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

IOP Conf. Series: Journal of Physics: Conf. Series **898** (2017) 042049 doi:10.1088/1742-6596/898/4/042049

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

Lucio Santi[1,2], Nicolás Ponieman[1], Soon Yung Jun[3], Krzysztof Genser[3], Daniel Elvira[3], Rodrigo Castro[1,2]

[1] Department of Computer Science, FCEN, University of Buenos Aires, Argentina
[2] ICC-CONICET, Argentina
[3] Fermi National Accelerator Laboratory †, PO Box 500, Batavia, IL 60510, USA

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

Abstract. Simulation of particle-matter interactions in complex geometries is one of the main tasks in high energy physics (HEP) research. An essential aspect of it is an accurate and efficient particle transportation in a non-uniform magnetic field, which includes the handling of volume crossings within a predefined 3D geometry. Quantized State Systems (QSS) is a family of numerical methods that provides attractive features for particle transportation processes, such as dense output (sequences of polynomial segments changing only according to accuracy-driven discrete events) and lightweight detection and handling of volume crossings (based on simple root-finding of polynomial functions). In this work we present a proof-of-concept performance comparison between a QSS-based standalone numerical solver and an application based on the Geant4 simulation toolkit, with its default Runge-Kutta based adaptive step method. In a case study with a charged particle circulating in a vacuum (with interactions with matter turned off), in a uniform magnetic field, and crossing up to 200 volume boundaries twice per turn, simulation results showed speedups of up to 6 times in favor of QSS while it being 10 times slower in the case with zero volume boundaries.

1. Introduction

A significant challenge in high energy physics (HEP) particle simulations is an accurate and efficient tracking of particles affected by physics processes in complex detector geometries consisting of a variety of materials and many adjacent 3D volumes of different shapes. Every time a particle crosses a volume boundary, the numerical method that solves the underlying ordinary differential equations (ODEs) [1] 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.

Geant4 [2] is the most widely used simulation toolkit in contemporary HEP experiments. It provides classical numerical methods based on time discretization [3] (in particular, variations of the Runge-Kutta family of numerical solvers [4]) where time advances by well-defined steps of either fixed or adaptive duration. Since spatial discontinuities will rarely coincide with the start of a new step, which mostly occur after a physics interaction, custom iterative algorithms are needed to approximate the actual time of the discontinuity in an accurate way. When these events are very frequent, they can dominate the CPU time of the numerical method and considerably reduce its performance.

A new family of numerical integration methods that addresses efficiently the above type of challenges was developed relatively recently. The Quantized State System (QSS) methods [3, 5]

discretize the state variables instead of discretizing the time and solve ODEs using discrete–event approximations of continuous models. QSS methods approximate the state variables through discrete quanta; new integration steps can only happen whenever a state variable deviates by a predefined amount from its expected solution. As each variable is simulated independently from each other, at its own pace, QSS are inherently asynchronous methods.

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.

This paper is organized as follows: Section 2 briefly discusses the Geant4 transportation chain and introduces the QSS theory. Section 3 describes the experimental setup under study. Next, Section 4 presents a performance comparison between Geant4 and a standalone QSS simulation engine. Finally, Section 5 contains a summary, conclusions and plans for the future.

2. Background information on particle transport in Geant4 and QSS methods

2.1. Particle transport in Geant4

Geant4 computes particle trajectories in a magnetic field by means of Runge-Kutta family of numerical methods. In particular, the fourth-order Runge-Kutta (RK4) is the most widely used one. Figure 1 illustrates how a charged particle q is transported along a step of length h proposed by a physics process. Suppose the step starts at point (1). Then, the following procedure is applied:

