GQLink: an implementation of Quantized State Systems (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] Department of Computer Science, FCEN, University of Buenos Aires, Argentina
[2] ICC-CONICET, Argentina
[3] CIFASIS-CONICET, Argentina
[4] Fermi National Accelerator Laboratory †, PO Box 500, Batavia, IL 60510, USA

E-mail: {lsanti,rcastro}@dc.uba.ar

Abstract. Simulations in high energy physics (HEP) often require the numerical solution of ordinary differential equations (ODE) to determine the trajectories of charged particles in a magnetic field when particles move throughout detector volumes. Each crossing of a volume interrupts the underlying numerical method that solves the equations of motion, triggering iterative algorithms to estimate the intersection point within a given accuracy. The computational cost of this procedure can grow significantly depending on the application at hand. Quantized State System (QSS) is a recent family of discrete-event driven numerical methods exhibiting attractive features for this type of problems, such as native dense output (sequences of polynomial segments updated only by accuracy-driven events) and lightweight detection and handling of volume crossings. In this work we present GQLink, a proof-of-concept integration of QSS with the Geant4 simulation toolkit which stands as an interface for co-simulation that orchestrates robustly and transparently the interaction between the QSS simulation engine and aspects such as geometry definition and physics processes controlled by Geant4. We validate the accuracy and study the performance of the method in simple geometries (subject to intense volume crossing activity) and then in a realistic HEP application using a full CMS detector configuration.

1. Introduction

High energy physics (HEP) particle simulations deal intensively with the tracking of particles affected by physics processes in complex detector geometries (adjacent 3D volumes of different shapes and materials). The accuracy and efficiency of the tracking algorithms are of central interest to optimize the use of available computing resources. Every time a charged particle crosses a volume boundary, the numerical method that solves the underlying ordinary differential equations (ODEs) [1] in a magnetic field needs to be interrupted. As this situation can happen many times during the lifetime of a particle in a typical HEP setup, the underlying continuous models describing particle trajectories must cope with frequent discontinuities. These spatial discontinuities will seldom coincide with the start of a new step. Thus, custom iterative algorithms are required to approximate the timestamp of the event time of each discontinuity in an accurate way. When these events are very frequent, they can dominate the processing time of the numerical method and considerably degrade its performance.

Contemporary HEP experiments are widely based on the Geant4 [2] simulation toolkit. It uses classical numerical methods that rely on time discretization [3], in particular variations of the Runge-Kutta family of solvers [4]. The Quantized State System (QSS) methods [3, 5], on the other hand, discretize the state variables instead of discretizing the time and solve ODEs using discrete–event approximations of continuous models. A feature of QSS particularly relevant in the context of HEP simulations is that these methods handle discontinuities very efficiently [6]. In QSS, the state variables follow piecewise polynomial trajectories. The detection of a volume crossing is modeled by solving a zero-crossing function, which can be done efficiently using the aforementioned polynomials (at least for some categories of the surface boundaries). Since each new QSS step is, by definition, a discontinuity in the quantized variable –strictly speaking, a discrete event– this task is naturally supported by the numerical method.

In previous works [7, 8] we verified the potential of QSS to offer speedups in HEP simulations, in particular in scenarios with heavy volume crossing activity. Yet, our studies were limited to compare Geant4 and QSS Solver [9], a standalone simulation toolkit that provides optimized implementations of QSS methods. A salient limitation of this approach is that physics processes were turned off for comparability purposes, restricting the comparisons to simple setups conceived as baselines. In this work we enable the capability to evaluate performance in realistic HEP applications by means of a proof-of-concept integration of QSS within Geant4.

This paper is organized as follows: Section 2 briefly introduces the QSS theory and the Geant4 transportation chain and then summarizes our preliminary results on QSS suitability for HEP simulations. Section 3 provides a high-level description of GQLink. Next, Section 4 discusses the verification of the method and presents a performance comparison between pure Geant4 and GQLink in two different setups. Finally, Section 5 contains a summary, conclusions and plans for the future.

2. Background

2.1. The Quantized State System (QSS) numerical integration methods

QSS are numerical methods that solve systems of ordinary differential equations (ODEs) in the form of Eq. 1, where $\mathbf{x}(t)$ is the *state vector* and $\mathbf{u}(t)$ is the *input vector* representing independent variables for which no derivatives are present in the system.

