GQLink

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

Lucio Santi

Universidad de Buenos Aires

September 8, 2016

Table of contents

Introduction

Motivation of this work

- Simulation in HEP involves numerical solutions to 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 invest considerable computational efforts to handle these discontinuities.

Motivation of this work

- Quantized state system (QSS) methods[2] are a family of novel numerical integrators with attractive features for these type of problems.
- The goals pursued in this work are:
  - To develop a proof-of-concept implementation of QSS within Geant4,
  - To address its suitability as an alternative integrator, and
  - To evaluate its performance in a realistic HEP application.

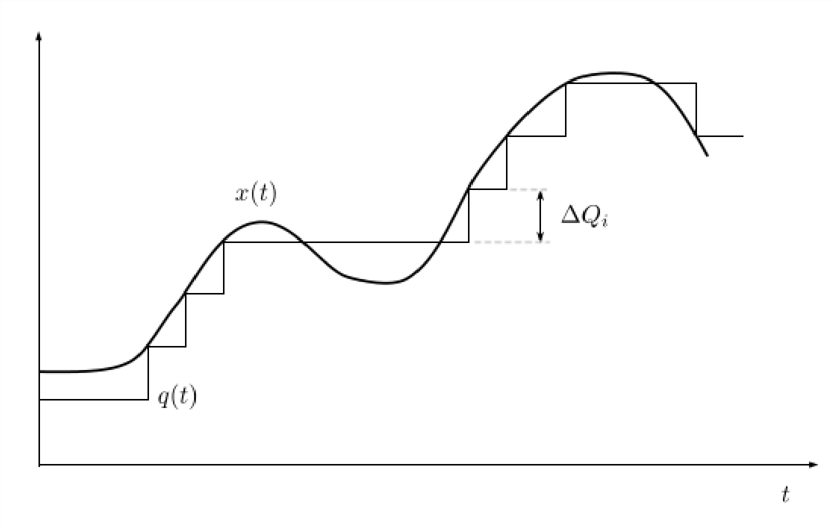
Quantized State System (QSS) methods

Quantized state systems methods

- QSS methods are based on **state quantization**.
- As opposed to traditional solvers (e.g., Runge-Kutta family), which discretize time, QSS discretizes the system's state.
- **State variables** are thus approximated by **quantized variables**.
- The relation between both is given by a **quantization function**.

QSS: example

- Asynchronous "steps" of $q(t)$ dictated by the quantization of the state variable $x(t)$.



Definition

- Consider the initial-value problem

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), t) \\ \mathbf{x}(t_0) = \mathbf{x}_0 \end{cases}$$

- QSS simulates the following approximate system,

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{q}(t), t) \\ \mathbf{q}(t_0) = \mathbf{x}_0 \end{cases}$$

where $\mathbf{x}(t)$ and $\mathbf{q}(t)$ are related by a (hysteretic) quantization function.

QSS1: quantization function

- In first-order QSS (QSS1), the quantization function is defined as follows:

$$q_i(t) = \begin{cases} x_i(t), & \text{if } |q_i(t^-) - x_i(t)| \geq \Delta Q_i \\ q_i(t^-), & \text{otherwise.} \end{cases}$$

where ΔQ_i is called the *quantum* –the maximum deviation allowed between x_i and q_i.

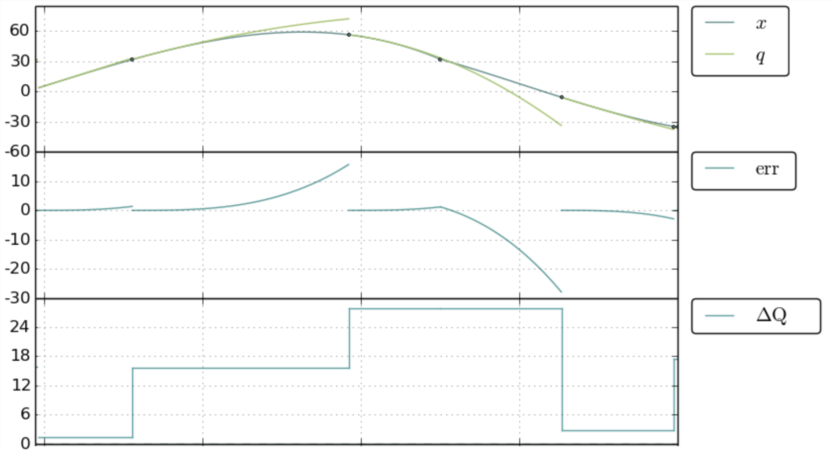
- Derived from the relative precision demanded by the user.

