GQLink

An implementation of Quantized State System (QSS) methods in Geant4

Lucio Santi$^{1,2}$, Federico Bergero$^3$
Soon Yung Jun$^4$, Krzysztof Genser$^4$, Daniel Elvira$^4$
Rodrigo Castro$^{1,2}$

$^1$Universidad de Buenos Aires  $^2$ICC-CONICET  $^3$CIFASIS-CONICET (Argentina)  $^4$FNAL (USA)

ACAT 2017, 21-25 August
University of Washington, Seattle, WA
Table of contents

1 Introduction

2 Background information
   • The Geant4 simulation toolkit
   • Quantized State System (QSS) methods and QSS Solver
   • Preliminary comparison between Geant4 and QSS Solver

3 Geant4 to QSS Link (GQLink): an implementation of QSS within Geant4

4 Results
   • CMS application analysis
   • Alternative scenarios

5 Work in progress

6 Conclusions
Introduction
Motivation of this work

• Simulation in HEP involves the numerical solution of ODE systems in order to determine the trajectories described by charged particles in a magnetic field.

• As a particle moves through a detector, each volume crossing interrupts the underlying numerical solver.

• Traditional methods can invest considerable computational efforts to handle very frequent discontinuities accurately (detection of intersection points).
Motivation of this work

- **Quantized State System** methods (QSS, Kofman 2001 [7]) is a modern family of numerical integration methods exhibiting attractive features for this type of HEP simulation scenarios.

- The goals pursued in this work are:
  - To develop a proof-of-concept implementation of QSS within the Geant4 simulation toolkit.
  - To address its suitability as an alternative production integrator, and
  - To characterize its performance in a realistic HEP application.
Background information
• **Geant4** [1] is the most widely used simulation toolkit in contemporary HEP experiments.

• Provides classical numerical methods based on **time discretization** [3] (variations of the Runge-Kutta family of numerical solvers [4]).

• Uses custom iterative algorithms to approximate the event times of each spatial discontinuity (which mostly occur after a physics interaction).

• When these events are very frequent, they can dominate the CPU time dedicated to the integration method, and reduce considerably its performance.
Transportation of a charged particle $q$ along a step of length $h$ proposed by a physics process:

$\Rightarrow$ a total of 11 RHS evaluations involved for the 4th order Runge-Kutta.
Quantized State System methods

- QSS methods are based on **state variable quantization**.
- As opposed to traditional solvers which discretize **time** (e.g., Runge-Kutta family) QSS methods discretize the system’s **state variables**.
- **Continuous state variables** are thus **quantized** and approximated by their corresponding **quantized variables**.
- The relation between both is given by a **quantization function** which is in charge of the **error control** and **accuracy control**.

<table>
<thead>
<tr>
<th>ODE system</th>
<th>Quantized system</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \dot{x}(t) = f(x(t)) )</td>
<td>( \dot{x}(t) = f(q(t)) )</td>
</tr>
</tbody>
</table>
QSS1: first order quantization function

\[ q_i(t) = \begin{cases} 
  x_i(t) & \text{if } \left| q_i(t^-) - x_i(t) \right| \geq \Delta Q_i \\
  q_i(t^-) & \text{otherwise}
\end{cases} \]

- \( \Delta Q_i \) is the quantum.
  - Maximum deviation allowed between \( x_i \) and \( q_i \) (error control).
  - Derived from the precision demanded by the user.
- Higher order methods (QSS\( n \)) follow essentially the same principle.
  - From the definition above, in QSS1 \( q(t) \) follows piecewise constant trajectories.
  - In QSS\( n \), \( q(t) \) is composed of piecewise \((n - 1)\)-th order polynomials.
QSS features

• QSS features attractive for HEP problems
  ▶ Asynchronicity
  Decoupled, independent computation of changes in states variables.
  ▶ Lightweight discontinuity handling
  Boundary crossings detected by finding roots of polynomial equations.
  ▶ Dense trajectory output

• Selected speedups reported for QSS vs. time-slicing methods modeling processes in various domains [6][2]

<table>
<thead>
<tr>
<th>Model</th>
<th>Features</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>advection-reaction</td>
<td>$10^4$ state variables</td>
<td>30x</td>
</tr>
<tr>
<td>spiking neurons</td>
<td>$10^3$ state variables</td>
<td>35x</td>
</tr>
<tr>
<td>logic inverters chain</td>
<td>4000 neurons, 80 connections per neuron</td>
<td>100x</td>
</tr>
<tr>
<td>cellular division</td>
<td>100 cells, 600 state variables</td>
<td>100x to 1000x</td>
</tr>
</tbody>
</table>
The QSS Solver [5] is an open-source standalone simulation tool. Provides C implementations for several QSS methods. Provides also implementations of some traditional algorithms (e.g., Dormand-Prince method). Our GQLink interface partially relies on the QSS Solver’s simulation engine.
Preliminary comparison between Geant4 and QSS Solver

- Circular 2D particle motion, uniform magnetic field, crossing equidistant parallel planes.
  - Known exact analytic solution facilitates error analysis.
  - Physics processes turned off.