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t)) \qquad (1) \qquad\qquad \dot{\mathbf{x}}(t) = f(\mathbf{q}(t), \mathbf{u}(t)) \qquad (2)$$

Traditional ODE solvers (like the Runge-Kutta family) make use of *time slicing*: given the current and past state and state derivative values, the solver estimates the next value of the state x_{k+1} one “time step” Δt into the future, i.e., at $t_{k+1} = t_k + \Delta t$. QSS solvers operate differently: they make use of *state space quantization*. Rather than discretizing the time axis, they discretize the state variable axis, evaluating the first time instant t_{k+1} in the future at which the state variable differs from its current value by one “quantum level” ΔQ, i.e., when $x_{k+1} = x_k \pm \Delta Q$. The system described by Eq. 1 is thus approximated by the quantized system shown in Eq. 2, where $\mathbf{q}(t)$ is the *quantized state vector* resulting from the quantization of the state variables $x_i(t)$. In the first-order QSS method (QSS1) each $q_i(t)$ follows a piecewise constant trajectory that is updated by a (hysteretic) *quantization function* when the difference between $q_i(t)$ and $x_i(t)$ reaches the *quantum* ΔQ_i, a value derived from the precision demanded by the user.

In QSS1, $\mathbf{q}(t)$ follow piecewise constant trajectories, which implies that $\mathbf{x}(t)$ follow piecewise linear trajectories. Higher-order QSS methods generalize this behavior: in QSSn, $\mathbf{x}(t)$ follow piecewise n-th degree polynomial trajectories and $\mathbf{q}(t)$ follow piecewise $(n-1)$-th degree polynomial trajectories [5].

In the context of HEP applications, QSS exhibit several attractive features, such as inherent asynchronicity (each state variable updates its value independently at self clocked time instants dictated by its own dynamics and the accuracy ΔQ), dense trajectory output (at any given time t_0, QSS approximates $\mathbf{x}(t_0)$ using the piecewise polynomial trajectories computed for the interval in which t_0 lies in) and efficient handling of frequent discontinuities (a discontinuity is modeled by a zero-crossing function which is in turn defined in terms of the QSS polynomials).

Regarding practical tools, *QSS Solver* [9] is an open-source standalone software that provides optimized C implementations of different QSS methods as well as other traditional algorithms (e.g., Dormand-Prince method). Equations are expressed in μ-Modelica [10], a subset of the more general Modelica language [11].

2.2. Particle transport in Geant4

Geant4 traces particle trajectories in a magnetic field by means of the Runge-Kutta family of numerical methods. Figure 1 illustrates how a charged particle q is transported along a step of length h proposed by a physics process. In this context, the step computation involves a total of 11 right-hand side ODE evaluations for the 4th-order Runge-Kutta.

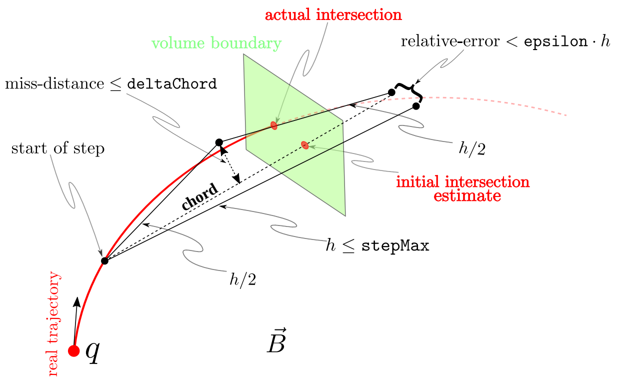


Figure 1: A charged particle trajectory in a magnetic field ($\vec{B}$) using the 4th-order Runge-Kutta method.

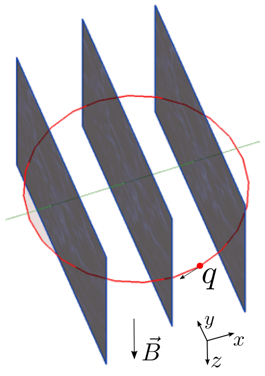
A comprehensive explanation of the stepping algorithm can be found in [7]. In particular, we stress that the intersection point detection algorithm can be rather expensive, as it needs to iterate back and forth until a candidate point satisfying the intersection accuracy constraints is found. This suggests that methods such as QSS, which naturally provide a lightweight handling of discontinuities, are good candidates for simulating HEP setups with frequent volume crossings. Indeed, the preliminary results summarized in the next section support this hypothesis.

