybrid automaton is non-deterministic if for a given initial condition there are a whole family of state trajectories (solutions).

References

[1] T. A. Henzinger, *The Theory of Hybrid Automata*, in Verification of Digital and Hybrid Systems, edited by M. K. Inan, and R. P. Kurshan, NATO ASI Series F: Computer and Systems Sciences, Springer-Verlag, v.170, pp.265-292, 2000.

2 Stochastic Hybrid System

A stochastic hybrid system (SHS) is a collection

$$H := ((Q,d,\mathcal{X}),b,\sigma,Init,\lambda,R)$$

where
• Q is a countable/finite set of discrete states (modes);
• $d : Q \to \mathbb{N}$ is a map giving the dimensions of the continuous state invariants;
• $\mathcal{X} : Q \to \mathbb{R}^{d(.)}$ maps each $q \in Q$ into an open subset X^q of $\mathbb{R}^{d(q)}$;
• $b : X(Q,d,\mathcal{X}) \to \mathbb{R}^{d(.)}$ is a vector field;
• $\sigma : X(Q,d,\mathcal{X}) \to \mathbb{R}^{d(\cdot)\times m}$ is a $X^{(\cdot)}$-valued matrix, $m \in \mathbb{N}$ (more generally, m may also depend on the discrete mode),
• $Init : \mathcal{B}(X) \to [0,1]$ is an initial probability measure on $(X, \mathcal{B}(X))$;
• $\lambda : \overline{X}(Q,d,\mathcal{X}) \to \mathbb{R}^+$ is a transition rate function;
• $R : \overline{X} \times \mathcal{B}(\overline{X}) \to [0,1]$ is a stochastic kernel.

Let us denote by X the whole space, i.e.

$$X = \cup\{(q,X^q)|q \in Q\}.$$

Define the boundary set $\partial X^q := \overline{X^q} \backslash X^q$ of X^q and the whole space boundary

$$\partial X = \cup\{(q,\partial X^q)|q \in Q\}.$$

The executions of an SHS form a stochastic process, called *stochastic hybrid process*, which is built as a *Markov string*. This is obtained by the concatenation of a sequence of diffusion processes (z_t^i), $i \in Q$.

A **stochastic hybrid (SH) process** is a stochastic process

$$x_t = (q(t), z(t))$$

such that there exists a sequence of stopping times

$$T_0 = 0 < T_1 < T_2 < \ldots$$

such that for each $k \in \mathbb{N}$,
• $x_0 = (q_0, z_0^{q_0})$ is a $Q \times X$-valued random variable extracted according to the probability measure $Init$;
• For $t \in [T_k, T_{k+1})$, $q_t = q_{T_k}$ is constant and $z(t)$ is a solution of the stochastic differential equation (SDE):

$$dz(t) = b(q_{T_k}, z(t))dt + \sigma(q_{T_k}, z(t))dW_t \tag{1.1}$$

where W_t is the m-dimensional standard Wiener process (m might depend on q_{T_k});
• $T_{k+1} = T_k + S^{i_k}$ where S^{i_k} is a stopping time
• The probability distribution of $x(T_{k+1})$ is governed by the law $R\left((q_{T_k}, z(T_{k+1}^-)), \cdot\right)$.

3 Model reduction

We consider three types of dynamical models:

$$\text{(i)} \quad \left\{ \begin{array}{rcl} \delta x(\cdot) & = & G(x(\cdot),u(\cdot)) \\ y(\cdot) & = & H(x(\cdot),u(\cdot)) \end{array} \right. \qquad \text{where} \quad x \in \mathbb{R}^N, u \in \mathbb{R}^m, y \in \mathbb{R}^p$$

$$\Downarrow \text{ linearize}$$

$$\text{(ii)} \quad \left\{ \begin{array}{rcl} E_\sigma \delta x(\cdot) & = & A_\sigma x(\cdot) + B_\sigma u(\cdot) \\ y(\cdot) & = & C_\sigma x(\cdot) \end{array} \right. \qquad \begin{array}{l} E, A \in \mathbb{R}^{N\times N}, B \in \mathbb{R}^{N\times m}, \\ C \in \mathbb{R}^{p\times N} \end{array} \tag{1.2}$$

$$\text{(iii)} \quad \left\{ \begin{array}{rcl} M\left[\delta^2 q(\cdot)\right] + D\left[\delta q(\cdot)\right] + K\, q(\cdot) & = & F\, u(\cdot) \\ y(\cdot) & = & G\, q(\cdot) \end{array} \right. \qquad \begin{array}{l} q \in \mathbb{R}^N, M, D, K \in \mathbb{R}^{N\times N}, \\ F \in \mathbb{R}^{N\times m}, G \in \mathbb{R}^{p\times N} \end{array}$$

where δ is either the derivative operator $\delta x(t) = \dot{x}(t)$, $t \in \mathbb{R}$, or the shift $\delta x(k) = x(k+1)$, $k \in \mathbb{Z}$, depending on whether the system is continuous- or discrete-time[1] The logical law σ is the switching law.

Here, (i) is a class of dynamical systems that are finite-dimensional and described by a set of explicit first-order differential equations; the description is completed with a set of observation variables. Very often, this explicit nonlinear system is linearized around some equilibrium trajectory, the resulting system (ii) being linear, time-varying or time-invariant or switched. Finally, (iii) describes second-order systems. Often the physical origin of these systems implies that the matrices M, D and K are symmetric and moreover M is positive (semi-)definite and invertible. For mechanical systems the matrices M, D and K represent, respectively, the *inertia*, *damping* and *stiffness* matrices, u corresponds to the vector of *external* forces, F is the input distribution matrix, y is the output measurement vector, G is the output measurement matrix, and $q(\cdot)$ to the vector of *internal generalized coordinates*. Discrete systems can be treated equally well as the continuous case.

For linear time-invariant systems (1.2)(ii) and second-order systems (1.2)(iii), we can define the transfer function relating directly the outputs to the inputs

$$T_f(s) = C(sE - A)^{-1}B + D \quad \text{or} \quad T_f(s) = G(s^2M + SD + K)^{-1}F. \tag{1.3}$$

The goal of model reduction is to construct much simpler system that will have similar outputs to those of the original system for the same inputs. Different methods have been developed but there is no general technique for model reduction that can be considered as optimal in an overall sense since the reliability, performance and adequacy of the reduced system strongly depends on the system characteristics. Model reduction methods usually differ in the error measure they attempt to minimize.

The main idea is that a high-dimensional state vector is actually belongs to a low-dimensional subspace. Provided that the low-dimensional subspace is known, the ordinary differential equations can be projected on it. The projection gives us a required low-dimensional approximation.

Available methods for linear ODEs are listed in the table below (according to [1]).

[1] Notice here that the dot $\cdot$ is replaced in the continuous case by t and in discrete case by k.

Method	Advantages	Disadvantages
SVD-based (Truncated Balanced Approximation, Singular Perturbation Approximation, Hankel-Norm Approximation)	Have a global error estimate, can be used in a fully automatic manner	Computational complexity of conventional implementations is $O(N^3)$, can be used for systems with order less than a few thousand unknowns only
Low-rank Gramian approximants and matrix sign function method	Have a global error estimate and the computational complexity is acceptable	Currently under development
Pade approximants (moment matching) via Krylov subspaces by means of either the Arnoldi or Lanczos process	Very advantageous computationally, can be applied to very high-dimensional linear systems	Does not have a global error estimate. It is necessary to select the order of the reduced system manually or with some engineering tricks

Systems of the form (1.2)(iii), in principle, can be always transformed to the first order and then model reduction methods for the first order system can be applied. However, the reduced system in this case is obtained as the first order ODEs and this is not always desirable.

In the special case of proportional damping ($D = \alpha M + \beta K$), it is possible to ignore the damping matrix when the projection subspace is constructed but then to restore the reduced damping matrix from the reduced mass and stiffness matrices. The most interesting is that moment matching happens automatically for any values of α and β, that is, in a way, we have parametric model reduction in respect to α and β for free, without any extra efforts. When damping is not proportional, one can derive second order Krylov subspaces in order to perform model reduction directly for the second order system. This happens to have many advantages as compared with the transformation to the first order system [1, 2, 3, 4].

Nonlinear model reduction in the general case is a challenge. The most popular method is Proper Orthogonal Decomposition. It can be generalized by means of empirical Gramians. There is a generalization of balancing for a nonlinear model. An interesting new approach is trajectory piecewise model reduction.

References

[1] A. C. Antoulas, *Approximation of Large-Scale Dynamical Systems*, SIAM, 2005.

[2] P. Benner and V. Mehrmann and D. C. Sorensen, *Dimension Reduction of Large-Scale Systems*, Lecture Notes in Computational Science and Engineering, Vol.45, Springer, 2005.

[3] W. H. A. Schilders and H. A. van der Vorst and J. Rommes, *Model Order Reduction: Theory, Research Aspects and Applications*, Mathematics in Industry, Vol.13, Springer, 2008.

[4] P. Benner and M. Hinze and E. J ter Maten, *Model Reduction for Circuit Simulation*, Lecture Notes in Electrical Engineering, Vol.74, Springer, 2011.

2

Collection of Problems

This section contains a brief description of all the problems in the collection. We give below a list of these problems

Contents

BOUNCING BALL

Properties

Zeno behavior, physical system with impact.

By Y.Chahlaoui & M.Korovina

Description

The bouncing ball is an especially interesting hybrid system, as it exhibits Zeno behavior. It is the most canonical example of a hybrid system. It is a physical system with impact. Here, the ball (thought of as a point-mass) is dropped from an initial height and bounces off the ground, dissipating its energy with each bounce. The ball exhibits continuous dynamics between each bounce; however, as the ball impacts the ground, its velocity undergoes a discrete change modeled after an inelastic collision.

Formalization

A model for a bouncing ball can be represented as a simple hybrid system (Figure) with single discrete state and a continuous state of dimension two

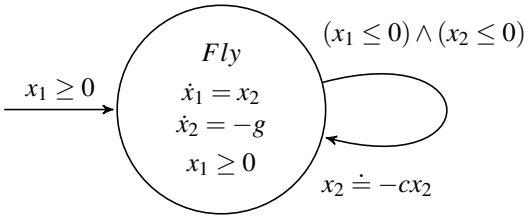
$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix},$$

where x_1 denotes the vertical position of the ball and x_2 its vertical velocity.

The continuous motion of the ball is governed by Newton's laws of motion. This is indicated by the differential equation that appears in the vertex (box), where g denotes the gravitational acceleration. This differential equation is only valid as long as $x_1 \geq 0$, i.e., as long as the ball is above the ground. This is indicated by the logical expression $x_1 \geq 0$ that appears in the vertex below the differential equation.

The ball bounces when $x_1 = 0$ and $x_2 \leq 0$. This is indicated by the logical expression that appears near the beginning of the edge (arrow). At each bounce, the ball loses a fraction of its energy. This is indicated by the equation $x_2 := -cx_2$ (with $c \in [0,1]$) that appears near the end of the edge. This is an assignment statement, which means that after the bounce the speed of the ball will be c times the speed of the ball before the bounce, and in the opposite direction.

Starting at an initial state with $x_1 \geq 0$ (as indicated by the logical condition next to the arrow pointing to the vertex), the continuous state flows according to the differential equation as long as the condition $x_1 \geq 0$ is fulfilled. When $x_1 = 0$ and $x_2 \leq 0$, a discrete transition takes place and the continuous state is reset to $x_2 := -cx_2$ (x_1 remains constant). Subsequently, the state resumes flowing according to the vector field, and so on.



Softwares and references

[1] V. Acary and B. Brogliato, *Numerical Methods for Nonsmooth Dynamical Systems*, Lecture Notes in Applied and Computational Mechanics 35, 2008.

THERMOSTAT

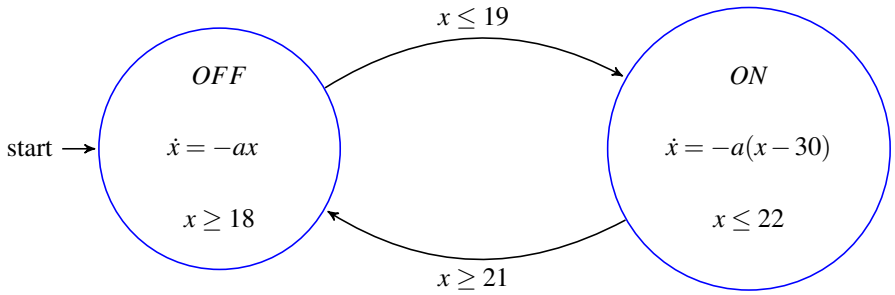
Properties

By M. Korovina

The hybrid automaton Th is non-blocking, non-deterministic, not Zeno.

Description

Consider a room being heated by a radiator controlled by a thermostat. Assume that the thermostat is trying to keep the temperature at around 20 degrees. The hybrid automaton of Figure 1 models a thermostat. The variable x represents the temperature. In control mode OFF, the heater is off, and the temperature falls according to the flow condition $\dot{x} = -ax$. In control mode ON, the heater is on, and the temperature rises according to the ow condition $\dot{x} = -a(x-30)$. Initially, the heater is ON and the temperature is 20 degrees. According to the jump condition $x < 19$, the heater may go on as soon as the temperature falls below 19 degrees. According to the invariant condition $x > 18$, at the latest the heater will go on when the temperature falls to 18 degrees.



Formalization

The hybrid automaton of the thermostat is [1] $Th = (Q, X, Init, D, E, G, R)$
Discrete state space: $Q = \{OFF, ON\}$;
Continuous state space: $X = \mathbb{R}$, where x is the room temperature;
Discrete transitions: $E = \{(OFF, ON), (ON, OFF)\}$;
Continuous flow: $f(OFF, x) = -ax$, $f(ON, x) = -a(x-30)$;
Invariants: $D(OFF) = \{x \in \mathbb{R} | x \geq 18\}$, $D(ON) = \{x \in \mathbb{R} | x \leq 22\}$;
Guards: $G(OFF, ON) = \{x \in \mathbb{R} | x \leq 19\}$, $G(ON, OFF) = \{x \in \mathbb{R} | x \geq 21\}$;
Resets: $R(OFF, ON, x) = R(ON, OFF, x) = \{x\}$.

Software and references

[1] T. A. Henzinger, *The Theory of Hybrid Automata*, 11th Annual Symposium on Logic in Computer Science (LICS), IEEE Computer Society Press, pp. 278–292, 1996.

DIGITALLY CONTROLLED THERMOSTAT MODEL

Properties

By P. Kowalczyk

Depending on the system parameters we might encounter a family of periodic orbits, quasi-periodic oscillations or a banding structure of quasi-periodicity. When the system evolution is governed by generic non-linear functions a fixed point attractor and chaotic dynamics are present in this more general case. Further details of the dynamics of the presented example can be found in [3]. Some other relevant papers are [1],[2].

Description

Simple models of thermostat control are frequently used as illustrations of hybrid systems, as it was shown in the previous example. The control system is described as hybrid because it couples a continuous state variable for the temperature with a discrete variable (on/off) for the state of the control system. Here we consider another example of a thermostat model with

$$\frac{dh}{dt} = \begin{cases} r_1 & \text{if } c = \texttt{on} \\ -r_2 & \text{if } c = \texttt{off}, \end{cases} \tag{2.1}$$

where r_i, $i = 1,2$ are positive constants representing the rate of heating or cooling of the room. The state of c changes from `on` to `off` if h rises above a threshold $h_{\ 2}$ and from `off` to `on` if h falls below $h_{\ 1}$, with $h_1 < h_2$. However, unlike in the previous example, assume that the control monitors the ambient temperature at regular but discrete times, $n\tau$, and immediately switches the control if either of the threshold inequalities $h < h_1$ or $h > h_2$ are satisfied at that time. Although this is not of immediate relevance to thermostats, one can imagine situations where continuous monitoring of a variable cannot be done either for technical reasons, or for reasons of economy, and the thermostat model provides a simple and well understood example of the ways in which the control operation can be changed under these circumstances.