QSS features

- **Asynchronicity**
  - Each state variable is simulated at its own pace.
  - The time at which a given state variable triggers its next integration step is independent for separate states.
  - An integration step over state variable $x_i(t)$ only demands evaluation of those equations depending on $x_i(t)$.
- **Lightweight discontinuity handling**
  - Discontinuities in QSS models are detected by zero-crossing functions.
  - In turn, this is achieved by finding polynomial roots.

Higher order QSS methods

- Higher order QSSn methods follow essentially the same principle as QSS1.
- In QSS2, $q(t)$ is a piecewise linear function, whereas in QSS3 $q(t)$ is piecewise parabolic:



Integration steps

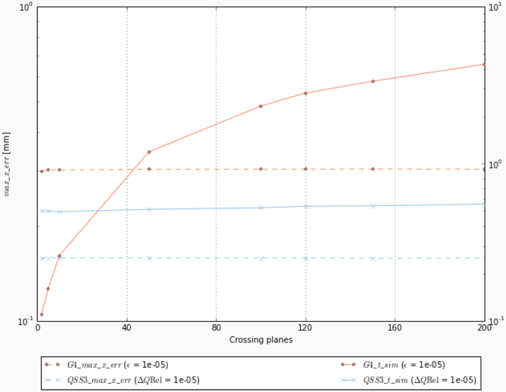
- An integration step consists in determining the upcoming quantization time t and updating $q(t)$ according to the quantization function.
  - Quantization times are found by computing polynomial roots.
- In turn, every derivative affected by this change should be recomputed.
- Any other derivative not depending on $q(t)$ can be safely skipped.

Standalone tool: QSS Solver

- The QSS standalone solver[1] is an open-source tool to simulate QSS models.
- Provides not only C implementations of every QSS family member but also custom versions of other traditional algorithms (e.g., Dormand-Prince method).
- GQLink's core is based on the engine of this tool.

Preliminary comparison between Geant4 and QSS Solver

- Circular motion in uniform magnetic field.
- Equidistant planes along the trajectory of the particle.



- With 200 plane crossings and a track length of 100 m, QSS Solver was 8x faster than Geant4[3].

GQLink: an implementation of QSS3 within Geant4

GQLink: QSS within Geant4

- GQLink is a proof-of-concept implementation of QSS in Geant4.
- It is based on:
  - Version `10.02.p01` of Geant4 (released February 26, 2016).
  - QSS Solver engine source code as of March 2016.

Components

- QSS Solver code was integrated into Geant4's building process.
- Three new shared libraries:
  - `libqss`: QSS core functionality.
  - `libgqlink`: interface API between Geant4 and QSS.
  - `libmodel`: model definition and structure (i.e., Lorentz equations).