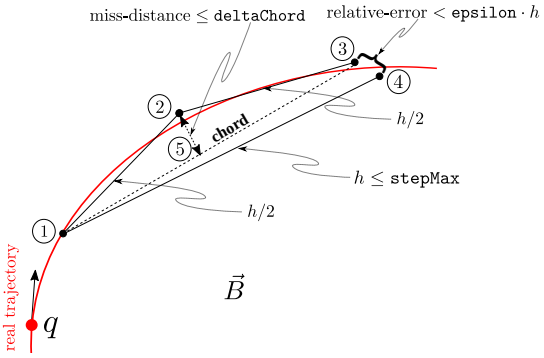


Figure 1: Particle transport sketch

(i) A step of length h is proposed. This value is limited by the physics processes affecting the particle, the detector geometry and a user-specified step limiting parameter, `stepMax`, which is enforced when set.

(ii) Then, two *half-steps* of length $h/2$ are computed: from (1) to (2) and from (2) to (3), using RK4.

(iii) Next, the closest distance between the **chord** (1)-(3) and the mid-point (2) of the approximate trajectory, termed *miss-distance*, is compared against `deltaChord` (another user-settable accuracy parameter). The miss-distance is given by segment (2)-(5).

(iv) If miss-distance > `deltaChord`, the step length h is reduced and the process starts over from (i).

(v) Otherwise, Geant4 determines whether the resulting numerical error is acceptable. This error is given by the combination of the errors in the distance and velocity between (3)

and (4) which we call *relative-error*. It is compared with a fraction of h given by `epsilon`, which in turn is derived from three other user-settable accuracy parameters: `epsilonMin`, `epsilonMax` and `deltaOneStep`.

a. If relative-error $<$ `epsilon` $\cdot\, h$, the step is accepted and the particle is moved to point (3).
b. Otherwise, h is reduced iteratively so as to improve the precision until the above condition is satisfied.

(vi) Finally, if segment (1)-(4) happened to cross a volume boundary, Geant4 computes the intersection point on the boundary using a custom iterative algorithm based on RK4.

As it can be seen, simulation performance strongly depends on the computing efforts needed by the numerical integration methods. In particular, the aforementioned 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.

2.2. 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:

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t)) \tag{1}$$

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. They evaluate 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 following quantized system:

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

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 related to $x_i(t)$ by the following *hysteretic quantization function*:

$$q_i(t) = \begin{cases} x_i(t) & \text{if } |q_i(t^-) - x_i(t)| \geq \Delta Q_i \\ q_i(t^-) & \text{otherwise} \end{cases} \tag{3}$$

where ΔQ_i is the *quantum*, i.e. the maximum deviation allowed between $q_i(t)$ and $x_i(t)$, and $q_i(t^-)$ is the left-handed limit of $q_i(t)$. Figure 2 illustrates the relation between $x_i(t)$ and $q_i(t)$. The hysteretic effect introduced by the quantization function $\mathbf{q}(t)$ prevents the occurrence of an infinite number of state changes within a finite time interval [5].

From Eq. 3, it can be seen that $q_i(t)$ changes its value whenever its difference with $x_i(t)$ exceeds the quantum ΔQ_i. This is called an *integration step*. Between these steps, in QSS1, quantized states $\mathbf{q}(t)$ follow piecewise constant trajectories, as shown in Figure 2. Since the derivatives $\dot{\mathbf{x}}(t)$ are expressed in terms of $\mathbf{q}(t)$, they are also piecewise constant. In consequence, state variables $\mathbf{x}(t)$ follow piecewise linear trajectories. Higher-order QSS methods generalize