Formalization

In the current case, for the analytical purposes, it is convenient to represent the system using return maps.

If the temperature evolves according to (2.1) the control variable in the new model described above switches from `on` to `off` at $n\tau$ if $c(n\tau) = \texttt{on}$ and $h(n\tau) > h_{\ 2}$, in which case $c(n\tau_+) = \texttt{off}$ (where $c(s_+) = \lim_{t \downarrow s}(t)$) and similarly it switches from `off` to `on` at $n\tau$ if $c = \texttt{off}$ and $h(n\tau) < h_{\ 1}$, in which case $c(n\tau_+) = \texttt{on}$. We have used strict inequalities for mathematical convenience. Once operational the temperature is restricted to the interval $[h_1 - r_2\tau, h_2 + r_1\tau]$ so we can choose the origin of time so that $h(0) \in [h_1 - r_2\tau, h_1)$, $c(0) = \texttt{off}$ and $c(0_+) = \texttt{on}$. The heater will remain on until $t = N_1\tau$ where

$$h(N_1\tau) = h(0) + N_1 r_1 \tau > h_2, \;\; h(0) + (N_1 - 1) r_1 \tau \leq h_2. \tag{2.2}$$

It will then switch off and the room will cool until $t = N_2\tau$ where

$$h((N_1 + N_2)\tau) = h(N_1\tau) - N_2 r_2 \tau < h_1, \;\; h(N_1\tau) - (N_2 - 1) r_2 \tau \geq h_1. \tag{2.3}$$

Rearranging the two inequalities of (2.2) gives

$$N_1 - 1 \leq \frac{h_2 - h(0)}{r_1\tau}, \quad N_1 > \frac{h_2 - h(0)}{r_1\tau}, \tag{2.4}$$

or

$$N_1 = \lfloor \frac{h_2 - h(0)}{r_1\tau} \rfloor + 1 = \lfloor \frac{h_2 + r_1\tau - h(0)}{r_1\tau} \rfloor, \tag{2.5}$$

where $\lfloor x \rfloor$ is the floor function, the largest integer less than or equal to x (this is where there is some convenience in the choice of strict inequality referred to in the opening paragraph of this section). Similarly

$$N_2 = \lfloor \frac{h(N_1\tau) - h_1}{r_2\tau} \rfloor + 1 = \lfloor \frac{h(N_1\tau) + r_2\tau - h_1}{r_2\tau} \rfloor. \tag{2.6}$$

Putting these together with (2.2) and (2.3) gives the simplified equations

$$\begin{aligned} h(N_1\tau) &= h(0) + r_1\tau \lfloor \frac{h_2 + r_1\tau - h(0)}{r_1\tau} \rfloor, \\ h((N_1+N_2)\tau) &= h(N_1\tau) - r_2\tau \lfloor \frac{h(N_1\tau) + r_2\tau - h_1}{r_2\tau} \rfloor. \end{aligned} \tag{2.7}$$

Since $h((N_1+N_2)\tau) \in [h_1 - r_2\tau, h_1)$ as was $h(0)$ we can now iterate this process to obtain the sequence $h_0 = h(0), h_1 = h(N_1\tau), \ldots$ of values of the temperature at switching points:

$$\begin{aligned} h_{2n+1} &= h_{2n} + r_1\tau \lfloor \frac{h_2 + r_1\tau - h_{2n}}{r_1\tau} \rfloor, \\ h_{2n+2} &= h_{2n+1} - r_2\tau \lfloor \frac{h_{2n+1} + r_2\tau - h_1}{r_2\tau} \rfloor, \end{aligned} \tag{2.8}$$

with $h_{2n} \in [h_1 - r_2\tau, h_1)$ and $h_{2n+1} \in (h_2, h_2 + r_1\tau]$. Composing these we get a single equation for even switching levels as a function of the previous even switching temperature. These equations are now conventional difference equations, albeit involving the floor function, as opposed to the hybrid system we began with, and to understand the dynamics of the original flow we have to analyse the dynamics of the discrete map (2.8).

Softwares and references

[1] G. Haller and G. Stépán, *Micro-chaos in digital control*, Journal of Nonlinear Science, 6:415–448, 1996.

[2] E. Enikov and G. Stépán, *Micro-chaotic motion of digitally controlled machines*, Journal of Vibration and control, 4:427–443, 1998.

[3] P. Glendinning and P. Kowalczyk, *Dynamics of a hybrid thermostat model with discrete sampling time control*, Dynamical Systems, 24(3):343–360, 2009.

COMPASS GAIT ROBOT WALKING

Properties

By H. Dallali & M. Brown

The problem of developing flexible, robust and energy efficient control strategies for humanoid robotic walking is currently receiving a lot of attention. A key part of this recent work is this idea of passive dynamics, where the basic robot is designed to walk as smoothly as possible, without requiring control actuation. Probably the simplest model such is the compass gait robot [1].

Description

The basic compass gait robot is a 2 link model, where each link represents a leg 2.1. There is no upper body. The stance "foot" remains stationary and the stance leg pivots about that point. Power is provided by gravity and the swing foot/leg rotates until a ground impact is detected. The double support phase is assumed to happen instantaneously and the swing and support legs are swapped and a reduction in the angular velocity occurs due to the impact. Stable limit cycle walking occurs when the energy gained during the swing phase is equal to the energy lost at the impact.

Figure 2.1: Compass Gait.

where m and M are the leg and waist masses, respectively $l(=a+b)$ is the leg length where a and b are the distances of the leg mass from the leg tip and hip, respectively, ϕ is the slope angle and θ_s and θ_{ns} are the stance and swing leg angles, respectively.

For this basic model, the swing dynamics are described by:

$$M_n(\theta)\ddot{\theta}+N_n(\theta,\dot{\theta})\dot{\theta}+g_n(\theta)/a=0$$

where the angular position $\theta=[\theta_s\theta_{ns}]$ and angular velocity and the matrices/vector M_n, N_n, g_n are given by:

$$M_n(\theta)=\begin{bmatrix} \beta^2 & -(1+\beta)\beta\cos(2\alpha) \\ -(1+\beta)\beta\cos(2\alpha) & (\mu+1)(1+\beta)^2+1 \end{bmatrix}$$

$$N_n(\theta,\dot{\theta})=\begin{bmatrix} 0 & (1+\beta)\beta\dot{\theta}_s\sin(\theta_s-\theta_{ns}) \\ -(1+\beta)\beta\dot{\theta}_{ns}\sin(\theta_s-\theta_{ns}) & 0 \end{bmatrix}$$

$$g_n(\theta)=\begin{bmatrix} g\beta\sin(\theta_{ns}) \\ -((\mu+1)(1+\beta)+1)g\sin(\theta_s) \end{bmatrix}$$

This can be represented in state space form:

$$\dot{x} = \frac{d}{dt}\begin{pmatrix} \theta \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \dot{\theta} \\ -M_n^{-1}(\theta)[N_n(theta,\dot{\theta})\dot{\theta} + g_n(\theta)/a] \end{pmatrix}$$

where the state vector $x = [\theta_s \theta_{ns} \dot{\theta}_s \dot{\theta}_{ns}]^T$ is composed of the angular position and velocity of the stance and swing legs, respectively.
The reset map at impact time is given by:

$$x^+(t) = \begin{bmatrix} J & 0 \\ 0 & H(\alpha) \end{bmatrix} x^-$$

where x^- and x^+ are the states immediately before and after the impact, respectively and the reset matrices are given by:

$$\begin{aligned}
J &= \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \\
N_n(\alpha) &= Q_n^{+-1}(\alpha) Q_n^-(\alpha) \\
Q_n^-(\alpha) &= ma^2 \begin{bmatrix} -\beta & -\beta + (\mu(1+\beta)^2 + 2(1+\beta))\cos(2\alpha) \\ 0 & -\beta \end{bmatrix} \\
Q_n^+(\alpha) &= ma^2 \begin{bmatrix} \beta(\beta - (1+\beta)\cos(2\alpha)) & (1+\beta)((1+\beta-\beta\cos(2\alpha)) + 1 + \mu(1+\beta)^2 \\ \beta^2 & -\beta(1+\beta) + \cos(2\alpha) \end{bmatrix}
\end{aligned}$$

where $\alpha = (\theta_{ns} - \theta_s)/2$ is half the interleg angle. The J matrix simply swaps the swing and stance leg angles and the H matrix again swaps the angular velocities and also reduces them due to the energy loss at impact.
The impact event is detected when:

$$\cos(\theta_s + \phi) = \cos(\theta_{ns} + \phi)$$

The basin of attraction for stable limit cycles is relatively small and it is necessary to provide initial conditions close to the stable limit cycle. One set of values is given here. This basic compass gait robot has been extended in many ways to provide simple active control so that the robot can walk on flat or inclined surfaces (or so that the robot can walk at a different speed down the incline). Such actuation can occur at the waist, stance ankle or as an impulsive toe push off.

Softwares and references

[1] A. Goswami, B. Thuilot and B. Espiau, *A study of the passive gait of a compass-like biped robot: symmetry and chaos*, International Journal of Robotics Research, v.17, n.12, 1998.

[2] http://www.ambarish.com

STOCHASTIC HYBRID SYSTEM

Properties

By Manuela Bujorianu

This example presents a class of stochastic hybrid systems that occur in the modelling of the switching behaviour of electrical devices (electric space heaters, air conditioners), associated with some form of energy storage. Understanding their switching dynamics is important from different practical reasons.This model has been developed by R.P. Malahamé and C.Y. Chong in [1].

Description

Electrical devices connected to a power system are subjected to two sets of inputs:

- voltge and frequency on the power system side, and
- service demand and other relevant processes, such as atmospheric conditions in the case of weather sensitive devices (e.g. electric space heaters, air conditioners, etc) on the user side.

The effect of the first set of inputs is directly a function of the physical construction of the device as an energy converter. The effect of the second set of inputs is a function of the mechanism that controls the operating state of the device. This natural division of inputs and effects makes possible to view every electrical device associated with two distinct interconnected models:

- a device *response model* reacting to voltage and frequency for a given operating state; and
- a device *functional model* reacting to service demand and other processes.

Formalization

The output of the functional model is a discrete $q(t)$ scalar or vector representing the operating state of the device at time t.

It is assumed that the operating of an individual electric space heater is controlled by a thermostat, the state of which depends on the temperature of the dwelling. The complete state vector comprises a continuous part characterizing the evolution of the temperature and a discrete boolean part characterizing the operating state of the device. the model is as follows.

Continuous state

The continuous temperature state $x(t)$ evolves according to the following first order Itô stochastic differential equation based on a linearised energy balance analysis.

$$Cdx(t) = -a'(x(t) - X_a(t))dt + R'q(t)b(t)dt + \sigma' dv(t)$$

where:

- C - thermal capacity of the dwelling in joules/^{o} C
- a' - is the average thermal resistance of the floors, walls, ceiling of the dwelling in watts/^{o} C
- $X_a(t)$ - ambient in ^{o} C
- R' - power rating of the heating device in watts
- $q(t)$ - the operating state of the device (1 for on and 0 for off)

- $b(t)$ - the state of the boolean control signal sent by the utility (1 for no control and 0 for a centrally controlled interruption of the load)
- $\sigma' v$ - a Wiener process of intensity σ', simulating all unaccounted for processes of heat gain or heat loss (fluctuating number of people in the residence, doors, windows being opened and closed, refrigerators, cooking, etc).

The choice of the Wiener process as a noise model is justified by the fact that such processes can be viewed as collection of random independent thermal shocks, the duration of each is very small compared to the thermodynamic time constants of the dwelling.

Discrete state

The evolution of the discrete state $q(t)$ is governed by a thermostat with temperature setting x_- and a deadband $(x_+ - x_-)$. $q(t)$ switches from 1 to 0 when $x(t)$ reaches x_- and from 0 to 1 when $x(t)$ reaches x_+. No switching occurs otherwise.

Thus, the model is composed of two interconnected subsystems; a linear part characterised by a stochastic continuous state $x(t)$, the evolution which depends on $q(t)$, and a nonlinear part $q(t)$ with jumps depending on $x(t)$ deterministically.

The vector $\begin{bmatrix} x(t) \\ q(t) \end{bmatrix}$ is a hybrid state Markov process.

Softwares and references

[1] Malhame, R.P., Chong, C.Y.: *Stochastic Hybrid State Systems for Electric Load Modelling*. Proc. of IEEE Conference on Decision and Control (1983).

STATE CONSTRAINED STOCHASTIC REACHABILITY ANALYSIS

Description

Marius C. Bujorianu

For the definition of stochastic hybrid system (SHS) and stochastic hybrid process (SH) we refer to Introduction Under standard assumptions (about the diffusion coefficients, non-Zeno executions, transition measure, etc), any SH process is a strong Markov process. State-constrained reachability analysis denotes a reachability problem with additional conditions (constraints) on the system trajectories. Let us consider A, B two Borel measurable sets of the state space X with disjoint closures, i.e.

$$A, B \in \mathcal{B}(X) \text{ and } \overline{A} \cap \overline{B} = \emptyset.$$

We consider two fundamental situations. Suppose that the system paths start from a given initial state x and we are interested in a target state set, let say B. These trajectories can hit the state set A or not. Therefore, we may define two new concepts:

- Obstacle avoidance reachability. In this interpretation B is a safe set, whilst A is not. The goal is to compute the probability, denoted by
$$p^B_{\neg A}(x),$$
of all trajectories that start from a given initial state x and hit the set B without hitting the state set A (as illustrated in Fig.1).
- Waypoint reachability. In this interpretation we are interested to compute the probability, denoted by
$$p^B_A(x),$$
of all trajectories that hit B only after hitting A (as illustrated in Fig.2).

The connection between the two types of stochastic reachability is given by the formula

$$p^B_{\neg A}(x) + p^B_A(x) = \varphi_B(x)$$

where φ_B is the reachability function for the target set B. Therefore, the computations of the probabilities corresponding to the two types of reachability are equivalent. To have an easy notation, it is more convenient to work with the waypoint reachability, which will be called from now on just simply *state-constrained reachability*.

Figure 2.2: Obstacle avoidance reachability

Now we consider the executions (paths) of the stochastic hybrid process that start in $x = (q, z) \in X$. When we investigate the state-constrained reachability, we ask the probability that these trajectories visit A before visiting eventually B. Mathematically, this is the probability of

$$\{\omega | x_t(\omega) \notin B, \forall t \leq T_A\}.$$

Moreover, using the first hitting time T_B of B, we are interested to compute

$$p^B_A(x) = P_x[T_A < T_B]. \tag{2.9}$$

Figure 2.3: Waypoint reachability

An autonomous system has to determine the safest path for a vehicle to move through the area of operation to accomplish a given mission.