- With 200 plane crossings and a track length of 100 m, QSS Solver is 6x faster than Geant4[9][8].
Geant4 to QSS Link (GQLink): an implementation of QSS within Geant4
GQLink: QSS within Geant4

- GQLink is a proof-of-concept implementation of QSS in Geant4.
  - Geant4 version 10.03.p01 (released February 24, 2017).
  - QSS Solver engine from version 3.0 (as of March 2016).
- Provides three new shared libraries to Geant4:
  - libqss: QSS core functionality.
  - libgqlink: interface API between Geant4 and QSS.
  - libmodel: model definition and structure (i.e., Lorentz equations).
- QSS methods have complete control over the propagation for each Geant4 transportation step.
  - QSS manages accuracy in its own terms (through the control of the quantum $\Delta Q$).
GQLink: not another Geant4 stepper.

An abstract, clean, single entry point interface to the QSS Solver family of numerical integration methods.
Detection of boundary crossings

- Boundary crossings are detected through Geant4’s geometry library.
- Follows same call pattern as in standard Geant4 simulations:
  - **LocateGlobalPointWithinVolume**
    - Notifies the geometry navigator that the particle has moved to a new position inside the current volume.
  - **IntersectChord**
    - Computes an initial estimation of an intersection by means of a linear segment between the endpoints of the step.
  - **EstimateIntersectionPoint**
    - Refines the initial estimation mentioned above through an iterative procedure.
Detection of boundary crossings

Cheaper particle transport until the crossing point using QSS polynomial dense output instead of iterative procedures:

- QSS dense output not fully exploited yet for boundary crossing detection \( \Rightarrow \) main goal driving our current work.
Results
CMS application analysis

- GQLink validation was performed against a standalone Geant4 application featuring:
  - Full CMS (RunI) detector geometry.
  - Volume base magnetic field excerpted from CMSSW.
  - Particle gun shooting $\pi^-$ particles (10 GeV, $10^4$ events).
  - Pythia $pp \rightarrow H \rightarrow ZZ$ ($Z$ to all channels) ($\sqrt{s} = 14$ TeV, 50 events).

- Step count distribution for $\pi^-$ (left) and secondary electrons (right) for $10^4$ single $\pi^-$ events, showing equivalency of GQLink simulations:
CMS application: performance comparison

- **Single $\pi^-$ events**
  - GQLink $\sim$17% slower.

- **Pythia $H \rightarrow ZZ$ events**
  - GQLink $\sim$22% slower (5.86 hours vs. 4.8 hours).
  - Geant4 stepper: G4ClassicalRK4 (accuracy set to $\epsilon = 10^{-5}$).
Alternative scenario: helix and parallel planes

- Different scenario: helix trajectory crossing parallel equidistant planes & more frequent boundary crossings.
- Physics processes turned off.
- Using G4ClassicalRK4 stepper (accuracy set to $\epsilon = 10^{-5}$).

- GQLink outperforms Geant4 when using $\geq 300$ planes ($\sim 35\%$ faster for 700 planes).
Work in progress
Work in progress

- Exploitation of QSS capabilities for efficient geometry crossing detection:
  - Conversion of Geant4 solids into faceted polyhedrons.
  - Finding intersection points by (analytically) solving a polynomial equation given by the equation of a plane and the QSS polynomials approximating the trajectory.
  - Candidate plane given by the face crossed by the linear segment joining the endpoints of a step.
Conclusions
Conclusions

• We developed **GQLink**, a prototype for QSS methods within Geant4.

• Validation: number of steps and tracks produced are statistically consistent with Geant4’s for both toy examples and realistic HEP applications.

• Performance:
  ▶ We found that GQLink can outperform Geant4 in certain simplified scenarios involving tracking only.
  ▶ Preliminary tests revealed GQLink is currently $\sim 17\%$ slower than standard Geant4 in a full CMS realistic scenario (using single $\pi^-$ events).

• From an abstract viewpoint, GQLink also opens new possibilities to interface Geant4 with any external stepper.
Thank you!

Questions?
J. Allison et. al., the Geant4 Collaboration.  
**Recent developments in geant4.**  

F. Bergero, J. Fernndez, E. Kofman, and M. Portapila.  
**Quantized State Simulation of Advection–Diffusion–Reaction Equations.**  

F. E. Cellier and E. Kofman.  
**Continuous System Simulation.**  
B. Cockburn and C.-W. Shu.


J. Fernández and E. Kofman.

A Stand–Alone Quantized State System Solver. Part I.


Quantized State Simulation of Spiking Neural Networks.

E. Kofman and S. Junco.


N. Ponieman.

Aplicación de Métodos de Integración por Cuantificación al Simulador de Partículas Geant4.


Application of State Quantization-Based Methods in HEP Particle Transport Simulation.