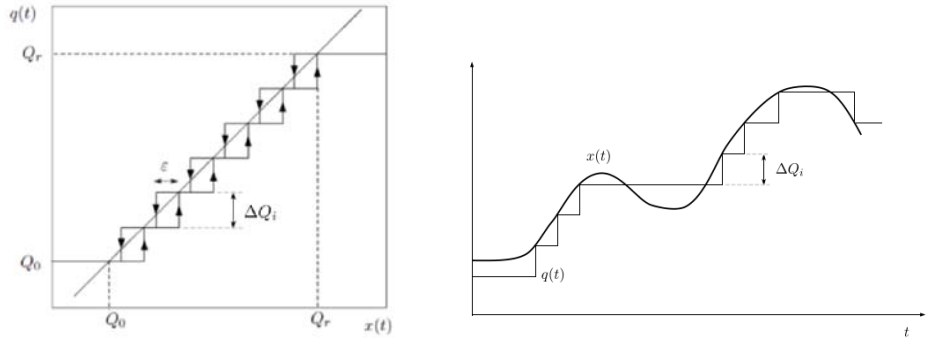


Figure 2: State quantization with an hysteretic window of height ΔQ in QSS1 (left). State quantization trajectory $q(t)$ in QSS1 for a given state variable trajectory $x(t)$ (right).

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.

A QSS method can simulate any ODE system in the form of Eq. 1 offering, among others, the following properties [5]:

- It is inherently asynchronous: each state variable updates its value independently at self clocked time instants dictated by its own dynamics and the accuracy ΔQ (cf. time slicing methods where *all* state variables are scanned synchronously at each Δt). This can offer significant performance advantages e.g. when simulating sparse systems.
- It provides dense 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.
- It is particularly efficient at simulating systems with very frequent discontinuities. A discontinuity is modeled by a zero-crossing function which is in turn defined in terms of the QSS polynomials. Thus, *detecting* a discontinuity calls only for finding the roots of a polynomial. The latter is computationally inexpensive for at most third-order QSS methods as it does not require iterative approximations.
- QSS1 to QSS3 provide a global error bound (controlled by ΔQ), limiting globally the error of the numerical solution of an analytically stable, time-invariant system.

Regarding practical tools, *QSS Solver* [7] is a standalone software that provides optimized implementations of different QSS methods. Equations are expressed in μ-Modelica [8], a subset of the more general Modelica language [9].

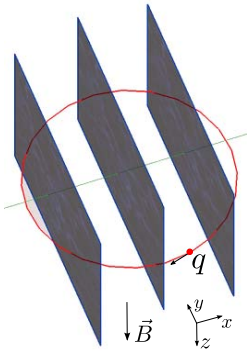
3. Case study: Circulating particle with frequent volume crossings

The setup studied in this work consists of a single electron under a uniform, static magnetic field along the $\hat{z}$ plane, i.e., $\vec{B} = (0, 0, B) = B\hat{z}$ and initial velocity $\vec{v} = v\hat{x}$. The particle then follows a circular trajectory in the $(\hat{x}, \hat{y})$ plane. Because of its simplicity, the case offers a closed form analytic solution which facilitates the error analysis. Figure 3b shows the underlying equations of motion and their analytic solution.

We designed a test scenario where discontinuities play a significant role in the form of frequent boundary crossings. The goal is to assess the potential benefits offered by QSS in terms of its lightweight event detection and handling features. Thus, we extended the setup by inserting equidistant parallel planes along the trajectory of the particle, as shown in Figure 3a.

4. Simulation experiments and discussion

We start with an individual analysis of the Geant4 application and QSS Solver in order to ensure that they produce simulations that are sufficiently close to the exact solution. After this, we compare them in two different scenarios: a planes-free scenario (i.e. with no boundary crossings) and then using an increasing number of crossing planes.



(a)

$$(1)\ \begin{cases} \dot{x}(t) = v_x(t) & \dot{v_x}(t) = \frac{q}{m\gamma} B\, v_y(t) \\ \dot{y}(t) = v_y(t) & \dot{v_y}(t) = -\frac{q}{m\gamma} B\, v_x(t) \end{cases}$$

$$(2)\ \begin{cases} x(t) = R\,\sin(\omega t) & v_x(t) = \ R\,\omega\,\cos(\omega t) \\ y(t) = -R + R\,\cos(\omega t) & v_y(t) = -R\,\omega\,\sin(\omega t) \\ R = \frac{m\,\gamma\,v}{q\,B} \quad \omega = \frac{q\,B}{m\,\gamma} \quad \gamma = \frac{1}{\sqrt{1-v^2/c^2}} \end{cases}$$