Let us suppose the dynamics of an autonomous system (an agent), in different operational modes, can be described by a hybrid system. For example, an UAV dynamics can be described by a hybrid system that makes discrete transitions whenever it reaches an waypoint. Considering different randomness factors (environment, noisy measurements, communication failures, etc), we may use a randomized version of the given hybrid system. Of course, the agent dynamics can be thought of as a particularization of the SHS general model. For example, one can easily imagine that the agent has no proper jumps, but only switchings from one continuous path to another. This randomized model of hybrid system will play a reference role. It will describe the desired dynamics. The description of this hybrid system should come with a fairly good representation of its state space. That means we need to have the locations of threats, obstacles, and restricted fly areas.

Formally, suppose that the agent dynamics can be described by a stochastic hybrid process M with the state space X. The obstacle is represented by a measurable set A, and the goal is given by a measurable set B. Therefore, we can use as uncertainty representation of X the following family of probability distributions

$$\{p_A^B(x)|x \in X\} \tag{2.10}$$

where $p_A^B(x)$ is the solution of the Dirichlet boundary value problem. The aim of the trajectory design problem for this agent would be to find "trajectories" in the space (2.10) with low collision probability. Then, on this space, we can define an harmonic potential field (HPF) method that aims to identify the paths with the lowest obstacle collision probabilities. In this setting, the potential field equations will be

$$\begin{cases} \nabla\{(1-p_A^B(x))\nabla V(x)\} = 0 & x \in X \backslash (A \cup B) \\ V(x) = 1 & x \in A \\ V(x) = 0 & x \in B. \end{cases} \tag{2.11}$$

A provably-correct path may be generated using the gradient dynamical system:

$$\dot{x} = -\nabla V(x).$$

The modified differential operator in (2.11) will expand in

$$(1-p_A^B(x))\nabla^2 V(x) + \nabla(1-p_A^B(x))\nabla V(x) = 0$$

which leads to

$$\nabla^2 V(x) = -\frac{1}{(1-p_A^B(x))}[-\nabla(1-p_A^B(x))][-\nabla V(x)]$$

Notice that $-\nabla V(x)$ is the direction at which motion is to be driven and $-\nabla(1-p_A^B(x)) = \nabla p_A^B(x)$ is a vector pointing in the direction of increasing risk.

The trajectories are generated using sampling. At a given state, the direction of moving is chosen by picking a state in the neighborhood with the smallest associated probability. In practice, the probabilities will not be computed for each state in the neighborhood, but only for a finite set of states chosen according

to a sampling policy. For example, a sampling policy in the plane would be to construct a circle in a suitable metric and pick a point on this circle. The remaining states are obtained by picking new points for each 45 degrees. Computationally, eight probabilities will be computed at each step. This sampling method could be made adaptive by considering the values of the probabilities on samples. If these values are small, fewer samples will be necessary in the next step. Contrary, for high values of the probabilities, more samples will help to find quicker the way down the ridge. The path generated by (2.11) will avoid the states from where the agent with a.s. probability 1 will collide with obstacle, i.e. the state for which the $p_A^B(x) = 1$.

In Figure 3. uncertainty representation of the state space corresponds to the graph of the harmonic function Re $u(x,y)$ where

$$u(x,y) = \frac{e^{-(x+yI)^2}}{(x+yI)^2} + (x+yI)^4$$

and x, y vary within the interval $[-2,2]$.

Figure 2.4: Trajectory in the uncertainty representation

The system trajectory is depicted as a thick curve. It can be observed that the trajectory follows a principle of getting quicker down the ridges onto the "valleys".

The uncertainty representation has an extra-dimension when compared to the state space. The generated trajectory will have also an extra-dimension. The trajectory will carry, in addition to the state parameters, information on the uncertainty measure (in our case, the state constrained reach probability). One possibility to construct a trajectory in the original state space is to construct an intermediate model, called *colored SHS*. In practice, only a finite number of values of state constrained reachability probability can be computed. Therefore, the uncertainty representation of the state space will be a map colored with a finite number of colors.

Definition 1 *A colored stochastic hybrid system is a collection*

$$(H, \mathcal{C}, \mathsf{C})$$

where

- $H := ((Q,d,\mathcal{X}),b,\sigma,Init,\lambda,R)$ *is a general stochastic hybrid system*
- $\mathcal{C}$ *is a finite set, whose elements are called colors*
- $\mathsf{C} : X \to \mathcal{C}$ *is a function that we call* coloring

Figure 2.5: Trajectory in the colored state space

A colored stochastic hybrid system can be attached to its uncertainty representation by defining a color to the a value set of the state constrained reachability probability. In this way, coloring can be used as an effective tool for model reduction.

An example of colored SHS is illustrated in Figure 4. In this example five colors are used for the uncertainty representation: the target state set is represented in light blue, the obstacle state set (which, in this example, is not connected) is represented in black. The probability represented in the figure is the complementary $\varphi_B(x) - p^B_{\neg A} = p^B_A$ of the obstacle avoidance probability $p^B_{\neg A}$ (of reaching B, starting from x, by avoiding A). This probability has the value 1 on the black colored areas (corresponding to the obstacles) and null value on the light blue colored area (corresponding to the target). The areas colored in white, grey, dark grey and dark blue correspond to intermediate, increasing values of the complementary of the constrained reachability probability. For example, the white colour corresponds to the states with the constrained reachability probability values ranging from 0.001 to 0.250. The grey colour corresponds to the states with the constrained reachability probability values ranging from 0.251 to 0.700. Similarly, the dark grey colored states correspond to probability values ranging 0.701 and 0.850, and the blue colored states correspond to probability values ranging 0.851 and 0.999. The trajectory is depicted in red. Remark that the trajectory "tries" to get quickly out of the dark colored state areas.

An immediate example at hand for colored state spaces is the case of flight within a geographical area contaminated with ash clouds. It is reasonable to consider the probability directly proportional to ash concentration. The North Atlantic Operations Bulletin 2010-009 imposes, from 16 May 2010: "Areas of Low Contamination" - areas where it is forecast that the concentration of volcanic ash will be below 2x10-3 g/m3; "Areas of High Contamination" - the forecasted concentration of ash is between 2-4 mg/m3 and "No Fly Areas". These areas are modeled by colored sets in our model. The light blue state sets will be the surrounding area of the destination airport. The model states corresponding to the "No Fly Areas" will be considered colored in black. The area of high contamination will form the blue colored state regions. The white state areas correspond to the physical areas of no contamination. The grey colored state regions will correspond to the areas of low contamination.

It is important to underline that, in the previous example, the colors are dynamic (they change over time corresponding to the volcanic ash concentration determined by the ash clouds dynamics). The colored SHS model can be extended to cope with this situation by considered a time indexed family of coloring functions

$$\{\mathcal{C}_s : X \to \mathcal{C}\}_{s \in S \subset [0,\infty)}$$

OPTIMAL DECENTRALIZED CONTROL

Properties

Feedback control dynamical system, non-convex optimisation.

By H. Dallali &
M. Brown

Description

Feedback is perhaps one of the most fundamental concept in control engineering, dynamical systems, and many other areas including biology and economic. In control engineering, the synthesis of feedback for a variety of complex systems has been studied for many years. The idea behind designing feedback control systems is to control the dynamics of a given system to achieve certain stability and performance criteria in the face of uncertainties, unmodelled dynamics, noise and disturbances. The feedback itself can be static or dynamic depending on the type of the system and the level of performance, required. The structure of the feedback can also be centralized or decentralized. The centralized feedback refers to the case where the measurements and control calculations are performed at a single location. However, in some physical systems due to constraints on the rate of information transfer among sensors and actuators centralized control is not feasible and it is desired to process the information locally at each subsystem which is called decentralized control. Although information is processed locally, the interactions and overall global dynamics of the large system must be considered when the decentralized feedback is designed. In addition, it is often desired to minimize the cost of the closed loop norm of the system while designing the feedback. Hence the optimal decentralized control can be cast as a non-convex optimisation problem.

Formalization

The general decentralized control problem can be formulated as the following optimisation problem

$$\begin{aligned} &\text{minimize } \|f(P,K)\| \\ &\text{s.t. } K \text{ stabilizes } P \\ &K \in S. \end{aligned} \tag{2.12}$$

where $\|f(P,K)\|$ is the norm of a closed loop map, K is the feedback that stabilizes the plant P and the subspace S defines the structure of K. The general solution to problem (2.12) is currently being studied in control systems community since this is a non-convex optimisation problem.

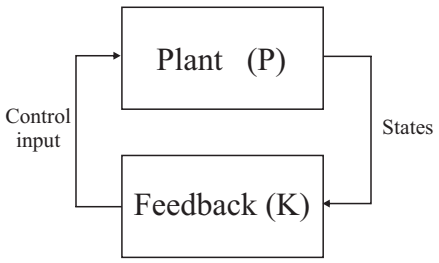


Figure 2.6: A closed loop feedback diagram.

In our case, the problem is simplified by limiting the solution of (2.12) to linear, block diagonal structure

gains as shown in (2.13)

$$K = \begin{bmatrix} K_1 & 0 & 0 & \cdots & 0 \\ 0 & K_2 & 0 & \cdots & 0 \\ 0 & 0 & \ddots & \cdots & 0 \\ \vdots & \vdots & \cdots & K_{n-1} & 0 \\ 0 & 0 & \cdots & 0 & K_n \end{bmatrix} \tag{2.13}$$

where n is the number of joints and K_i blocks are row vectors of dimension 1 by 3 for rigid models and 1 by 5 for compliant models. Each block K_i contains three PID gains for rigid robots or five PD-PID gains for compliant robots that provide feedback from the i^{th} joint position, velocity, motor position, motor velocity and integrator states.
Consider the discrete time system

$$\mathbf{x}(k+1) = A\mathbf{x}(k) + B\mathbf{u}(k) + \eta \tag{2.14}$$

Assuming that the state $\mathbf{x}$ is available for measurement and the pair (A,B) are controllable, the feedback can be expressed as $\mathbf{u}(k) = -K\mathbf{x}(k)$ and the closed loop system is

$$\mathbf{x}(k+1) = (A-BK)\mathbf{x}(k) + \eta \tag{2.15}$$

where $(A-BK)$ is asymptotically stable. The steady-state state covariance matrix $P = E[\mathbf{x}\mathbf{x}^T]$ is the solution to Lyapunov equation

$$P - (A-BK)P(A-BK)^T - \hat{Q} = 0 \tag{2.16}$$

where the noise covariance matrix is $E[\eta\eta^T] = \hat{Q}$. If every entry in the noise vector has the same variance β and the entries are all statistically independent or uncorrelated then the noise covariance is $E[\eta\eta^T] = \beta I$. The LQR based decentralized feedback optimisation problem is

$$\begin{aligned} &\min_{\mathbf{u}=-\text{diag}\{K_i\}\mathbf{x}} \quad J \\ &\text{subject to} \;\; \mathbf{x}(k+1) = A\mathbf{x}(k) + B\mathbf{u}(k) \end{aligned} \tag{2.17}$$

and the proposed discrete time, LQR-LMI formulation is

$$\begin{aligned} &\min_{(P,Y,X)} \;\; tr[QP] + tr[X] \;\; \text{subject to:} \\ &\begin{bmatrix} (P-\beta I) & (AP-BY) \\ (AP-BY)^T & P \end{bmatrix} > 0 \\ &\begin{bmatrix} X & R^{\frac{1}{2}}Y \\ Y^T R^{\frac{1}{2}} & P \end{bmatrix} > 0 \end{aligned} \tag{2.18}$$

where Q and R are the LQR penalties and P is the solution of the Lyapunov equation. In addition, $Y = KP$ is an auxiliary variable introduced to cater for the nonlinearity introduced by the product of K and P in the Lyapunov equation $P - \beta I - (A-BK)P(A-BK)^T > 0$.
The static state feedback is obtained by

$$K = YP^{-1} \tag{2.19}$$

Ideally, the decentralized structure must be directly imposed on the feedback gain K as in (2.13). However, due to the product between K and P that causes nonlinearity, the decentralized structure is indirectly imposed on the LMI variables P and Y. It can be easily shown that if these variables have a block diagonal structure the resulting feedback gain in (2.19) will have the decentralized structure. Hence in this paper, P and Y are defined to have similar structure as in (2.13). The only difference is that diagonal blocks of P are square matrices and diagonal blocks of Y have the same dimension as blocks of K. Moreover, symmetric decentralized gains are desirable in bipedal walking to avoid switching the gains while the robot changes

the supporting leg. These gains are derived by providing a mirror reflection of the blocks with respect to the central block.

There are effective LMI solvers that use Semidefinite programming (SDP) to solve (2.18). Further information about this method is provided in [1].

Softwares and references

[1] H. Dallali, G.A. Medrano-Cerda and M. Brown, *Decentralized PID Joint Servo Design for a Compliant Humanoid Robot via LQR-LMI Approach*, Available online at *http://eprints.ma.man.ac.uk/1651/*, 2011.

DIGITAL CONTROL AND ITERATED FUNCTION SYSTEMS

Properties

By P. Shmerkin

- If the maps g_i are all expanding, then the largest controllable region is the only nonempty compact set K satisfying $K = \bigcup_{i=1}^{m} g_i^{-1}(K)$.
- For the case of two expanding maps in $\mathbb{R}^2$ and three expanding maps in $\mathbb{R}^3$, there are easy ways to verify if the controllable region has nonempty interior.

Description

We consider the problem of keeping a dynamical system near a certain point in the phase space. This point needs not be a stable equilibrium; it could even be an unstable equilibrium. This can be achieved by applying periodic forces from different directions (at least 2) for fixed periods of time. This is known in the control literature as bang-bang control, although we prefer the term digital control as we are looking at a wider class of situations.

This setting can be modeled by iterated function systems (IFS). Indeed, let τ be the time period over which each force is applied, and let g_i be the time-τ map of the original system perturbed by the force F_i ($i = 1, \ldots, m$, where m is the number of different forces available). Then, starting from a point x in phase space, we can move the system to any of the points $g_i(x)$.

In the classical IFS theory the maps g_i are required to be contractions, but in our case this will never happen, in fact the maps g_i will have strong expanding properties in certain directions. We define a region K in phase space to be controllable if, for any $x \in K$, there is i such that $g_i(x) \in K$ or, in other words, $K \subset \bigcup_{i=1}^{m} g_i^{-1}(K)$. This generalizes the classical notion of attractor of an IFS.

Even though the maps g_i are not contracting, the classical theory turns out to be useful in helping find appropriate controllable regions. This can be seen in simple models such as an inverted pendulum in either 2 or 3 dimensions.

FEWNOMIALS

Description

Suppose that $p(x) = a_d x^d + \cdots + a_0$ is a polynomial with real coefficients. Descartes's rule of signs says that the number of positive real zeros of p is at most the number of sign changes in the non-zero coefficients of p. This implies that if p has exactly k non-zero monomials then p has at most $k-1$ positive real roots, and this is independent of the degree of p!. Is there a version of this result for polynomials in many variables?

By G. Jones

Khovanskii's theorem

Consider a system of equations

$$\begin{aligned} p_1(x) &= 0 \\ &\vdots \\ p_n(x) &= 0 \end{aligned}$$

where $p_1, \ldots, p_n$ are polynomials in n variables with real coefficients and $x \in \mathbb{R}^n$. If there are at most k distinct monomials occurring in $p_1, \ldots, p_n$ then the system has at most

$$2^{\frac{k(k-1)}{2}}(n+1)^k$$

non-degenerate solutions (i.e. solutions x at which the Jacobian determinant is non-zero) in the positive quadrant of $\mathbb{R}^n$.

These systems are called *fewnomial* systems.