2.3. Preliminary results

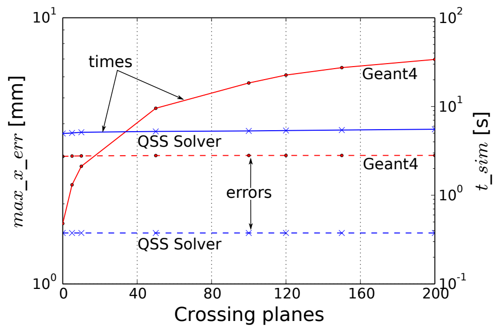
In [7, 8] we studied the performance of QSS Solver in the context of a simplified HEP setup consisting of a circular 2D motion of a charged particle under an uniform magnetic field. The particle crosses equidistant parallel planes as shown in Figure 2a. We found configurations with 200 planes and a track length of 100 m where QSS Solver is 6x faster than Geant4 (Figure 2b) with better accuracy, when the physics interactions are turned off.

3. Geant4 to QSS Link (GQLink): linking QSS libraries to Geant4

GQLink is a proof-of-concept mechanism to make QSS methods available within Geant4. It is currently based on the Geant4 toolkit version `10.03.p01` and the QSS Solver toolkit version 3.0. Designed as an abstract, clean, single entry point interface to the QSS Solver family of numerical integration methods, GQLink is virtually ready to support other integrators in the future. This is depicted in Figure 3a. It must be noted that GQLink was not conceived as yet another Geant4 *stepper*, although its main function is to replace the tasks typically performed by *steppers* in Geant4. Being integrated into the Geant4 building process, GQLink provides three new shared libraries: `libqss`, containing QSS core functionality, mostly derived from the QSS Solver engine, `libgqlink`, which offers the interface API between Geant4 and QSS, and `libmodel`, where the actual structure of the model to be simulated is defined (typically the Lorentz equations of motion).



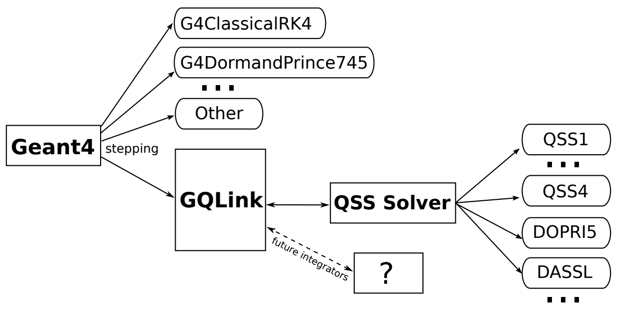
(a) Experimental setup



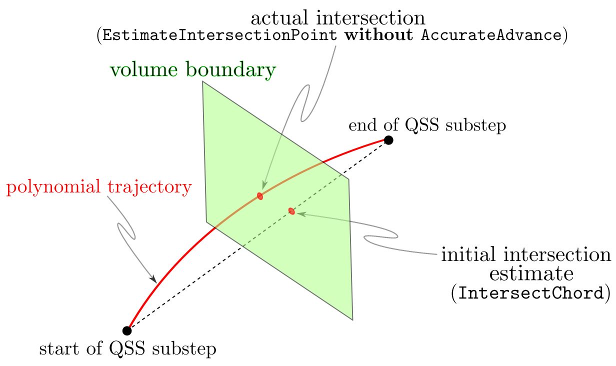
(b) Maximum error and simulation time vs. number of crossing planes

Figure 2: Preliminary comparison between Geant4 and QSS Solver

It is worth noting that currently boundary crossings are still detected through Geant4's geometry library. This is achieved by following the same call pattern as in standard Geant4 simulations: after each substep, `LocateGlobalPointWithinVolume` is called so as to notify the geometry navigator that the particle has moved to a new position inside the current volume. Then, `IntersectChord` computes an initial estimate of an intersection by means of a linear segment between the endpoints of the substep, which is later refined through an iterative procedure in `EstimateIntersectionPoint`. Still, this last function was improved by the QSS polynomial dense output information; we offer a cheaper particle transport by evaluating the polynomials instead of running the iterative, adaptive stepping algorithm implemented in `AccurateAdvance`. Figure 3b sketches how an intersection point is computed in GQLink.