(b)

Figure 3: Setup sketch (a) and equations of motion for an electron in a constant magnetic field (b.1) with their analytic solution with cyclotron radius R and frequency ω (b.2)

4.1. Experimental setup

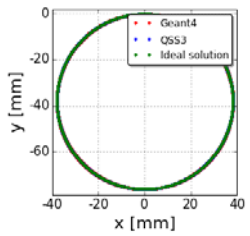
For each simulation a track length of 1 km was set. In the setup, we used $B = 1\,\text{Tesla}$ and $v = 0.999\,c$ (where $c = 299.792458\,\text{mm/ns}$), which yielded a circle radius R of about 38.085386 mm. We also used the following default values for accuracy control in Geant4: `epsilonMin` = `epsilonMax` = `epsilon` = 10^{-5}, `deltaOneStep` = 10^{-2} mm, `deltaChord` = 0.25 mm, `deltaIntersection` = 10^{-5} mm and `stepMax` = 20 mm. In the case of QSS Solver, we set $\Delta\text{QRel} = 10^{-5}$ to be comparable with `epsilon`.

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). We used Geant4 version 10.01.p01 and QSS Solver version 3.0.

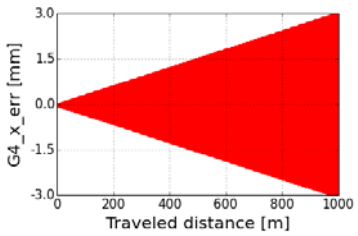
We finally stress that these were tracking-only simulations, i.e., all material related physics processes were intentionally turned off in Geant4 and not implemented in QSS Solver.

4.2. Simulation results

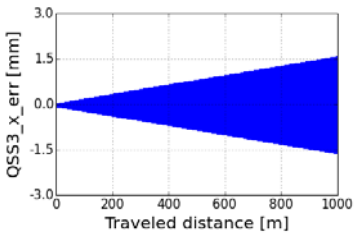
Figure 4a compares the x-y trajectories simulated by Geant4 and QSS Solver against the ideal analytic solution (first 10 revolutions). We can see that both trajectories are indistinguishable by a naked eye from the analytic solution. Figures 4b and 4c show the behavior of the error in $x(t)$ (G4_x_$err(t)$ and QSS3_x_$err(t)$, respectively) as a function of the distance traveled by the particle. The error, which is computed as the difference between the simulated and the analytic position of the particle in the x axis, undergoes an oscillating evolution, with amplitudes increasing monotonically (oscillations are hidden to a naked eye in Figures 4b and



(a) First 10 revolutions of Geant4 and QSS Solver trajectories compared to the ideal solution



(b) Evolution of G4_x_$err(t)$ during a track (dense oscillations with an increasing amplitude)



(c) Evolution of QSS3_x_$err(t)$ during a track (dense oscillations with an increasing amplitude)

Figure 4: Particle trajectories (a) and error evolution for independent simulations with Geant4 (b) and QSS Solver (c) (experimental setup with no planes)

4c, as these graphs show full error shapes producing a dense set of points). This increase is an expected consequence of having used numerical methods not specifically suited for marginally stable systems [3] like the one in Eq. 1 from Figure 3b. The error in $y(t)$ was verified to follow the same behavior.

Although the numerical error introduced by both methods has the same growth pattern, the scatter plot and histograms presented in Figure 5 show that QSS3 errors are smaller and have a smaller standard deviation.

