Efficient Simulation of Large Scale Models.

Quantized State Systems Algorithms

Ernesto Kofman

CIFASIS–CONICET
Departamento de Control – FCEIA – UNR.

21/11/2019

Outline

Outline

Outline

State Space Representation of ODEs

Lumped models coming from different domains (physics, chemistry, engineering, economics, population dynamics, etc.) are usually represented as sets of ODEs of the form:

$$\begin{aligned}
\dot{x}_1(t) &= f_1(x_1(t), \cdots, x_n(t), t) \\
\dot{x}_2(t) &= f_2(x_1(t), \cdots, x_n(t), t) \\
&\vdots \\
\dot{x}_n(t) &= f_n(x_1(t), \cdots, x_n(t), t)
\end{aligned} \tag{1}$$

where t represents the time, $x_i(t)$ are the state variables, and $\dot{x}_i(t)$ are the state derivatives with respect to the time.

State Space Representation of ODEs

These type of ODEs can be also the result of the spatial discretization of partial differential equations (PDEs).

$$\begin{aligned}
\dot{x}_1(t) &= f_1(x_1(t), \cdots, x_n(t), t) \\
\dot{x}_2(t) &= f_2(x_1(t), \cdots, x_n(t), t) \\
&\vdots \\
\dot{x}_n(t) &= f_n(x_1(t), \cdots, x_n(t), t)
\end{aligned} \tag{1}$$

where t represents the time, $x_i(t)$ are the state variables, and $\dot{x}_i(t)$ are the state derivatives with respect to the time.

State Space Representation of ODEs

The ODE system of Eq.(1) can be alternatively written using compact vector notation as:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), t) \tag{2}$$

where

$$\mathbf{x}(t) \triangleq [x_1(t)\, x_2(t), \cdots, x_n(t)]^T$$

is the state vector, for which we generally know an initial state:

$$\mathbf{x}(t_0) = \mathbf{x}_0 \tag{3}$$

Continuous System Simulation

In order to simulate a system represented by Eq.(2), the ODE must be solved from the initial state $\mathbf{x}_0$ obtaining the solution $\mathbf{x}(t)$ in some interval $t \in [t_0, t_f]$.

ODEs cannot (in general) be solved by analytical means.

For this reason, Numerical Integration Methods for ODEs are used in order to obtain approximate solutions.

Outline

Classic Numerical Integration Algorithms

Given a system

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), t)$$

with the known initial state $\mathbf{x}(t_0) = \mathbf{x}_0$, the goal of numerical integration methods is to obtain an approximate solution in some instant of time: $t_1, t_2, \cdots, t_N$ in the interval $[t_0, t_f]$.

$$\tilde{\mathbf{x}}_1 \approx \mathbf{x}(t_1),\ \tilde{\mathbf{x}}_2 \approx \mathbf{x}(t_2), \cdots, \tilde{\mathbf{x}}_N \approx \mathbf{x}(t_N),$$

The time interval $h_k \triangleq t_{k+1} - t_k$ is called step size, which can be either constant or variable. In the second case, it is automatically adjusted by step size control algorithms.

Classic Numerical Integration Algorithms

One-Step Methods

These are methods that compute $\mathbf{x}_{k+1}$ using only information on $\mathbf{x}_k$. (Runge–Kutta algorithms).

Example: Forward Euler (first order)

$$\mathbf{x}_{k+1} = \mathbf{x}_k + h \cdot \mathbf{f}(\mathbf{x}_k, t_k)$$

Multi-step Methods

These are methods that compute $\mathbf{x}_{k+1}$ using information on $\mathbf{x}_k$ and from some previous steps ($\mathbf{x}_{k-1}$, etc).

Example: 3rd order Adams-Bashford (AB3)

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \frac{h}{12}(23 \cdot \mathbf{f}_k - 16 \cdot \mathbf{f}_{k-1} + 5 \cdot \mathbf{f}_{k-2})$$

Classic Numerical Integration Algorithms

Implicit Methods

One-Step or Multi-step implicit methods use formulas involving future state information, what leads to implicit equations.

Example: Trapezoid Rule (2nd order)

$$\mathbf{x}_{k+1} = \mathbf{x}_k + 0.5 \cdot h \cdot [\mathbf{f}(\mathbf{x}_{k+1}, t_{k+1}) + \mathbf{f}(\mathbf{x}_k, t_k)]$$

Third Order Backward Difference Formulae (BDF3):

$$\mathbf{x}_{k+1} = \frac{18}{11}\mathbf{x}_k - \frac{9}{11}\mathbf{x}_{k-1} + \frac{2}{11}\mathbf{x}_{k-2} + \frac{6}{11}h \cdot \mathbf{f}_{k+1}$$

Stability

Implicit algorithm usually preserve numerical stability irrespectively of the step size.