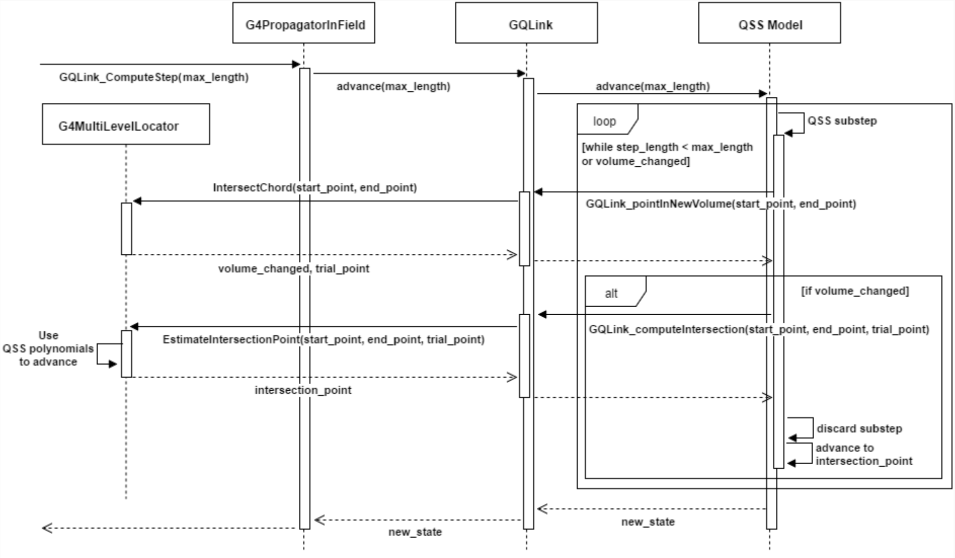
Step control

- GQLink is not a new Geant4 stepper, but an abstract interface to the QSS Solver library.
- QSS methods have complete control over each Geant4 step.
  - Usual accuracy parameters (e.g., **deltaOneStep**) do not affect GQLink simulations.
  - QSS manages accuracy in its own terms (through the control of $\Delta \mathrm{Q}$).
- `G4Transportation::AlongStepGPIL` calls a new method, `GQLink_ComputeStep`, that propagates the particle in the field using QSS.

Detection of boundary crossings

- Boundary crossings are detected through Geant4's geometry library.
- Follows same call pattern as in standard Geant4 simulations:
  - `LocateGlobalPointWithinVolume`
  - `IntersectChord`
  - `EstimateIntersectionPoint`
- **Improvement:** `AccurateAdvance` no longer used inside `EstimateIntersectionPoint`. QSS polynomials offer a cheaper alternative.

Stepping in GQLink: sequence diagram



CMS application analysis

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

Step count distribution (10^4 single π^- events)

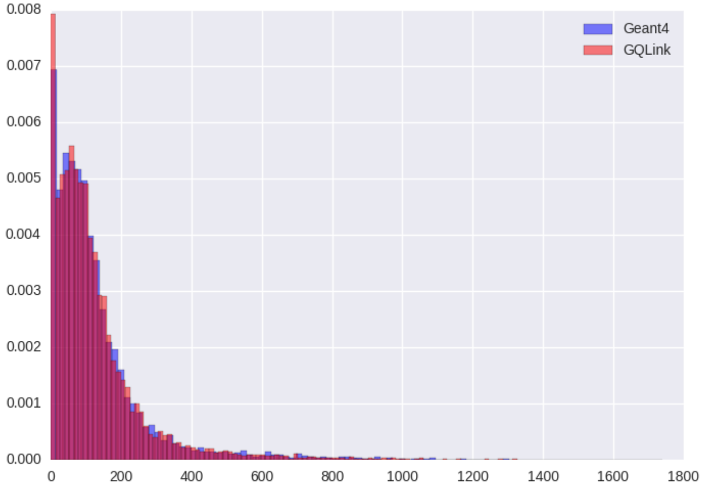


Figure 1: pion steps

Step count distribution (10^4 π^- events)

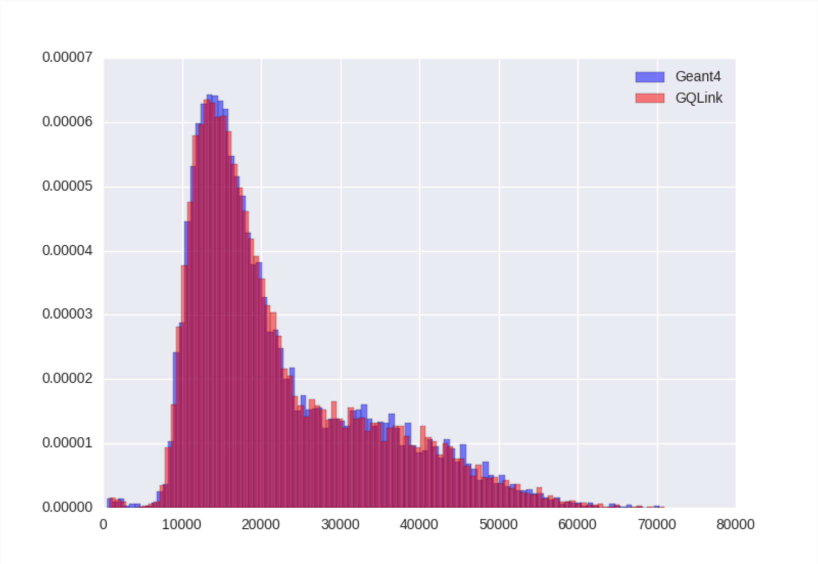
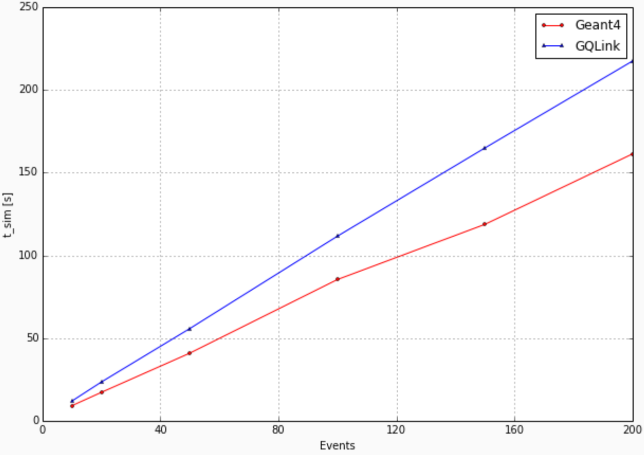