(a) High-level architecture



(b) Computation of intersection points

Figure 3: GQLink implementation details

We finally stress that QSS dense output is not fully exploited yet for boundary crossing detection; this is in fact the main goal driving our current work.

4. Results and discussion

We start with a statistical validation of GQLink in the context of a CMS application in order to ensure that it produces equivalent results than Geant4. After this, we compare their computing performance in this same CMS setup and also in a simplified setup with a controlled, increasing number of boundary crossings. All simulations were run on a dedicated Intel Xeon E5-2620 v2 CPU (clocked at 2.10 GHz) server with 8 GB of RAM and Ubuntu 14.04.4 LTS x86_64 (`3.13.0-49-generic` kernel) OS. The compiler used was `gcc` 4.8.4 (Ubuntu 4.8.4-2ubuntu1~14.04.1).

GQLink validation was performed against a standalone Geant4 application featuring full CMS (RunI) detector geometry, volume base magnetic field excerpted from CMSSW and a particle gun shooting π^- particles (10 GeV, 10^4 events). In Figure 4, we show how the particle steps are distributed after 10^4 single π^- events, for pions (left) and secondary electrons (right). We observe that the number of steps produced are statistically consistent with those computed by Geant4. This behavior is in fact present in every other particle type as well as in the number of tracks.

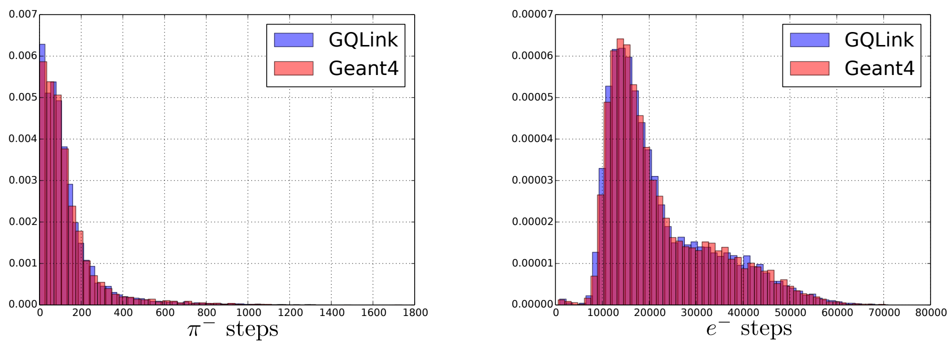
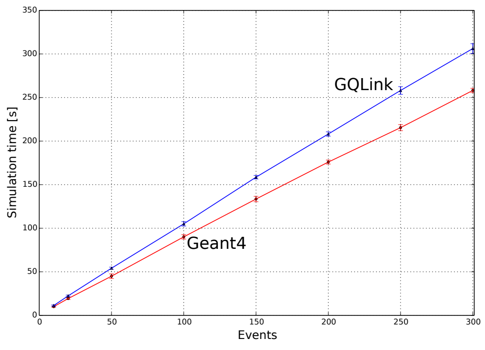


Figure 4: Number of steps for π^- (left) and secondary electrons (right) from simulation of 10^4 single π^- events

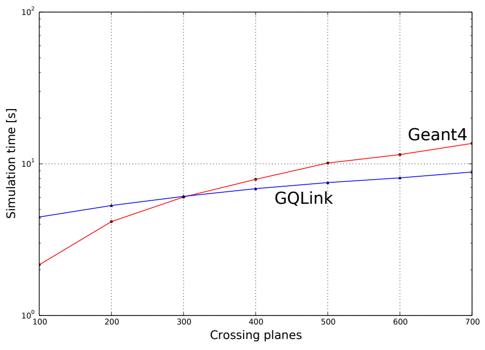
As for the computing performance, Figure 5a compares the simulation time of GQLink and Geant4 for an increasing number of events in the same CMS setup presented above. For each number of events, a total of 50 simulations were run. The plot shows their mean values along with the minimum and maximum values attained. In this case, GQLink performed about 17% slower than Geant4. We inferred that this is due to a combination of issues: a relatively small number of boundary crossings (about 8% of the total number of steps), a greater average number of substeps per integration step (up to 3x with respect to Geant4, with the set relative accuracy of 10^{-5}) and frequent reinitializations of QSS state variables after each Geant4 step (required to update the state derivatives after possible changes in the direction of the particle).