Rather than proving this directly, Khovanskii proves the following result, which involves certain transcendental equations. Suppose that $q_1, \ldots, q_n$ are real polynomials in $n+k$ variables, and let $c_1, \ldots, c_k \in \mathbb{R}^n, c_i = (c_{i,1}, \ldots, c_{i,n})$. Then the systems of equations

$$\begin{aligned} q_1(x, \exp(c_{1,1}x_1 + \cdots + c_{1,n}x_n), \ldots, \exp(c_{k,1}x_1 + \cdots + c_{k,n}x_n) &= 0 \\ &\vdots \\ q_n(x, \exp(c_{1,1}x_1 + \cdots + c_{1,n}x_n), \ldots, \exp(c_{k,1}x_1 + \cdots + c_{k,n}x_n) &= 0 \end{aligned}$$

has at most

$$2^{\frac{k(k-1)}{2}} \prod \beta_i (1 + \sum \beta_i)^k$$

non-degenerate solutions in $\mathbb{R}^n$, where β_i is the degree of q_i.

The result on fewnomials follows immediately, by taking $x_i = \exp(y_i)$. To prove the result on exponential equations, Khovanskii proceeds by induction on k (note that for $k = 0$ the result follows from Bezout's theorem) and replaces the last exponential with a new variable. The proof is too long to give here, but in the end it, like Descartes's rule, comes down to a clever application of Rolle's theorem.

References

[1] Khovanskiĭ, A. G., *Fewnomials*, Translations of Mathematical Monographs, American Mathematical Society, v.88, 1991.

THE MOVING AVERAGE TRANSFORMATION

Properties

By J. Hook

The Moving Average Transformation is a novel systematic procedure for smoothing non-smooth dynamical systems. It transforms discontinuous systems into equivalent continuous systems and continuous but non-differentiable systems into equivalent differentiable systems and so on. This smoothing enables us to apply standard techniques which rely on differentiability or higher order smoothness. In this part we introduce the transformation, the notion of dynamical equivalence and present a numerical example of the smoothing procedure.
In principal this technique could be applied to any non-smooth dynamical system. The smoothed systems exhibit a rich topology which can give us insight into the original systems dynamics and the differentiability of the vector field enables us to apply standard techniques such as the calculation of Lyapunov exponents from data.
It is my hope to develops further applications of the technique for real life engineering systems.

Description

Discontinuities in non-smooth models of physical systems are approximations of highly non-linear smooth processes that take place on very short time scales. The Newtonian impact of a ball bouncing on the ground is discontinuous in velocity but in reality the ball and ground are compliant, elastic bodies and discontinuities in velocity are physically impossible! Introducing compliance gives us a continuous but non-smooth system. The dynamics are non-smooth because there is only a force between the ball and ground when they are in contact, but this too is a false assumption. The electric fields of the atoms in the ball and ground are continually interacting in a smooth way it is just that the forces produced by these interactions are only significant during the short impacting phase of the dynamics.

Of course we approximate these fast processes by non-smooth discontinuities to obtain simpler, more manageable models of physical processes. Ironically the piecewise smooth nature of the resulting models can make them more difficult to study. For example non-differentiability makes it impossible to apply many standard numerical techniques to these systems [1, 2, 3, 4, 5]. For a discontinuous system like the bouncing ball the topology of the state space and any attractors it contains is broken up by jumps in the evolution so that an attractor will typically appear to be comprised of several disconnected parts, if we patch these together by connecting jump take offs and landings we arrive at a new topological space which can tell us much more about the systems dynamics than the topology of the original disconnected object.

There is therefore some motivation for transforming non-smooth systems (which as outlined are approximations of smooth systems) back into smooth systems. Several authors have tried a fairly direct approach, adding a small region where fast, smooth dynamics replace the non-smooth discontinuity in a process called regularization [5]. Whilst this doesn't necessarily mean introducing more layers of physically realistic modeling, since the extra components can be chosen to be as simple as possible, it still feels like a step in the wrong direction. The non-smooth discontinuities are a simplification and we should be able to exploit this trade off in realism by studying more mathematically appealing tractable systems.

In addition to these approximated physical processes there is a huge catalogue of non-smooth systems in control theory where digital switching between different modes really should be thought of as being intrinsically non-smooth [6]. Abstract non-smooth systems are also of great interest mathematically as they can generically exhibit bifurcation structures which would be impossible or of high codimension in the space of smooth systems [7]. Although it is not natural to think of one of these systems as the limit of some smooth system it can still be advantageous to smooth them in some way as all the problems associated with the analysis of non-smooth physical systems are also present here.

In [9] we introduce the Moving Average Transformation which can be thought of as a change of variables which transforms discontinuous systems into continuous systems and continuous but non-differentiable

systems into differentiable systems. For a non-smooth system with evolution $\phi_t : X \to X$ the transformation is defined by

$$\Phi(x) = \int_{-1}^{0} \phi_\tau(x) d\tau \tag{2.20}$$

Since the evolution of the system is automatically incorporated into the transformation it is possible to systematically obtain smoothed systems using this technique.

The transformation provides an explicit link between a non-smooth system and its smoothed *Dynamically Equivalent* counterpart. This enables us to better understand topological aspects of the dynamics and apply standard numerical techniques that rely on differentiability.

Crucially the technique is totally systematic, introduces no extraneous dynamics and provides a clear equivalence between the original and transformed (smoothed) system.

Relationship between original and smoothed system

The theory surrounding the moving average transformation is quite subtle and investigated fully in [9]. Rather than proving that the transformation smooths non-smooth systems here we shall instead, by way of motivation, present a very simple example. Although the method here is quite ad hoc it illustrates the desired relationship between the original and smooth systems as well as the necessary properties of the transformation between them.

Consider a unit mass attached to a spring with unit stiffness along with a wall where the mass undergoes Newtonian impacts positioned at $x = 0$ where the spring is at its natural length . We assume that the spring is light and linear and that there is no friction so that away from the wall the dynamics are governed by the linear differential equation

$$\frac{d}{dt}\begin{pmatrix} x \\ \dot{x} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}\begin{pmatrix} x \\ \dot{x} \end{pmatrix} \tag{2.21}$$

When the mass hits the wall there is an impact with restitution c so that whenever we reach the set $In = \{(x,\dot{x}) \in \mathbb{R}^2 : x = 0, \dot{x} < 0\}$ we instantaneously apply the map $R(x,\dot{x}) = (x, -c\dot{x})$ which maps In to $Out = \{(x,\dot{x}) \in \mathbb{R}^2 : x = 0, \dot{x} > 0\}$. Since we are mapped instantaneously from In to Out the state space of the system is given by $M = \{(x,\dot{x}) \in \mathbb{R}^2 : x \geq 0\}/\{(x,\dot{x}) \in \mathbb{R}^2 : x = 0, \dot{x} < 0\}$.

The system then evolves by flowing according to (2) until hitting In then mapping to Out and flowing according to (2) again... Define $\phi_t : M \mapsto M$ to be the time t evolution map that takes a point in the state space foreword in time t seconds.

We seek a smoothing transformation T which can be thought of as a change of variables with the property that the transformed system $\varphi_t = T \circ \phi_t \circ T^{-1}$ is smooth.

Claim The transformation $T : M \mapsto \{(y_1, y_2) \in \mathbb{R}^2\}$ defined in polar coordinates by

$$T(\theta, r) = (2\theta, rc^{\frac{\theta}{\pi}}) \tag{2.22}$$

provides an equivalence between our discontinuous system and the linear system evolved by the ODE

$$\frac{d}{dt}\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} \frac{1}{\pi}\log c & 2 \\ -2 & \frac{1}{\pi}\log c \end{pmatrix}\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \tag{2.23}$$

defined on $\mathbb{R}^2$ equipped with the Euclidian metric.

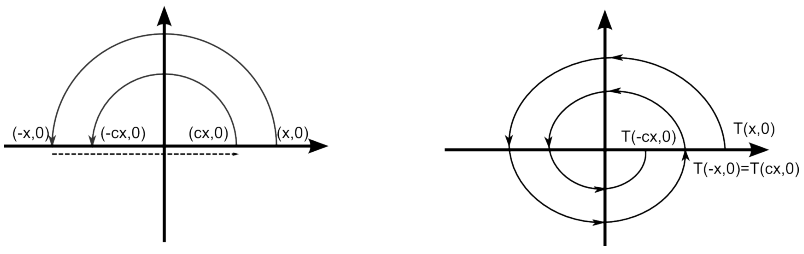


Figure 2.7: An orbit to the original system (Left) and its image under T (right).

Proof This rests quite heavily on our definition of equivalent! Clearly the two systems are not topologically equivalent since continuity is a topological invariant.

Definition If ϕ is the evolution of a non-smooth system which includes some jumps that are applied instantaneously on reaching a set Σ then we say that the transformation T provides a *Dynamical Equivalence* between ϕ and φ via $\varphi_t = T \circ \phi_t \circ T^{-1}$ if

- T is continuous
- T^{-1} exists and is continuous everywhere except $T(\Sigma)$
- $\frac{\delta T}{\delta x}$ exists and is non-zero everywhere

This is a relaxation of the usual definition for topological equivalence where T^{-1} is required to be continuous everywhere. Clearly our claim holds for this notion of equivalence.

Definition We define a *Cheat Metric d* on a discontinuous system to be a continuous metric which identifies jump take offs and landings so that $d(s,R(s)) = 0$ for all $s \in \Sigma$. To define d everywhere else we could just measure the length of the shortest path between two points but allow the path to jump over the discontinues in the same way as ϕ.

Viewed with the cheat metric all systems have continuous orbits, hence the cheat. This means that the original system with Euclidian metric is not topologically equivalent to the original system with cheat metric which in tern is topologically equivalent to the smoothed system with Euclidian metric.

The second condition for dynamical equivalence that T^{-1} exists and is continuous everywhere except $T(\Sigma)$ is equivalent to the condition that T^{-1} is continuous w.r.t a cheat metric. Therefore dynamically equivalent systems are topologically equivalent when viewed with a cheat metric.

The relationship between our original discontinuous system and the dynamically equivalent smooth system is quite subtle. If we equip M with the Euclidian metric then T^{-1} is discontinuous so the two systems are not topologically equivalent. However the evolutions are still interchangeable via $\varphi_\tau = T \circ \phi_\tau \circ T^{-1}$ so we can use the new system to describe the original. Since our new system is linear it is easy to solve and we can therefore obtain a neat expression for the evolution

$$\phi_t = T^{-1} \circ \frac{t}{\pi} \log c \begin{pmatrix} \sin 2t & \cos 2t \\ -\cos 2t & \sin 2t \end{pmatrix} \circ T \tag{2.24}$$

Complications caused by the discontinuity have all been captured in the transformation T. Since the new system is differentiable we can also calculate the stability of the fixed point at the origin which is determined by by the real part of (4)'s eigenvalue pair $\lambda_\pm = \frac{1}{\pi} \log c \pm 2i$. This agrees with the standard non-smooth analysis using the saltation matrix formulation [2].

In [9] we show that the Moving Average Transformation automatically has the same properties as T. We are also able to investigate the action of the transformation in the vicinity of all typical non-smooth discontinuities, providing us with an atlas of possible behaviors for the transformed systems.

Numerical example

The Moving Average Filter can easily be applied to time series data from a computer simulation of a non-smooth system. Since this data is equivalent to unsmoothed output from the smoothed system we can use it to study the systems obtained by applying the Moving Average Transformation to non-smooth systems. The theory developed in [9] enables us to understand the complex relationship between the non-smooth system and its smoothed counterparts.

Friction oscillator

Systems with static and dynamic friction can exhibit sliding behavior. In our model we consider a mass on a linear spring subjected to a periodic force resting on a rough moving belt with a piecewise linear friction force

$$Fric(y) = \begin{cases} ay+b & y>0 \\ -ay-b & y<0 \end{cases} \tag{2.25}$$

where y is the velocity of the mass relative to the surface of the belt. The force acting on the block is given by

$$F(x,\dot{x},t) = kx - d\dot{x} + l\sin(\omega t) + Fric(\dot{x}-u) \tag{2.26}$$

The state space of this dynamical system is three dimensional (position,velocity,forcing-phase). We will embed the phase variable as an interval $[0,2\pi]$ rather than on the circle as it enables us to embed the the whole state space in $\mathbb{R}^3$ and the jump from 2π to 0 gives us some discontinuities in the flow which we can resolve with the transformation.

Procedure

For a given set of parameter values we simulate the oscillator on a computer and record a time series in the three dimensional state variable. The recording is just a long sequence $(x_i,\dot{x}_i,\tau_i)_{i=1}^N$ where we use a time-step of $0.05s$. The smoothing filter is just a summed average applied to the now discrete time data.

$$\Phi(x,\dot{x},\tau)_i = \frac{1}{40}\sum_{j=i-39}^{i}(x_j,\dot{x}_j,\tau_j) \tag{2.27}$$

So that we are averaging over a period of $2s$. The accuracy of our routine is actually slightly better than this as we can approximate the average between time steps during the integration routine which improves the resolution of the sum.

To obtain a differentiable time series we simply apply the same filter a second time to the data.

Data

We use the following parameters $k=1.2$, $d=0.0$, $w=1.02$, $l=1.9$, $u=3.4$, $a=0.04$ and $b=0.1$ whose dynamics exhibit a grazing orbit that gives rise to chaotic dynamics on what appears to be a fully 2 dimensional attractor. We plot the time series for the three systems, discontinuous, continuous/non-differentiable and differentiable, see fig 2,a.

Results

Both applications of the filter give us useful insight into the original systems dynamics. When we transform the system from a discontinuous system to a continuous system we change the topology of the attractor. As outlined in the introduction we claim that the topology of the continuous system gives a better characterization of the dynamics and that is the case here. Homologically the discontinuous system is equivalent to two points whereas the continuous system is a figure of eight, the two possible loops in the attractor provide the topological mechanism for chaos, see fig 2,b.

On the second application of the filter we arrive at a differentiable system which can be thought of as an ODE with continuous RHS. We can therefore locally approximate this ODE from our data and use this to compute lyupanov exponents along the flow. The clear positive exponent shows that we have a chaotic system, see Fig 2,c.

References

[1] Bernardo M. and Budd C. and Champneys A.R. and Kowalczyk P., *Piecewise-smooth Dynamical Systems*, Springer, 2008.