Outline

Stiff Systems

These are systems containing simultaneous fast and slow dynamics.

Stiff systems require the use of implicit and variable step algorithms

Discontinuous Systems

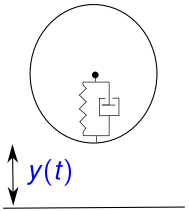
This is a simple bouncing ball model:

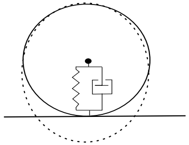
$$\dot{y}(t) = v(t)$$

$$\dot{v}(t) = \begin{cases} -g & \text{if } y(t) > 0 \\ -g - \frac{k}{m}\cdot y(t) - \frac{b}{m}\cdot v(t) & \text{if } y(t) \le 0 \end{cases}$$

Notice that the ODE is discontinuous at $y = 0$.

Numerical methods usually produce unacceptable errors when a step crosses a discontinuity. Thus, the discontinuity events must be detected and the simulation must be restarted from that point.





Large Scale Discontinuous Models

Suppose now that we want to simulate a system of N bouncing balls:

- The cost of computing function $\mathbf{f}(\mathbf{x}, t)$ growths linearly with N.
- The occurrence of discontinuities (bounces) also growths linearly with N.
- The time between successive discontinuities then diminishes with N, and so does the step size.
- The computational costs are then at least quadratic with N.

The problem of simulating particles crossing through different volumes has similar features.

Outline

Basic Idea

All classic numerical algorithms try to answer the following question:

Given that the state at time t_k is $\mathbf{x}(t_k)$, what would be the state value at time $t_k + h$?.

Quantized State Systems (QSS) algorithms are based on inverting the question:

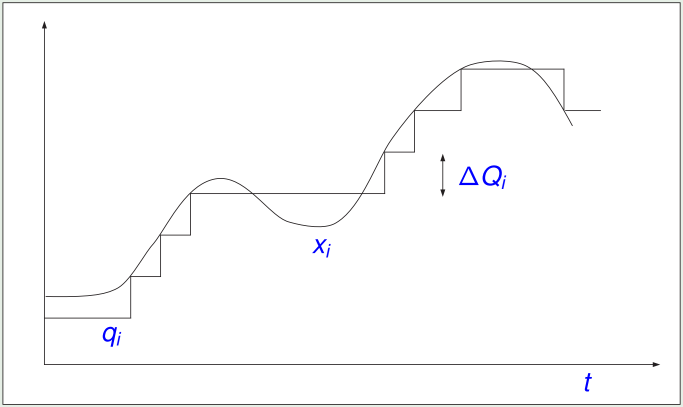
Given that the state at time t_k is $\mathbf{x}(t_k)$, when it is going to deviate from its current value by a quantity ΔQ?.

This question has a different answer for each state variable x_i and consequently, QSS algorithms are asynchronous.

Outline

QSS1 Method

Hysteretic Quantization Function



State $x_i(t)$, Quantized State $q_i(t)$, and quantum ΔQ_i in QSS1 method.

QSS1 Method

Definition

Given a system

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), t)$$

its QSS1 approximation is given by

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{q}(t), t)$$

where $\mathbf{q}(t)$ and $\mathbf{x}(t)$ are component-wise related by hysteretic quantization functions

- $\mathbf{q}(t)$ is the quantized state vector.
- Each hysteretic quantization function is defined by a parameter ΔQ_i called Quantum.

QSS1 Features

QSS1 offers some advantages over classic algorithms is certain cases:

- It only performs computations when a state experiences a significant change (intrinsic step–size control).
- The computations only involve the state that changes and those whose derivatives are affected by that change. (sparsity and local activity exploitation).
- Zero Crossing Function can be straightforwardly detected (efficient discontinuity detection).
- After the occurrence of a discontinuity, QSS1 only recomputes the state derivatives affected by that discontinuity (efficient discontinuity handling).

QSS1 Properties

QSS1 has strong theoretical properties:

- Convergence: The simulation error goes to zero as $\Delta Q \to 0$.
- Practical Stability: Provided that the original system is stable, the QSS1 solutions finish around the equilibrium point.
- **Computable Global Error Bound!**: In linear systems, the global error bound can be computed as a linear function of the quantization.

QSS – Features

Main advantages

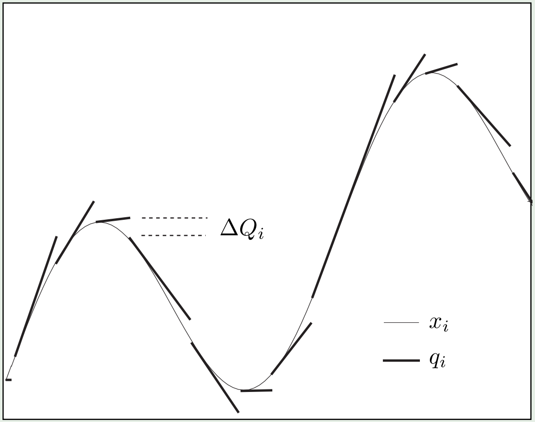
- Simulation of systems with local activity.
- Simulation of discontinuous systems.