However, when the number of boundary crossings is controlled and can be forced to increase, GQLink's performance progressively improves and it eventually outperforms Geant4. This behavior is shown in Figure 5b, where we consider a simplfied experimental setup very similar to that of Figure 2 but with a single particle describing a helicoidal trajectory and with physics interactions turned off. We see that GQLink outperforms Geant4 when using at least 300 crossing planes, and can be up to $\sim$35% faster for 700 planes. This is a consequence of the improvement derived from the QSS polynomials implemented in `EstimateIntersectionPoint`, as mentioned in Section 3.

Finally, we also analyzed the correctness and performance of the method using 50 Pythia $pp \to H \to ZZ$ events (Z to all channels) ($\sqrt{s} = 14$ TeV) in the context of the CMS application. In this case, we found that GQLink successfully simulated the 50 events in 5.86 hours, whereas Geant4 took 4.8 hours. This is an increase of approximately 22%. Note, however, that the same limitations discussed above apply in this scenario as well –namely, very few boundary crossings and greater number of QSS substeps per integration step.



(a) CMS application (single π^- events)



(b) Helix and parallel planes; physics interactions turned off

Figure 5: Performance comparison between GQLink and Geant4

5. Conclusions

We developed GQLink, a prototype for QSS methods (as implemented by the standalone simulation toolkit QSS Solver) within Geant4. It is conceived as an abstract interface that allows connecting Geant4 to arbitrary external integrators, not limited to those provided by QSS Solver. In order to validate its correctness and study its performance, we used a realistic HEP application using a full CMS detector geometry, with magnetic field and both particle gun events, shooting pions, as well as Pythia $pp \to H \to ZZ$ events (Z to all channels) ($\sqrt{s} = 14$ TeV). We found that the number of steps and tracks produced are statistically consistent with those of standard Geant4, suggesting the equivalence of the simulations. As for the performance, since these scenarios have a very light boundary crossing activity, we found that GQLink is currently about 17% slower. On the other hand, we studied a simplified scenario, with physics interactions turned off, where a single particle describing helical trajectories moves throughout an increasing, controlled number of crossing planes. In this setup, GQLink outperformed Geant4, with a 35% speedup for 700 crossing planes. This is due to an improvement in the Geant4 geometry library, where we resorted to the knowledge stored in each QSS polynomial trajectory in order to avoid some iterative stepping procedures used when locating intersection points. We are currently working on fully exploiting these QSS capabilities for efficient volume crossing detections. We aim at handling the detections completely on the QSS side, which is expected to provide considerable performance boosts.

Acknowledgments

† Operated by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the United States Department of Energy.

References

[1] Hairer E, Nørsett S and Wanner G 1987 Solving ordinary differential equations i - nonstiff problems
[2] Allison J 2016 *Nuclear Instruments and Methods A* **835** 186–225
[3] Cellier F E and Kofman E 2006 *Continuous System Simulation* (Springer-Verlag New York, Inc.)
[4] Cockburn B and Shu C W 1998 *Journal of Computational Physics* **141** 199–224
[5] Kofman E and Junco S 2001 *Transactions of SCS* **18** 123–132
[6] Kofman E 2004 *SIAM Journal on Scientific Computing* **25** 1771–1797
[7] Santi L, Ponieman N, Jun S Y, Genser K, Elvira D and Castro R 2016 *Journal of Physics: Conference Series*
[8] Ponieman N 2015 *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.
[9] Fernández J and Kofman E 2013 *Proc. of RPIC 2013* (Bariloche, Argentina)
[10] Bergero F, Floros X, Fernández J, Kofman E and Cellier F E 2012 *9th Int. Modelica Conference, Germany*
[11] Fritzson P 2004 *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1* (Wiley-Interscience, New York)