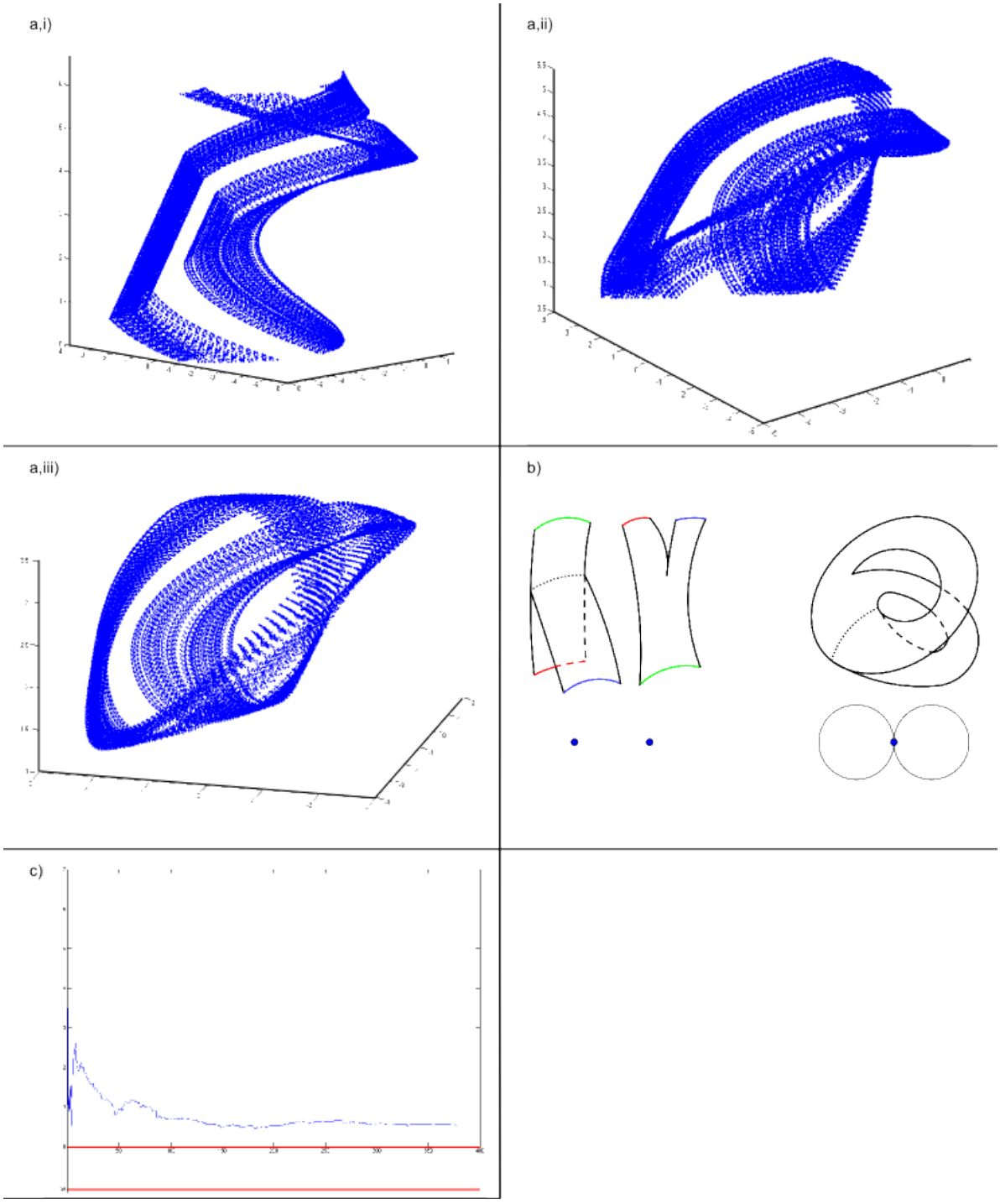


Figure 2.8: a) Time series recorded from i) Original discontinuous system, ii) Once transformed continuous system, iii) Twice transformed differentiable system, b) Convergence of Lyupanov exponent calculation, c) Topology of invariant set for discontinuous and transformed continuous system.

[2] Kunze M. and Küpper T., *Nonsmooth dynamical systems: An overview*, Ergodic Theory, Analysis, and Efficient Simulation of Dynamical Systems, pp.431-452, 2001.

[3] Arango I. and Taborda J.A., *Continuation of nonsmooth bifurcations in filippov systems using singular point tracking*, International Journal of Applied Mathematics and Informatics. Issue 1, v.1, pp.36-49, 2007.

[4] Llibre J. and Da Silva P.R. and Teixeira M.A., *Regularization of Discontinuous Vector Fields on* $\mathbb{R}^3$ *via Singular Perturbation*, Journal of Dynamics and Differential Equations, 19(2), pp.309-331, 2007.

[5] Gang T. and Frank L., *Adaptive Control of Nonsmooth Dynamic Systems*, Springer, 2001.

[6] Di Bernardo M. and Hogan S.J., *Discontinuity-induced bifurcations of piecewise smooth dynamical systems*, Phil. Trans. R. Soc. A., 368(1930), pp.4915-4935, 2010.

[7] Kunze M. and Küpper T., *Qualitative bifurcation analysis of a non-smooth friction-oscillator model*, ZAMP, v.48, pp.87-101, 1996.

[8] Huke J., *Embedding nonlinear dynamical systems: A guide to Takens' theorem*, Technical Report DRA, Malvern, 1993.

[9] Hook J, (*Smothing non-smooth sytems with the Moving Average Transformation*, In preperation, 2011.

DIMENSIONALITY REDUCTION VIA OPTIMIZATION ON THE GRASSMANNIAN

Abstract

By C. Wolshman

We present a summary of a method for performing dimensionality reduction on a set of data points in a high-dimensional Euclidean space, by optimizing a cost function over the Grassmannian. We include a compatible line search and a procedure for obtaining a sensible initial condition.

The Problem

Given a set of data points in $\mathbb{R}^n$,

$$X = \{x_1, \cdots, x_N\} \subset \mathbb{R}^n,$$

we want to find a projection $P : \mathbb{R}^n \to \mathbb{R}^d$ onto a d-dimensional subspace, such that $P|_X$ is injective - i.e. we want to be able to describe the data in a lower-dimensional space.

An orthogonal projection from $\mathbb{R}^n$ to $\mathbb{R}^d$ is represented by an orthonormal $n \times d$ matrix W, (orthonormal: $W^{\mathrm{T}}W = I_d$), whereby each column of W is a basis vector for $\mathbb{R}^d$ expressed in the basis of $\mathbb{R}^n$. Collectively they form an orthonormal d-frame in $\mathbb{R}^n$. The projection is then given by $x \mapsto W^{\mathrm{T}}x$.

Cost Function

We can obtain an optimal projection W by minimizing a suitable cost function. Let S be the set of unit secants,

$$S = \left\{ \frac{x-y}{\|x-y\|} : x \neq y \ \forall x, y \in X \right\}$$

The cost function $\mathcal{F} : V_d(\mathbb{R}^n) \to \mathbb{R}$ is given by [2]

$$\mathcal{F}(W) = \frac{1}{|S|} \sum_{k \in S} \left\| W^{\mathrm{T}} k \right\|^{-1},$$

where $V_d(\mathbb{R}^n)$ is the Stiefel manifold (the set of orthonormal d-frames in $\mathbb{R}^n$). As this cost function has the orthogonal symmetry, $\mathcal{F}(WQ) = \mathcal{F}(W)$ (where $Q \in \mathrm{O}(d)$), its value only depends on the subspace spanned by the frame, and thus on the Grassmannian, $Gr_d(\mathbb{R}^n)$ (the set of d-dimensional linear subspaces of $\mathbb{R}^n$). As the Grassmannian is a submanifold within the space of $n \times d$ real matrices, we will require some differential geometry to perform the optimization.

Determining a Sensible Initial Condition

Let $\sigma^i \in \mathbb{R}$ be the spread on the ith co-ordinate axis with corresponding basis vector e_i.

$$\sigma^i := \left| \max_j (x_j)^i - \min_j (x_j)^i \right|.$$

Let $a : \{1, \cdots, n\} \to \{1, \cdots, n\}$ be a bijection such that

$$\sigma^{a(1)} \geq \cdots \geq \sigma^{a(n)}$$

i.e. $a(1)$ is the axis with the largest spread and $a(n)$ is the axis with the smallest spread. Choose the d basis vectors with the largest spreads,

$$e_{a(1)}, \cdots, e_{a(d)}.$$

The initial condition is then the projection onto this frame,

$$W = \begin{bmatrix} e_{a(1)} & \cdots & e_{a(d)} \end{bmatrix}$$

Optimization on the Grassmannian

When minimizing the cost function, each step must remain on the Grassmannian. Optimization will therefore involve stepping along the geodesic - the direction and step amount are to be found.

For a simple gradient descent, the step direction, Λ, is given by

$$\Lambda = -G$$

where G is the (tangential) gradient [1]

$$G := (I_n - WW^{\mathrm{T}})\frac{\partial \mathcal{F}}{\partial W}$$

Geodesic

The geodesic along the Grassmannian from initial point W in the direction $\Lambda = U\Sigma V^{\mathrm{T}}$ (singular value decomposition) is given by [1]

$$W(\alpha) := \begin{bmatrix} WV & U \end{bmatrix} \begin{bmatrix} \cos\Sigma\alpha \\ \sin\Sigma\alpha \end{bmatrix} V^{\mathrm{T}}. \tag{2.28}$$

Parallel Translation

The parallel translation of an arbitrary vector Δ at W along the geodesic in the direction of $\Lambda = U\Sigma V^{\mathrm{T}}$ is given by [1]

$$\Delta(\alpha) := \left(\begin{bmatrix} WV & U \end{bmatrix} \begin{bmatrix} -\sin\Sigma\alpha \\ \cos\Sigma\alpha \end{bmatrix} U^{\mathrm{T}} + (I_n - UU^{\mathrm{T}}) \right) \Delta. \tag{2.29}$$

Line Search

After computing the search direction, Λ, we require a scalar, α such that

$$\psi(\alpha) := \mathcal{F}(W(\alpha)) \tag{2.30}$$

is approximately minimized. In order to use the line searching techniques, we require the derivative of ψ. The derivative with respect to α (i.e. along the geodesic) is

$$\psi'(\alpha) = \langle G(\alpha), \Lambda(\alpha) \rangle, \tag{2.31}$$

where $G(\alpha)$ is the gradient of $\mathcal{F}$ at α,

$$G(\alpha) := (I_n - W(\alpha)W^{\mathrm{T}}(\alpha)) \left.\frac{\partial \mathcal{F}}{\partial W}\right|_{W(\alpha)}, \tag{2.32}$$

and $\Lambda(\alpha)$ is the parallel transport of Λ along the geodesic to $W(\alpha)$. The inner product is the usual

$$\langle A, B \rangle = \mathrm{Tr}(A^{\mathrm{T}}B).$$

Let $\alpha^* > 0$ be a local minimum of ψ. First we establish an upper and lower bound, $\alpha_{\min} < \alpha^* < \alpha_{\max}$ such that $\psi'(\alpha_{\min}) < 0$ and $\psi'(\alpha_{\max}) > 0$. As this is part of a minimization problem, $\alpha = 0$ is almost

certainly a point with negative gradient (the step direction is chosen so that this is the case), we can use $\alpha_{\min} = 0$.

In order to find an upper bound, we use the value of α from the previous optimization step, denoted α_{old}. If $\psi'(\alpha_{\text{old}}) > 0$ we can use $\alpha_{\max} = \alpha_{\text{old}}$. If not, α_{old} is repeatedly multiplied by a factor of $\gamma > 1$ until $\psi'(\gamma^n \alpha_{\text{old}}) > 0$, we then use $\alpha_{\max} = \gamma^n \alpha_{\text{old}}$. We use $\gamma = 2$.

Once the bounds have been established, an iterative procedure is used to find a solution for α^*, as follows.

1. Use the linear interpolation of the two bounds to estimate the zero crossing,

$$c := \alpha_{\min} - \frac{\alpha_{\max} - \alpha_{\min}}{\psi'(\alpha_{\max}) - \psi'(\alpha_{\min})} \psi'(\alpha_{\min}) \tag{2.33}$$

2. Test the Wolfe conditions on c: return c if satisfied.
3. Compute $\psi'(c)$.
4. If $\psi'(c) > 0$ it becomes the new upper bound $\alpha_{\max} = c$. If $\psi'(c) < 0$, it becomes the new lower bound, $\alpha_{\min} = c$.
5. Repeat for a limited number of iterations, or until a tolerance is reached. A handfull of iterations is usually plenty.

Summary of the Method

1. Compute the set of unit secants S for the data set.
2. Determine a suitable initial condition using the method described above.
3. Perform the optimization using the following iterative algorithm.

Optimization Algorithm

1. Compute tangent vector for the step direction using the gradient descent,

$$\Lambda = (WW^{\mathrm{T}} - I_n)\frac{\partial \mathcal{F}}{\partial W}.$$

2. Compute SVD of tangent vector,

$$U\Sigma V^{\mathrm{T}} = \Lambda.$$

3. Step by an amount Δt along the geodesic in the given direction,

$$W(\Delta t) = \begin{bmatrix} WV & U \end{bmatrix} \begin{bmatrix} \cos \Sigma \Delta t \\ \sin \Sigma \Delta t \end{bmatrix} V^{\mathrm{T}}.$$

   (a) A suitable step amount, Δt, can be determined by a line search of the function

$$\psi(\Delta t) := \mathcal{F}(W(\Delta t)).$$

   (b) Perform the step: compute $W(\Delta t)$.

Softwares and references

[1] A. Edelman, T. Arias, S. Smith, *The Geometry of Algorithms with Orthogonality Constraints*, SIAM Journal on Matrix Analysis and Applications, V. 20, N. 2, 1999, pp. 303–353.

[2] D. Broomhead, M. Kirby, *Dimensionality Reduction Using Secant-Based Projection Methods: The Induced Dynamics in Projected Systems*, J. Nonlinear Dynamics, V. 41,2005, pp. 47–67.

TWO BALL NEWTON CRADLES

Description

By P. Glendinning

Newton's cradle is an executive toy in which a number of hard spheres are suspended from two horizontal bars so that when they hang in equilibrium the spheres are precisely aligned along the axis of the bars, with adjacent spheres just touching each other. Now, if an end ball is pulled back and then released so that it swings back to strike the remaining line of balls, the impulse of the impact is transmitted down the line and the ball at the other end swings up and the process repeats. The intermediate balls remain stationary throughout the process, serving only to transmit the impulse down the line. As such the cradle is a perfect illustration of the transmission of a Newtonian impulse.

That, anyway, is the story usually given to describe the motion of the cradle. In fact, the first collisions set up small movements in the other balls, with some moving backwards and some forwards, and this explains why most commercial cradles contain only five balls. This effect quickly becomes very visible if more balls are used.

To model the cradle more precisely, the forces at work when two hard spheres collide need to be understood. The detail is complicated – how are the balls deformed whilst they are in contact? – but a simple idealization, the Hertzian potential, is often used. This introduces a nonlinear interaction with a force proportional to the deformation of the balls to some power, often taken to be taken to be $\frac{3}{2}$. This means that even if the swinging of the pendulums is modelled by simple harmonic motion (one of the mainstays of physics, this means that the dynamics can be described by sines and cosines) the interaction cannot be solved explicitly.

Figure 2.9: Newton's Cradle

It turns out that by using the formalism of hybrid systems the simplest situation, the two ball Newton cradle, can be modelled explicitly *without* needing to solve the nonlinear problem; only two constants arising from that part of the interaction enter the final calculation, and these can act as parameters that describe how changing the contact interactions affects the dynamics of the cradle.

Formalization

Suppose the positions of the centres of mass of the two balls are at x_1 and $2r + x_2$ respectively, where r is the radius of the balls and so if x_1 and x_2 are both zero the centres of the mass balls are a distance $2r$ from each other.

Whilst $x_1 - x_2 < 0$ the balls are not in contact and each behaves as a pendulum, and

$$\ddot{x}_1 = -\omega^2 x_1, \quad \ddot{x}_2 = -\omega^2 x_2 \tag{2.34}$$

and $\omega^2 = g/\ell$, where g is the acceleration due to gravity, and ℓ the vertical length of the pendulum wires.