Figure 2: electron steps

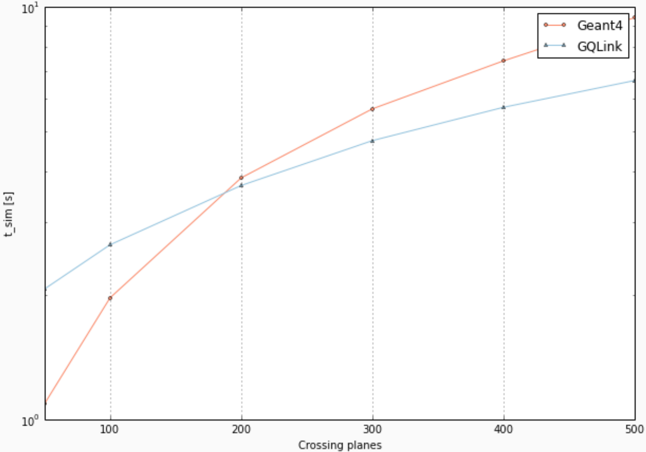
Performance comparison



- In this case GQLink simulated the CMS application ~34% slower than Geant4.

Alternative scenarios (helix and parallel planes)

- We studied GQLink's performance on a different scenario with more frequent boundary crossings (a helix trajectory crossing parallel equidistant planes).
- Also, we deliberately ensured there were no stepwise abrupt changes in the direction/velocity of the particle.



Conclusions and future work

Conclusions

- We developed and analyzed a prototype of QSS methods embedded in the Geant4 framework.
- We verified this numerical integrator produces meaningful results in the context of a realistic HEP application such as the CMS experiment.
- Preliminary performance tests revealed this new approach is currently about 34% slower than standard Geant4 for this application.

Conclusions

- GQLink currently uses QSS3, a third order method, whereas Geant4 uses combinations of fourth and fifth order Runge-Kutta methods.
- The observed simulation time can be partially explained by this fact, since lower order methods typically require more computational steps to achieve the same accuracy.
- QSS4 is still experimental, but GQLink will transprently support it once it becomes available.

Conclusions

- On the other hand, we found that GQLink can outperform Geant4 on certain scenarios.
  - Few stepwise direction/velocity changes enable GQLink to skip computationally expensive procedures to set up the new values upon starting a step.
  - Very frequent geometry crossings also leverage the QSS polynomials used inside `EstimateIntersectionPoint`.
- We aim at performing with QSS similarly to Geant4 in the standard cases, and outperform Geant4 in those cases where QSS features can be leveraged (which is scenario–dependent).
- From an abstract viewpoint, GQLink opens new possibilities by connecting Geant4 with external steppers (not limited to the QSS family).

Future work

- Exploit fully the QSS capabilities for efficient geometry crossing detection.
- Improve the performance of QSS for the reinitialization of momentum variables forced from Geant4 upon starting a new step.

Acknowledgments

- Rodrigo Castro (DC - UBA)
- Nicolás Ponieman (DF - UBA)
- Joaquín Fernández, Federico Bergero and Ernesto Kofman (UNR)
- Soon Yung Jun, Krzysztof Genser and Daniel Elvira (FNAL)

Thank you!

Questions?

References I

J. Fernández and E. Kofman.
A Stand–Alone Quantized State System Solver. Part I.
In *Proc. of RPIC 2013*, Bariloche, Argentina, 2013.

E. Kofman and S. Junco.
Quantized State Systems. A DEVS Approach for Continuous System Simulation.
Transactions of SCS, 18(3):123–132, 2001.

N. Ponieman.
Aplicación de métodos de integración por cuantificación al simulador de partículas geant4.
Master's thesis, Facultad de Ciencias Exactas y Naturales. Universidad de Buenos Aires., 2015.