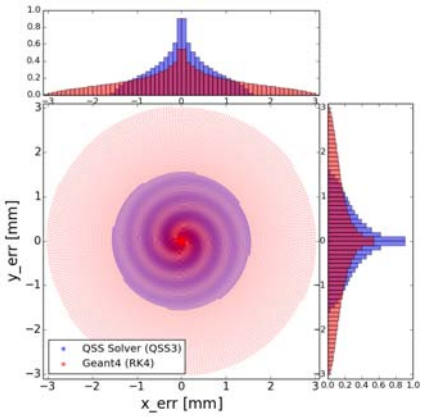
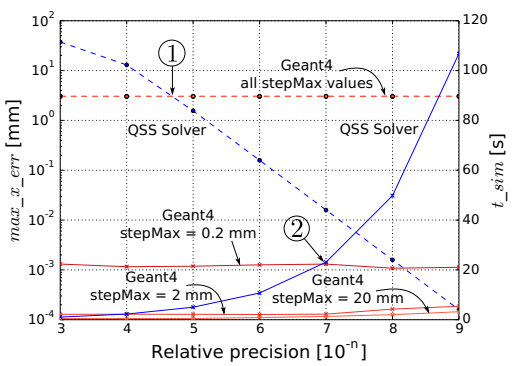


Figure 5: Distribution of errors in $x(t)$ and $y(t)$ for Geant4 and QSS Solver (experimental setup with no planes): histograms and scatter plots for $G4\_x\_err, G4\_y\_err, QSS3\_x\_err$ and $QSS3\_y\_err$

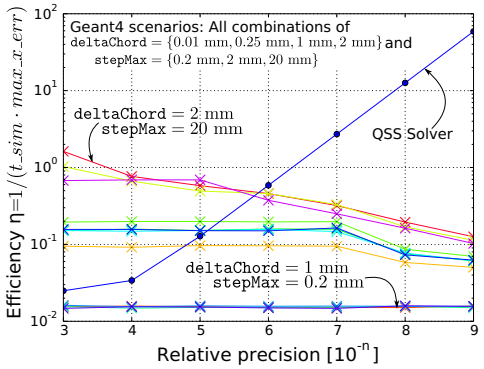
4.3. Performance comparison without crossing planes

Figure 6a shows a performance comparison in terms of a numerical error (defined here as the maximum error observed along the x coordinate of the particle's position, termed max_x_err) and simulation time (t_sim), for both methods. We show results obtained for three values of `stepMax` (0.2 mm, 2 mm and 20 mm).

Regarding the error behavior as related to the requested accuracy, we observe that for the set of chosen parameters (see Section 4.1) the Geant4 application does not seem to improve the error when the set relative precision is increased (i.e., when `epsilon` is decreased). A more exhaustive study including variations of other parameters can be found in [10]. In this matter,



(a) Maximum error (left y axis, dashed lines) and simulation time (right y axis, solid lines) vs. relative precision



(b) Combined method efficiency for different relative precisions: Geant4 scenarios (crosses) and QSS Solver (dots)

Figure 6: Performance comparison without geometry crossings

as seen from point (1) onwards, QSS3 error decreases approximately by an order of magnitude for each extra order of magnitude of the set relative precision (ΔQRel). Regarding the `stepMax` in case of Geant4, it does not seem to have a noticeable influence on the error (all three dashed lines appear to be on top of each other). As this user imposed parameter is used to force more steps, it may not improve the error when other parameters already impose stronger constraints. We also observe that the lower the value of `stepMax`, the higher the simulation time in Geant4 as more steps are needed to cover the same track length. Besides, when `epsilon` $\geq 10^{-7}$, t_sim increases more noticeably (cases for `stepMax` = 0.2 mm and 20 mm) as adaptive steps are taken to meet the error constraints (see Section 2.1). The solid line for QSS3 shows simulation times for different requested relative precision. In this case, we observe that QSS3 simulation time increases with the cubic root of the relative accuracy, which confirms a theoretical property of QSS3 [11]. When `stepMax` = 0.2 mm, between points (1) and (2), QSS3 outperforms Geant4 achieving both smaller error bounds and lower simulation times.