If $x_1 - x_2 > 0$ then the balls are in contact and the Hertzian force comes into play in addition to the gravitational force and

$$\ddot{x}_1 = -\omega^2 x - V'(x_1 - x_2), \quad \ddot{x}_2 = -\omega^2 x_2 + V'(x_1 - x_2) \tag{2.35}$$

where the Hetzian potential V models the visco-elastic forces (per unit mass), so for the Hertzian case

$$V(s) = \frac{K}{1+\alpha}\, s^{1+\alpha}. \tag{2.36}$$

It is natural to work in centre of mass (times two) and relative position coordinates

$$Q = x_1 + x_2, \quad q = x_1 - x_2 \tag{2.37}$$

and in terms of these variables we can see the system as a classic hybrid system: the evolution equation changes as q passes through zero.

It turns out that the only information that is needed to be able to solve explicitly for the values of Q and q just after a collision given their values just after the previous collision is the time of contact, τ, the fact that after contact the relative velocity is reversed (due to symmetry). Thus in the absence of dissipation the hybrid approach makes it possible to write down a difference equation for Q, $\dot{Q}$, q and $\dot{q}$ from one collision to the next. See [1] for the details.

Outcome

From the difference equation it is simple to predict the motion of a two-ball Newton cradle, and it is not the regular oscillation with stationary contact positions expected from the traditional Newtonian impulse account. The equations predict a slow oscillation of the collision point from side to side as the dynamics evolves with time. This effect can be verified by observation using a commercial cradle. It is possible that the insights provided by this analysis will provide new ways of investigating the Hertzian potential itself by fitting the observations to the parameters that determine the oscillations of the contact points. But whether this happens or not it is a nice mechanical example of the insights given by the hybrid (or non-smooth) dynamical approach.

Softwares and references

[1] P. Glendinning (2011) Two ball Newton's cradle, *Physical Review E*, **84** 067201.

MODELLING NETWORKS IN EPILEPSY

Description

M. Goodfellow

Epilepsy is a prevalent neurological disorder of complex networks in the brain. These networks exist at different scales of organisation from the sub-cellular level to large regions of nervous tissue. The latter is important as it supports large scale abnormal activity, thus leading to clinical symptoms, and is also interrogated in non-invasive recordings. At this level, the electroencephalogram (EEG) is still an accepted clinical tool with which to image the pathological nature of the epileptic brain.

Figure 2.10: An example intermittent "burst" of high amplitude activity on the mean field of the model described in [1]. The model is composed of 25 globally coupled compartments with heterogeneous parameters. The mean field is displayed at the bottom of the figure. Time courses are also shown for 3 constituent compartments. The heterogeneity in parameters places each of these compartments into a different oscillating regime in the equivalent uncoupled model, which is indicated as a time course above the bursting dynamics. The model predicts that during the high amplitude mean field burst some of the interacting components display different dynamics.

A defining feature of epilepsy is the ability of the brain to produce state transitions in the dynamics of the EEG during epileptic seizures. In addition, the epileptic brain is known to posses an abnormal "excitability", producing complex transient responses to electrical or sensory perturbation. Crucially these features are not only dependent upon the condition of epilepsy, but are also location-dependent *within* the epileptic brain (see e.g. [4, 2]). The multi-scale complexity of "epileptogenic" brain networks necessitates the use of mathematical modelling in order to conceptualise important processes and ways in which they may be affected. Thus epilepsy constitutes an interesting problem which can be addressed using networks of dynamical systems.

Formalisation

Recent modelling studies have investigated the role of tissue heterogeneity in the production of abnormal dynamics at an abstract level [1, 2]. It was found that both spontaneous rhythmic transitions as well as prolonged transient responses to perturbation can be underpinned by networks of connected, heterogeneous

elements. An example of interesting model dynamics due to network connectivity and heterogeneity in one of these models is given in Figure 2.10. This model [1] displays intermittent transitions in state on the mean field, as observed in EEG recorded from epilepsy patients. These studies frame in model terms the potential relationship between the nature of an "epileptogenic zone" and the kinds of dynamics which can be observed due to its presence. It is hoped that future work in this direction will aid the localisation of the epileptogenic zone, which is crucial to treatment in the severest of epilepsies.

Softwares and references

[1] Goodfellow, M., Schindler, K., Baier, G., 2011. Intermittent spike-wave dynamics in a heterogeneous, spatially extended neural mass model. Neuroimage 55 (3), 920–32.

[2] Goodfellow, M., Schindler, K., Baier, G., 2012. Self-organised transients in a neural mass model of epileptogenic tissue dynamics. Neuroimage 59 (3), 2644–60.

[3] Lopes da Silva, F., Blanes, W., Kalitzin, S. N., Parra, J., Suffczynski, P., Velis, D. N., 2003. Epilepsies as dynamical diseases of brain systems: basic models of the transition between normal and epileptic activity. Epilepsia 44 Suppl 12, 72–83.

[4] Valentin, A., Anderson, M., Alarcon, G., Seoane, J. J. G., Selway, R., Binnie, C. D., Polkey, C. E., 2002. Responses to single pulse electrical stimulation identify epileptogenesis in the human brain in vivo. Brain 125, 1709–1718.

UPDATING BASIS FOR POD - IMAGE ANALYSIS: TABLE

Properties

By Y. Chahlaoui

This example shows that a 10% basis set is often sufficient to reconstruct images with an acceptable visual quality. The small table tennis ball is not quite distinguishable in the approximations we have just a ghost trace. This phenomenon is due to the fact that we collect the dominant behaviour of the sequence and so we have a certain history of the sequence which influences the reconstruction.

Description

This set (we show some extracted images from the sequence in Figure 2.11) is used to compare different algorithms to compute and update the dominant basis generated by Proper Orthogonal Decomposition algorithm for model reduction of nonlinear systems (1.2)(i), without recomputing the basis set from scratch, which also has applications in active vision [1]. The sequence shows two ping-pong players. This sequence has an interesting characteristic which represents a certain difficulty for the approximation : the movement is very fast. The sequence contain 300 images. Each image consists of 288×352 pixels, and each pixel has 256 gray levels. We cut up each image into 352 columns and then we form one vector of dimension 101376 for each image. The matrix M_S has thus dimensions 101376×300. Figure 2.12 shows this in more detail.

Figure 2.11: Some images extracted from the Table sequence.

Handling directly the whole matrix M_S is very costly in time and memory. The run time of its SVD is about 1200 seconds while the construction of the matrix takes about 350 seconds. The total run time is thus about 1550 seconds and uses at least 245 Megabytes. Furthermore, the batch algorithm is not suitable for application in a dynamic environment because the inclusion of a single new image into the image set can require a complete recomputation of the basis set.

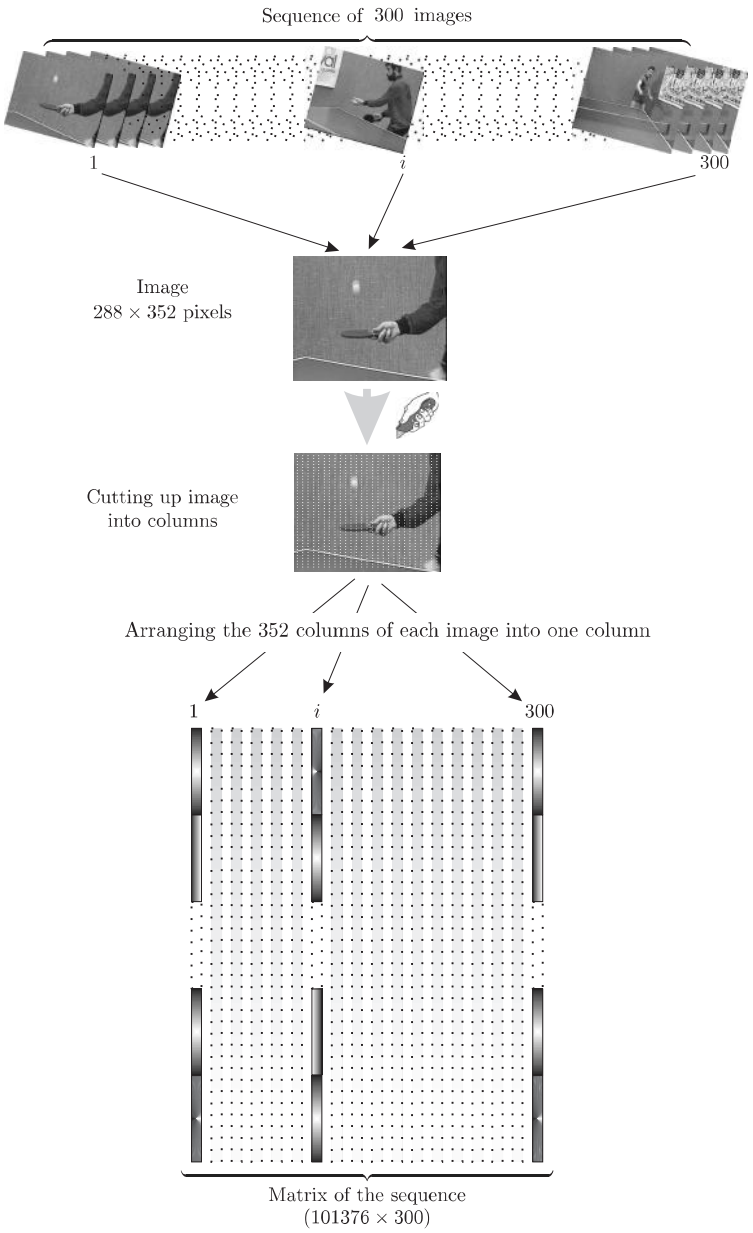


Figure 2.12: Transforming the sequence of images into a matrix.

Softwares and references

[1] Muruhan Tathinam and Linda R. Petzold, *A New Look at Proper Orthogonal Decomposition*, SIAM J. NUMER. ANAL. Vol. 41, No. 5, pp. 1893âĂŞ1925.

3

Software

This section contains a brief description of software developed and used by CICADA researches.

Contents

iGen

Description

By D. Tang

When we wish to investigate the behaviour of a complex physical system, such as the Earth's climate, it is often the case that an accurate, high-resolution computer model of the system could easily be written but that such a model would execute far too slowly to be of any practical use. In this case, we must create a 'parameterised' model that closely approximates the high-resolution model while requiring much lower computational resources.

iGen is a program that generates parameterised models from high-resolution models and supplies formal bounds on the error between the models. To create a parameterised model, we begin with a model whose resolution is high enough to resolve all the physical processes of interest. iGen takes the source code of this high-resolution model as its input and, rather than execute it, iGen analyses the structure of the code, applies appropriate approximations, derives the source code of a faster model and reports bounds on the error between the parameterised model and the high-resolution model. Because the parameterisation is formally derived from the high-resolution model, with bounded error, its output can be used to justify conclusions about the behaviour of the high-resolution model, and so about the physical system of interest.

iGen can also be used as a tool for exploring the behaviour of physical systems. Traditionally, a numerical experiment would consist of executing a model with one or more input values to get one or more corresponding output values. iGen offers an alternative type of numerical experiment in which the model is formally analysed to extract functional relationships between large-scale observables. If we are trying to understand the emergent behaviour of a complex system, a formally derived functional relationship between observables is of much more immediate use to us than a set of input/output pairs. For example, if we propose a theory that predicts some relationship between observables, then iGen's analysis of the high-resolution model could supply us with a formal derivation of this relationship from the underlying equations of motion. Alternatively, we may not yet have a fully formed theory, but the discovery of a simple functional relationship between observables would be a valuable piece of evidence that informs and directs our theory building. Further details of iGen's mechanism and application can be found in Tang and Dobbie (2011a, 2011b).

Example: Automatic derivation of the Lorenz equations.

As an illustration, iGen was used to analyse a model of Rayleigh-Benard convection in a 2-dimensional, laminar, incompressible fluid on an 80×28 grid. The model was wrapped so that its input consisted of three variables (X,Y,Z). These were converted to an initial state of the fluid according to

$$\psi(x,z) = \frac{\sqrt{2}(1+a^2)}{a} X \ \sin(\pi a x) \ \sin(\pi z)$$

and

$$\theta(x,z) = \frac{\sqrt{2} Y \ \cos(\pi a x) \ \sin(\pi z) - Z \ \sin(2\pi z)}{\pi r}$$

where $\psi(x,z)$ is the stream-function, $\theta(x,z)$ is the temperature perturbation, $a = \frac{1}{\sqrt{2}}$ is the aspect ratio of the convective cells and $r = 28$ is the non-dimensionalised Rayleigh number. Similarly, the output of the high-resolution model was converted back to the (X,Y,Z) phase space by extracting the appropriate lowest modes of ψ and θ according to

$$X = \frac{1}{\sqrt{2}(1+a^2)} \int \int \psi(x,y) \ \sin(\pi a x) \ \sin(\pi z) \, dx \, dz$$

$$Y = \frac{\pi R}{\sqrt{2} a} \int \int \theta(x,y) \ \cos(\pi a x) \ \sin(\pi z) \, dx \, dz$$

$$Z = -\frac{\pi R}{2a} \int \int \theta(x,y)\ \sin(2\pi z)\ dx\ dz$$

The final output of the wrapped model was the average rate of change of X, Y and Z over a 0.00001s simulation.

The variables, (X,Y,Z), correspond to the variables of the Lorenz equations (Lorenz '63), which describe a 3-variable parameterisation of Rayleigh-Benard convection. iGen analysed the wrapped model of Rayleigh-Benard convection and produced the following simplified code:

```
input(x,y,z)
  dx_dt = 9.95076*y + 9.94443*x
  dy_dt = -0.991175*x*z - 0.999187*y + 27.9712*x
  dz_dt = -2.65625*z + 0.997019*x*y
output(dx_dt, dy_dt, dz_dt)
```

which is a statement of the Lorenz equations with a slight difference in the constants. This represents an increase in execution speed of 5 orders of magnitude over the original model. The slight difference between iGen's output and the Lorenz equations is attributed to the finite resolution of the grid, the finite time over which the integration was performed and the accuracy of the algorithm used to solve the Poisson equation in the Rayleigh-Bernard model.

Softwares and references

[1] Lorenz, E. N., 1963: Deterministic nonperiodic flow, J. Atmos. Sci., 20: 130–141.

[2] Tang, D. F. and Dobbie, S., 2011a: iGen 0.1: a program for the automated generation of models and parameterisations, Geosci. Model Dev., 4: 785–795, doi:10.5194/gmd-4-785-2011.

[3] Tang, D. F. and Dobbie, S., 2011b: iGen 0.1: the automated generation of a parameterisation of entrainment in marine stratocumulus, Geosci. Model Dev., 4: 797–807, doi:10.5194/gmd-4-797-2011.

COMPUTATIONAL STEERING

Description

J. Brooke

Computational steering (CS) is an approach to maximizing the usefulness of large scale computational resources: it involves user intervention in a large computation, while it is still running, with the object of enhancing the usefulness or interest of the output. In CS, the current state of the computation is periodically assessed against certain user-supplied criteria, and on that basis changes are made before the computation is allowed to proceed. Intervention could be done interactively by the user, or automatically by some separate steering system. CS can be a tool in the exploratory, effectively experimental study of high-dimensional systems and can help to answer questions as to how these can be described as modelling the dynamics of infinite dimensional systems. This is of practical relevance to applied science and engineering as simulation is increasing replacing experiment in design and investigation of systems.

Control of a computational system, and CS in particular, requires some way of deriving relevant information about the current state of the computation, and of representing the information so that it is accessible to the optimizer/steerer. The quantities of interest (in CS these are referred to as the *monitoring space* may not correspond to the computational variables, (and may indeed change as the computation proceeds). In examples such as drag minimization in CFD determining these quantities is automatic and can be incorporated into the system; for interactive steering, as mentioned above, some sort of visualization might be appropriate, and we may or may not want to construct this visualization for each state through which the system passes. At an abstract level the monitoring space is the range of a mapping from the computation state space, and since the computation typically involves a much larger number of variables than the monitoring space we may loosely describe the latter as low dimensional.