Main disadvantages

- Spurious oscillations in stiff systems.
- The number of steps growths linearly with the quantum and the accuracy.

QSS2 Method

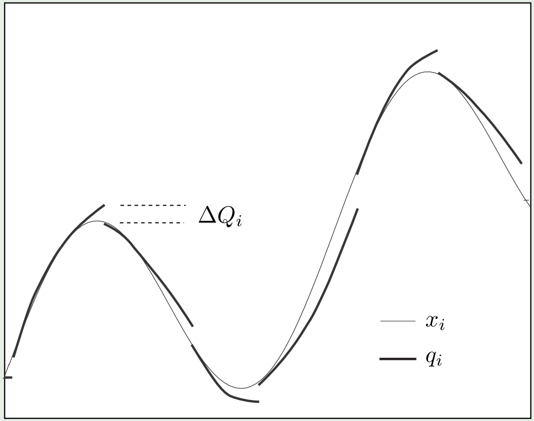
First order quantization



- Same advantages of QSS1.
- Second order accurate method.
- The number of steps growths with square root of the accuracy.

QSS3 method

Second order quantization



- Same advantages of QSS1.
- Third order accurate method.
- The number of steps growths with the cubic root of the accuracy.

QSS – Features

Main advantages

- Simulation of systems with local activity.
- Simulation of discontinuous systems.

Main disadvantages

- Spurious oscillations in stiff systems.
- The number of steps growths linearly with the quantum and the accuracy.

Linearly Implicit QSS Methods

- There are LIQSS methods of order 1, 2 and 3.
- These methods can efficiently integrate certain clases of stiff systems.
- LIQSS algorithms are very efficient for simulating power electronics systems and Advection-Diffusion-Reaction models.
- In spite of the word implicit, LIQSS can be explicitly implemented.

Outline

QSS Methods Implementation

There are two alternatives for QSS implementation:

1. Using DEVS simulation tools, which is simple but inefficient (most DEVS simulation tools have QSS libraries, with PowerDEVS being the most complete).
2. Using a direct approach, that is more involved as it requires structural information. The Stand–Alone QSS solver follows this idea and it automatically computes all the structural information that is necessary.

Parallelization

The Stand–Alone QSS Solver also includes a parallel implementation of the algorithms that is very efficient for multi-core simulation of large scale models.

Applications

Some applications where QSS methods have advantages over classic algorithms are the following ones:

- Systems involving Power Electronic Circuits.
- Large populations of heating / cooling systems (Energy Plus, Modelica).
- Some biological models, including Spiking Neural Networks.
- Advection–Diffusion–Reaction equations.
- High Energy Particle Physics? – Recent experiments with Geant4.

Outline

Outline

A Model of Several Particles

We consider a system of N particles moving in a 1D domain following Newton's laws:

$$\begin{aligned} \dot{x}_i(t) &= v_i(t) \\ m_i \cdot \dot{v}_i(t) &= F(t) - b_i(t) \cdot v_i(t) \end{aligned} \tag{4}$$

where

- $x_i(t)$ and $v_i(t)$ are the position and velocity of the i–th particle at time t.
- $b_i(t)$ models the (variable) friction experienced by the i–th particle, while m_i represents its mass.
- $F(t)$ is a constant force impulsing by all particles.

The domain is partitioned into M sections and the friction $b_i(t)$ of a particle traveling through certain region is proportional to the number of particles in that section.

Some Results

- We simulated the system with an increasing number of particles, from $N = 10$ until $N = 100,000$.
- We used DOPRI and LIQSS2 algorithms in the Stand Alone QSS Solver with the same accuracy settings.
- All the simulation where performed in a PC Intel i3 under Linux OS.

Particles	DOPRI (sec.)	LIQSS2 (sec.)
10	0.0009	0.0007
100	0.074	0.008
1,000	4.773	0.080
10,000	611.4	1.530
100,000	–	23.2

Outline

Conclusions

- QSS are alternative algorithms for ODE numerical integration.
- They can avoid the quadratic costs associated to the simulation of large scale discontinuous systems.
- Their use in HEP (for particle tracking simulation) is currently under study by Rodrigo Castro's group in collaboration with the Detector Simulation Group in Fermilab.
- We expect that this collaboration can be extended to other groups.
- We believe that specialized QSS algorithms (including specialized parallelization strategies) for particle tracking simulation can noticeably improve the existing results. However, their development requires an interdisciplinary approach.