Though we have particular interest in the quantities constituting the monitoring space, they are not under direct control; instead, intervention is restricted to some of the computational variables, typically a subset of the parameters of the system, these comprise the *control space*. Efficient optimization or steering may require us to know, or learn, something about the relationship between control and monitoring spaces.

Computational steering has many similarities to the analysis (including experimental analysis) and control of nonlinear dynamical systems. The computation itself may clearly be viewed as a dynamical system, and observation functions on the state space of a dynamical system play much the same role as the monitoring variables.

Computational steering and modelling of large-scale systems

The behaviour of complex systems such as fluids, fields (e.g. electromagnetic or graviational, biomolecules, is increasingly being investigated via numerical simulation since, typically, we cannot express solutions of the governing equations in closed analytic form, especially when the equations are nonlinear. The mathematical description of such systems has an infinite-dimensional phase space (e.g. systems of PDEs) and the numerical discretization required to solve the equations of such a system lead to phase spaces with millions of degrees of freedom (for very large calcuations on supercomputers). In investigating the behaviour of the solutions, we can often identify a set of key parameters that can bring about changes in behaviour of solutions and these parameters form an auxillary space L which is of much lower dimension than the solution space, e.g. R^k with k a small positive integer. If we denote the original system by S, its discretization by S' we can represent the time evolution of the system as

$$dX/dt = f(X,\lambda,t), X \in S', \lambda \in L.$$

It is often the case that the set of parameters λ represent some uncertainty in our knowledge of the system and we may wish to observe the behaviour of solutions as λ is varied. Recent developments in computational science have lead to the concept of controlled steering of numerical simulations by altering the parameters in a running simulation This may be done in a collaborative manner with teams of investigators performing numerical experiments and tracing multiple paths through parameter space (see Figure 3.1).

In order to interpret the behaviour of the system during such experiments, it needs to be monitored. This may be done by visualizing the behaviour of the numerical solution or by monitoring key measures such as energy, vorticity, temperature, or by sampling the full solution in some reduced way. These methods lead to a new space to express the monitoring and as the full system S evolves so we observe evolution in the monitoring space which we call M.

Data Structures in Computational Steering

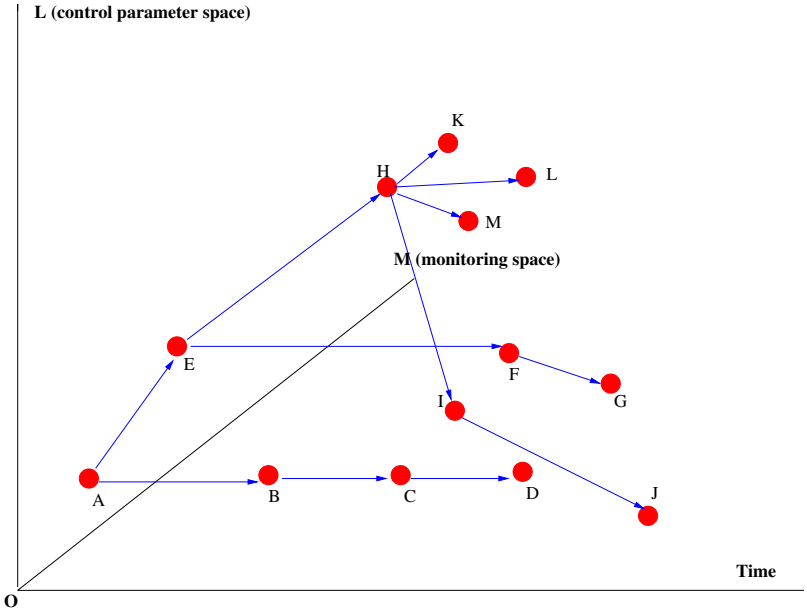


Figure 3.1: Development of a tree structure in a steered computation. The horizontal axis represents time, the vertical axis represents the control space (which may be of higher dimension) and the third axis represents the monitoring space (also may be of higher dimension). The red filled circles are the nodes reached by the evolution of the computation in both time and in the changes in control parameter values. The blue arrows represent particular computational paths, they will not necessarily be linear. See text for fuller explanation

The observations of the system are conceptualized as mappings $S' \to M$. Now the practical mechanics of running very large calculations means that the state of the system needs to be saved at given points in the time evolution. This can be because batch queues impose time limits or because of the possibility of machine failure. These "checkpoints" of the state will involved a representation of the instantaneous state in S' and some representation of the history of the parameter changes, i.e. the trajectory in the state space $S' \otimes L$. This is represented in Figure 3.1 where the checkpoints are shown as nodes and the piece-wise trajectories between the checkpoints as arrows (they are not necessarily reversible). The structure is thus a directed graph. We can have trajectories such as $A \to B \to C \to D$ that represent time evolution with no changes in parameter values, while all the other trajectories illustrated involve evolution in L also due to the effects of steering. The importance of history is shown by nodes D, I and F which represent the same point in L but not in $S' \otimes L$. Such computational steering requires massive computational resources and is a prime use of the distributed computing known as grid computing. The advantage of the structure shown in Figure 3.1 is that it provides the basis for extended investigations, each node can be the starting point for a new numerical experiment. Thus we move from the study of particular equations to the study of systems of equations where the control space allows for techniques such as optimisation, comparison with experimental data and experiments with the modelling of the

Computational Steering Software

The Reality Grid Steering library, free computational steering library, is available at

`http://code.google.com/p/computational-steering/`

The library has interfaces for programmers (Application Programmer Interfaces, API) for the major languages used in numerical programming, Fortran and C/C++. There are also APIs for Java, Python and Perl via wrapper interfaces. The web pages also provide reference applications showing how to call the steering library. It provides clients that can allow programs to be run remotely on very large computing resources but with all the control and monitoring being done via a user's laptop or mobile device. This allows the library to be deployed in a very flexible manner. The whole simulation can be run and controlled on a local device or can be run on remote resources and controlled locally. It is also possible for teams of researchers to control simulations in a collaborative manner, for example via the Access Grid (http://www.accessgrid.org). For further technical details see [2] original system S as well as S' e.g. via remeshing or adaptive meshing.

A concrete example

Figure 3.2: Development of steering as a trajectory through a one-dimensional parameter space. The horizontal axis shows the number of time steps and the vertical axis is the one-dimension lattice representing the values of the control parameter. States are marked as nodes on the trajectories and their structure is shown for each node (Figure courtesy of P.V. Coveney).

To illustrate the functionality of the software we describe its use in the computational modelling of a "mesophase" appearing in an oil-water mixture to which a quantity of surfactant has been added [1]. The surfactant creates boundaries between the oil and water phases that can self organise into surfaces with different geometrical structure. A particularly interesting mesophase is the gyroid phase [3] where the minimal surface is triply periodic, embedded (no self-intersections) and has no mean curvature although it possesses curvature everywhere. In Figure 3.2 we show a concrete example of the trajectories of Figure 3.1

from a simulation which uses Lattice-Boltzmann methods rather than discretization of partical differential equations. The nodes represent different spatial structures that change at bifurcation values. images showing the different structures that emerge at the bifurcations. Computational steering can be used to detect bifurcations and can be used to reconstruct the response of the system to parameter change. For an example of such exploration in a system governed by magnetohydrodynamic flow see[4]

Softwares and references

[1] Chin, J. and Harting, J. and Coveney, P.V. and Porter, A.R. and Pickles, S.M., *Steering in computational science: mesoscale modelling and simululation*, J. Contemporary Physics, V. 44, 2003, pp. 417-434.

[2] Pickles, S. M. and Haines, R. and Pinning, R. L. and Porter, A. R., *A Practical Toolkit for Computational Steering*, J. Philosophical Transactions of the Royal Society A, V. 363, 2005, pp. 1843-1853.

[3] Giupponi, G.and Harting, J. and Coveney, P.V., *A Emergence of rheological properties in lattice Boltzmann simulations of gyroid mesophases*, Journal of Physics: Conference Series, V. 73, 2006, pp. 533-539.

[4] A Salhi and J M Brooke, *A Detecting bifurcations in numerical simulation of fluid flow*, http://stacks.iop.org/1742-6596/216/i=1/a=012003, V. 216, N. 1, 2010.

iRRAM FOR REACHABILITY ANALYSIS

Description

By N.Muller & M.Korovina

One basic aspect of the hybrid systems is their evolution in time, *i.e.* the computation We have developed an algorithm and implemented using the iRRAM package [2] for *efficient and arbitrarily precise solutions* of the underlying IVPs up to the area where discontinuous jumps appear. There are two reasons for using high precision solutions: firstly, low precision might lead to incorrect assumptions about the location of these jumps points; and secondly – perhaps unexpectedly – high intermediate precision can sometimes increase the efficiency.

To illustrate the second aspect, consider the well-known Runge-Kutta methods used for the solution of IVPs. These methods are of fourth order, *i.e.*, the error depends on a bound on higher derivatives of the solution as well as on the fourth power of the step width. Although they are a reasonable choice if applied to double precision numbers, they will not always be optimal for higher precision solutions. As the step width has to be chosen according to the desired precision of the solution, methods with a fixed order lead to the number of steps growing exponentially in the number of bits of the solution. If the order can be chosen dynamically and arbitrarily high, significantly fewer steps associated with a much bigger step width are possible, which can lead to a polynomial time complexity.

Prototypical implementation.

We used the iRRAM package to implement a prototype for the algorithm proposed in [1], whose core structure uses dynamically constructed functions of types `FUNCTION<int,vector<REAL> >` (for sequences of real vectors) and `FUNCTION<REAL,vector<REAL> >` (for vector-valued functions on the real numbers). The implementation of such function objects in an imperative language like `C++` has been described in [3], it is based on a lazy evaluation technique. Thus we avoid the necessity to implement explicit (and computationally very expensive) representations for functions and sequences.

Using corresponding constructors

- `a=ivp_solver_simple (w,F)` yielding a vector power series `a` for flow conditions `F` and an initial condition `w` implementing equation

$$a_{\nu,\ell+1} = \frac{1}{\ell+1} \sum_{\substack{n_1,n_2,\ldots,n_d \in \mathbb{N} \\ n_1+\ldots+n_d=\ell}} \sum_{\substack{i_1,i_2,\ldots,i_d \in \{0,1\} \\ i_1+\ldots+i_d \le 1}} \left[c_{\nu,0,i_1,\ldots,i_d} \cdot a_{1,n_1}^{(i_1)} \cdot \ldots \cdot a_{d,n_d}^{(i_d)} \right] . \tag{3.1}$$

  ,

- `f=taylor_sum(a,R,M)` yielding the (vector-valued) sum function `f` for a (vector-)power series `a` and corresponding radius `R` and bound `M`, and

- `w=f(bs)` evaluating `f` at `bs` with `bs<R` as a limit using equation

$$\begin{aligned} \Big| \sum_{k=n+1}^{\infty} a_k \cdot z^k \Big| &\le \sum_{k=n+1}^{\infty} |a_k| \cdot |z^k| \le \sum_{k=n+1}^{\infty} M \cdot R^{-k} \cdot |z^k| \\ &\le M \cdot \left(\frac{|z|}{R}\right)^{n+1} \cdot \sum_{k=0}^{\infty} \left(\frac{|z|}{R}\right)^{k} = \frac{M \cdot R}{R-|z|} \cdot \left(\frac{|z|}{R}\right)^{n+1} . \end{aligned} \tag{3.2}$$

  to control the truncation error,

the core of the implementation is essentially just the following loop of 'big steps' interspersed with 'small steps' as mentioned in the previous section:

```
do { // big steps
  a = ivp_solver_simple (w,F);
```

```
   ... compute R,M ...
 f = taylor_sum(a,R,M);
 do { // small steps
      ... compute a step size from the distance to the guard ...
      ... accumulate the step size in a variable s  ...
      ... evaluate f(s) ...
      ... try whether a sufficiently good approximation has been found ...
      ... if yes: stop ...
 } until ( s is large enough for a big step )
 w= f(s)
}
```

As the evaluation of `f(s)` is a core part here, it is important to get a reasonably efficient implementation of the Taylor summation. The existing limit operators in the iRRAM package were not fitting, as they were not yet applicable to `FUNCTION` objects (they were essentially only usable for predefined algorithms). Additionally, the general heuristic of the iRRAM (that tries to compute limits with the maximal used precision) lead to an enormous waste of time; a new limit operator based on (3.2) had to be added. All other necessary operations were already present in the published version of the package.

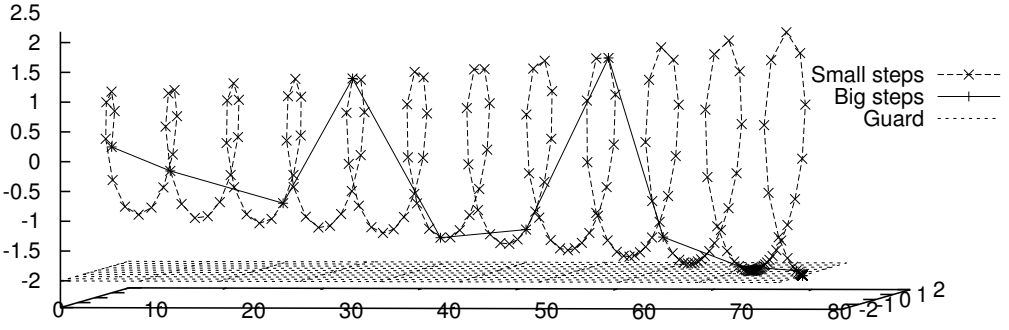
Example:

$$\dot{y}_1(t) = y_2(t) \quad ; \quad \dot{y}_2(t) = -y_1(t) + 0.02 \cdot y_2(t) \quad \text{with} \quad t_0 = 0, \overline{w}_0 = (0,1) \; .$$

Without the term $c_{1001} = 0.02$, the solution $\overline{y}$ would simply be the pair $(\sin, \cos)$; with the additional term we still have an oscillation, but with a growing amplitude.

As guard set we chose $G = \{(t, x_1, x_2) \in \mathbb{R}^3 \mid x_1 \leq -2\}$. Here the question was simply to approximate the first t_G where $y_1(t_G) = -2$ (we found $t_G \approx 73.5422061995...$). The size s_1 of the small steps was chosen as $s_1 = min\{\frac{\Delta(t_0)}{U}, \frac{R}{2}\}$ where $U = U_F((t_0, \bar{w}_0), \delta, \varepsilon)$, $R = min\{\delta, \frac{\varepsilon}{U}\}$; whenever the accumulated small steps grew larger than $\min\{\sqrt[4]{s_1} \cdot \sqrt{R}, \frac{R}{2}\}$, a big step was made. This bound of the big steps was chosen heuristically as an attempt to match the much higher complexity of the IVP solution at big steps with the more frequent Taylor summations at small steps.

The following graph shows an 3d-plot of the resulting trajectory constructed from the (linearly interpolated) points $(t_i, \overline{w}_i)$.



The following table shows a few results of computations with this IVP. Its interpretation is as follows: To approximate t_G with an error of at most 2^{-n}, the software chose a working precision of 2^{-p}, using b big steps (re-evaluations of the IVP), s small steps (evaluations of the Taylor sum) with a maximal index of ℓ_{max} (working with an order of ℓ_{max}) and took time t (on an AMD Athlon 64X2 Dual Core Processor 4600+).

result bits n	working bits p	#big steps b	#small steps s	ℓ_{max}	time t
20	242	9	223	108	$0.271s$
50	242	10	283	108	$0.298s$
100	242	10	384	108	$0.297s$
1000	1332	12	2200	430	$5.42s$
10000	11787	14	20361	3506	$308s$

To compare our results we used the IVP solvers from the popular high-level language octave, that is primarily aiming at numerical computations (www.gnu.org/software/octave), in order to solve the above IVP. We applied them just to approximate the trajectory starting from $t_0 = 0$ up to 73.543. Only between 73.542 and 73.543 we tried to find the point where it dropped below -2 (without even trying to verify that this was the first solution). Within a few milliseconds, a naive application of the solver gave a result near $t'_G = 73.54225$. As only 6 decimals were in common with our result of $t_G = 73.5422061995...$, we tried less naive ways, which initially produced the same result t'_G. Being convinced from the correctness of our own implementation, we continued playing with the octave solver; with further variations of its parameters applied in a quite elaborate way, we were able to get results different from both t_G and t'_G. The best combination we found resulted in 73.542208, now with 7 correct decimals, but within $0.7s$ of computation time. As the results from the octave solver quite erratically jumped around 73.5422 with further variations of the parameters, we believe that more than 6 decimals precision cannot reliably be expected. Our conclusion from these experiments is that solving IVPs might be an area where exact real arithmetic can actually compete with ordinary double precision arithmetic in terms of speed and precision.

To illustrate the effect of varying distances $|t_G - t_0|$ on our algorithm, we removed the perturbating coefficient 0.02. Additional we chose the guard set $G_\eta = \{(t,x_1,x_2) \in \mathbb{R}^3 \mid t \geq \eta\}$ for a given η and just printed 9 leading decimals of $y_1(t_{G_\eta})$. As $t_{G_\eta} = \eta$, this setting transformed our algorithm into a slow (but still exact) method to compute $\sin(\eta)$. The results in the following table show that further reductions in the error propagation in our software are necessary before it can really be applied for larger ranges of η. Again we compared our results with the octave IVP solver, which was significantly faster for larger η but had problems with its precision again.

η	$\sin(\eta)$	our implementation			octave	
		working bits p	#big steps b	time t	time	result
10	-0.544021110	136	2	$0.02s$	$0.007s$	$-0.5440211'86$
100	-0.506365641	242	10	$0.2s$	$0.06s$	$-0.50636'2329$
1000	0.826879540	1737	95	$17.5s$	$0.55s$	$0.8268'84089$
10000	-0.305614388	14807	681	$3706s$	$5.3s$	$-0.305'931729$

Softwares and references

[1] Norbert Th. Muller, Margarita V. Korovina, *Making big steps in trajectories*, In proc. CCA 2010, Electronic proceedings in theoretical Computer Science, v. 24, 2010, pp. 106-119.

[2] Norbert Th. Muller, *The iRRAM: Exact Arithmetic in C++*, Lecture notes in computer science, v. 2991, 2001, pp. 222–252.

[3] Norbert Th. Muller, *Enhancing imperative exact real arithmetic with functions and logic*, available at http://www.uni-trier.de/ mueller.

TOOLS FOR MODELING, SIMULATION AND VERIFICATION OF HYBRID SYSTEMS

Description

By Y. Chahlaoui

In general, there is no single tool covers all the needs of designers that use hybrid system as models to solve their problems. Each tool or language that has been proposed over the years to handle hybrid systems is based on somewhat different notions and on assumptions that make a fair comparison difficult. Moreover, sharing information among tools is almost impossible until now, so that the community cannot leverage maximally the substantial amount of work that has been directed to this important topic. Here, we give a very short survey of available languages and tools that have been proposed in the past years for the design and verification of hybrid systems. We give a short comparative summary of some of these tools for simulation and verification. For a more complete review and comparison see[1]. Table 3.1 lists tools and languages with information on the institution that supports the development of each project as well as pointers to the corresponding web site. Table 3.2 summarizes the distinctive features of the various modeling and design environments, programming languages, simulators and tools for hybrid systems.

Table 3.1: References for the various modeling approaches, toolsets.

Name	Institution	Web Page
CHARON	Univ. of Pennsylvania	www.cis.upenn.edu/mobies/charon
CHECKMATE	Carnegie Mellon Univ.	www.ece.cmu.edu/~webk/checkmate
d/dt	Verimag	www-verimag.imag.fr/~tdang/Tool-ddt/ddt.html
DYMOLA	Dynasim AB	www.dynasim.se/
ELLIPSOIDAL TOOLBOX	UC Berkeley	www.eecs.berkeley.edu/~akurzhan/ellipsoids
HSOLVER	Max-Planck-Institut	www.mpi-inf.mpg.de/~ratschan/hsolver
HYSDEL	ETH Zurich	www.control.ee.ethz.ch/~hybrid/hysdel
HYTECH	Cornell, UC Berkeley	www-cad.eecs.berkeley.edu/~tah/HyTech
HYVISUAL	UC Berkeley	ptolemy.eecs.berkeley.edu/hyvisual
MASACCIO	UC Berkeley	www.eecs.berkeley.edu/~tah
MODELICA	Modelica Association	www.modelica.org
PHAVER	VERIMAG	www.cs.ru.nl/~goranf
SCICOS	INRIA	www.scicos.org
SHIFT	UC Berkeley	www.path.berkeley.edu/shift
SIMULINK	The MathWorks	www.mathworks.com/products/simulink
STATEFLOW	The MathWorks	www.mathworks.com/products/stateflow
SYNDEX	INRIA	www-rocq.inria.fr/syndex

Other softwares classified on basis of the primary focus of the package are given below.

Repositories

ESCHER http://www.escherinstitute.org is an independent, non-profit research institute dedicated to the transition of government-sponsored information-technology out of the research environment and into practical use by industrial and government end users.

Mathtools http://www.mathtools.net is a technical computing portal for all scientific and engineering needs. The portal is free and contains over 20,000 useful links to technical computing programmers, covering MATLAB, JAVA, EXCEL, C/C++, FORTRAN and others.

Table 3.2: Nature, main purpose and features of the various languages, modeling approaches, and toolsets.

Name	Nature	Main Purpose	Additional Features
CHARON	modeling language	formal semantics for hierarchy, concurrency, refinement	simulator, type checker, interface to JAVA
CHECKMATE	verification tool	formal semantics (TEDHS) for simulation and verification	integrated with MATLAB SIMULINK/STATEFLOW
d/dt	verification tool	safety verification of hybrid systems with linear continuous dynamics	synthesis of safe switching controllers
DYMOLA	modeling and simulation	multi-engineering modeling and simulation	can build more integrated models using Modelica
ELLIPSOIDAL TOOLBOX	analysis	implementing in MATLAB ellipsoidal calculus and its application to the reachability analysis of continuous- and discrete-time systems	possibly time-varying linear systems, and linear systems with disturbances, for which ET calculates both open-loop and close-loop reach sets
HSOLVER	verification tool	safety verification of hybrid systems	accepts non-linear input constraints
HYSDEL	modeling language	modeling of discrete-time affine dynamical systems	generation of input for MATLAB simulation
HYTECH	symbolic model checker	modeling and verification of linear hybrid automata	support for parametric analysis
HYVISUAL	visual modeler	modeling and simulation of hybrid systems, hierarchy support	PTOLEMY II-based block-diagram editor
MASACCIO	formal model	support for concurrent, sequential, and timed compositionality	enables assume-guarantee reasoning
MODELICA	modeling language	object-oriented modeling of heterogeneous physical systems	MODELICA standard library, commercial tools
PHAVER	verification tool	safety verification of affine hybrid systems	support for equivalence/refinement between hybrid automata
SCICOS	hybrid system toolbox	modeling and simulation of hybrid systems	C code generation, interface to SYNDEX
SHIFT	programming language	modeling of dynamic networks of hybrid automata	C code generation, λ-SHIFT for real-time control
SIMULINK	interactive tool	analysis and simulation, hierarchy support, model discretizer	MATLAB-based, library of predefined blocks
STATEFLOW	interactive tool	FSM, statechart formalism, hierarchy support	chart animation, debugger
SYNDEX	system-level CAD	real-time code generation, distribution and scheduling	HW/SW codesign support, formal verification

Modeling

DSM Forum http://www.dsmforum.org exists to spread the knowledge and know-how of Domain-Specific Modeling (DSM). DSM raises the level of abstraction beyond programming by specifying the solution directly using domain concepts. It is an independent body made up of the leading DSM tool and solution providers, along with expert DSM users.

Simulation

ABACUSS http://yoric.mit.edu/abacuss2 (Advanced Batch And Continuous Unsteady-State Simulator), developed for chemical engineering systems, supports hybrid models, model inheritance, hierarchical model decomposition. It facilitates guaranteed state event location, batch process simulation, solution of high-index differential algebraic equations, dynamic and steady-state optimization, and dynamic sensitivity and uncertainty analyis.

DAEPACK http://yoric.mit.edu/DAEPACK is a software library for general numerical calculations. It is divided into two major libraries: symbolic analysis and transformation and numerical calculation. The symbolic analysis and transformation library consists of components for analyzing general Fortran-90 models and automatically generating the information required when using modern numerical algorithms, e.g., (i) sparsity pattern generation, (ii) discontinuity locking, and (iii) automatic differentiation.

AnyLogic http://www.xjtek.com/products/anylogic is a professional virtual prototyping environment. It enables you to rapidly build a simulation model of the system under development and its environment, including physical objects and human users. The modeling technology is based on UML-RT, Java, and algebraic-differential equations. AnyLogic offers a range of domain-specific libraries.

BaSiP is developed for simulation of recipe-driven production in complex multi-purpose batch plants.

DOORS aims at the creation of a prototype for a distributed, real-time simulator, the design process of mechatronic systems to support hardware-in-the-loop test.

gPROMS http://www.psenterprise.com/gproms/index.html represents the state-of-the-art in process modelling, simulation and optimisation technology.

HYBRSIM http://msdl.cs.mcgill.ca/people/mosterman/papers/icbgm99a/node2.html is an implementation of a hybrid bond graph modeling and simulation tool. It embodies a set of physical principles that govern discontinuous changes in physical system models. MODEL VISION http://www.xjtek.com is an object-oriented environment for the design of large dynamic systems that features: (i) Supporting B-Charts (UML-compatible statecharts integrated with differential equations) to specify hybrid behavior, (ii) Numerical methods from ODEPACK and Hairer-Norsett-Wanner collection, (iii) Supports matrix and vector data types, (iv) Includes standard device class libraries and enables the user to create his own, (v) Animation libraries and wizards for rapid creation of animated sketches, (vi) Generates complete portable Win32 and Java executable models supporting automation.

OmSim is an environment for modelling and simulation based on Omola. Omola is an object-oriented language for modelling of continuous time and discrete event dynamical systems.

SHIFT http://gateway.path.berkeley.edu/SHIFT is a programming language for describing dynamic networks of hybrid automata. Such systems consist of components which can be created, interconnected and destroyed as the system evolves. Components exhibit hybrid behavior, consisting of continuous-time phases separated by discrete-event transitions.

Smile is a simulation tool for energy systems. A ZimOO (an object-oriented specification language for hybrid systems in which continuous aspects are modeled by differential equations) specification of a simulation model serves as the basis for implementing the model in the simulation language Smile.

20-SIM http://www.20sim.com ("Twente Sim") is a modeling and simulation program that runs under Microsoft Windows and Sun-Unix. With 20-sim you can simulate the behavior of dynamic systems, such as electrical, mechanical and hydraulic systems or any combination of these systems.

Verification

KRONOS http://www-verimag.imag.fr/DIST-TOOLS/TEMPO/kronos is a tool developed with the aim to verify complex real-time systems. Components of real-time systems are modeled by timed automata and the correctness requirements are expressed in the real-time temporal logic TCTL. KRONOS checks whether a timed automaton satisfies a TCTL-formula.

MOBY http://theoretica.informatik.uni-oldenburg.de/~moby comprises a graphical editor for (i) PLC-Automata, a formal description technique for real-time systems, (ii) SDL-Specifications, and (iii) Object Z-Specifications. These specifications can be used for model checking (based on timed automata) and (graphical) simulation (based on high level Petri nets).

Spin http://spinroot.com/spin/whatispin.html is a widely distributed software package that supports the formal verification of distributed systems. The software was developed at Bell Labs in the formal methods and verification group.

STeP http://www-step.stanford.edu the Stanford Temporal Prover, is being developed by the REACT research group to support the verification of reactive, real-time and hybrid systems based on their temporal specification. STeP is not restricted to finite-state systems, but combines model checking with deductive methods to allow the verification such systems as parameterized (N-component) circuit designs, parameterized (N-process) programs, and programs with infinite data domains.

UPPAAL http://www.uppaal.com is a tool suite for validation and verification of real-time system modelled as networks of timed automata extended with data variables. The tools have WYSIWYG interfaces and features: graphical editing, graphical symbolic simulation and symbolic verification.

Verdict is the name of a computer tool for formal verification of discrete controllers for continuous processes, as they are, for example, in processing plants often encountered. The formal verification is a process by which a mathematical proof can be provided on a given that control a specific industrial process in every conceivable situation influenced so that a separate description given requirement is met to the desired process behavior.

IAR visualSTATE http://www.iar.com/website1/1.0.1.0/371/1/ is a suite of tools, that supports you all the way through the software development process in an iterative and interactive specification process. You rapidly create a virtual prototype of an an outline of your product that can be evaluated and validated against the specification. visualSTATE automatically generates code for your target system and the Tester tools and techniques include interactive simulation, real-link for in-target testing and complete dynamic formal verification.

References

[1] Carloni, Luca P. and Passerone, Roberto and Pinto, Alessandro and Sangiovanni-Vincentelli, Alberto L., *Languages and tools for hybrid systems design. Print version of Foundations and Trends in Electronic Design Automation Vol. 1, No. 1-2*, Boston, MA, 2006.

4

Contributors

We are grateful to the following people for contributing to the collection.

- John Brooke
- Martin Brown
- Marius Bujorianu
- Manuela Bujorianu
- Younes Chahlaoui
- Houman Dallali
- Gareth Jones
- James Hook
- Paul Glendinning
- Marc Goodfellow
- Margarita Korovina
- Piotr Kowalczyk
- Norbert Muller
- Daniel Tang
- Cris Welshman

1 Contributing to the Collection

Contributions of suggested new problems for the collection are welcome. The following rules should be followed when providing new problems.

Write a latex file called `problemname.tex`, where `problemname` is the proposed name of your example, describing the problem. The `tex` file should consist of a problem environment, with first line stating the relevant identifiers for the problem.

```
\problem{\textsc{Problem Name}}

\descrip{Description}{A short description that anyone can understand,
     like in textbooks (1st year)}

\descrip{Formalization}{we can consider different points of view
     \begin{itemize}
     \item Hybrid automata
     \item Fillipov system
     \item Others
     \end{itemize}
}

\descrip{Properties}{Classical properties already known (figures)}

\renewcommand*{\bibname}{} % This will define heading of bibliography to be empty, so you can...
\descrip{Softwares and references}{ \vspace{-11mm}     % ...place a normal section heading before the
\nocite{*}
\bibliographystyle{plain}

\begin{thebibliography}{9}
   \bibitem{XXX} XXX
\end{thebibliography}}
```

Index