On the other hand, Figure 6b presents a more synthetic look at error and simulation times by means of a combined custom efficiency metric $\eta = 1/(t\_sim \cdot max\_x\_err)$. Crosses represent Geant4 simulations for different combinations of `stepMax` and its accuracy-related parameters (`deltaChord` and `epsilon`), whereas dots correspond to QSS3 simulations for different ΔQRel values. According to this metric, and for this case study, Geant4's efficiency tends to decrease as the relative precision increases, while the opposite happens for QSS3. In fact, when `epsilon` and ΔQRel are made equal to 10^{-9}, this custom efficiency measure for QSS3 is nearly three orders of magnitude higher than that of Geant4 (for all tested combinations of accuracy parameters).

4.4. Performance comparison with crossing planes

Figure 7 compares the performance of both methods for an increasing number of plane crossings along the trajectory of the particle. The most salient observation from this Figure is that QSS Solver's QSS3 simulation time scales better than Geant4's RK4 in the presence of an increasing number of plane crossings, achieving speedups of up to 6 times in the case of 200 planes, despite being initially significantly slower when the number of planes is small. The smooth growth in t_sim is consistent with the lightweight discontinuity handling property of QSS, as discussed in Section 2.2. Even though the behavior of errors (both in Geant4 and QSS3) does not seem to be affected by the number of plane crossings, QSS3 presented better error bounds in this scenario (using a fixed relative precision of 10^{-5}).

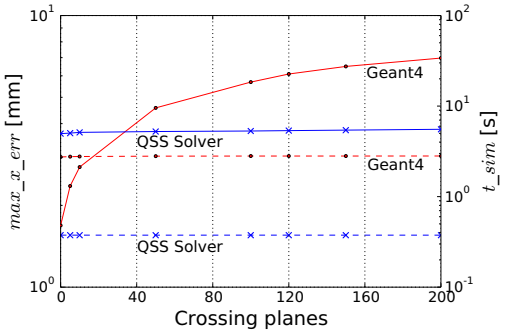


Figure 7: Maximum error (left y axis, dashed lines) and simulation time (right y axis, solid lines) vs. number of crossing planes (default experimental setup, relative accuracy = 10^{-5})

5. Conclusions and outlook

We studied performance of the Quantized State Systems (QSS) numerical solvers in the simulation of a charged particle motion in a uniform magnetic field. We relied on the

hypothesis that the asynchronous discrete–event nature of QSS methods can offer performance improvements in the numerical integration of particles' trajectories in scenarios with an increasing number of boundary crossings. This idea was rooted in the fact that classical discrete–time solvers (e.g. those based on the Runge-Kutta methods) need to apply iterative algorithms to detect discontinuities, while for QSS methods the handling of discrete events calls only for a lightweight root finding of piecewise polynomial segments (dense output). We chose a simple case with a well-known analytic solution (a particle moving in one plane in a constant magnetic field) and studied the simulation performance for discrete–time vs. discrete–event approaches. A Geant4 application was selected to test the discrete–time methods (using its default customized Runge-Kutta adaptive stepper). The QSS Solver simulator was selected to test the QSS3 discrete–event method.

Our results showed that QSS scaled significantly better than Runge-Kutta in situations with an increasing number of volume crossings and with an increasing requested accuracy. Also, QSS offered a straightforward and very predictable mechanism for accuracy vs. error control with an analytically calculable formula for the latter. As an example, in the scenario where the circulating particle crosses 200 planes twice per turn (with particle-matter interactions turned off in Geant4 and not implemented in QSS Solver), QSS Solver performed up to 6 times faster than Geant4 with an error up to 2 times smaller, despite being 10 times slower in the case with zero planes.

The results suggest that QSS methods have a potential to reduce simulation time in scenarios with heavy volume crossing activity, while offering a simple and very predictable accuracy control which is an inherent property of the methods. To verify this hypothesis we intend to fully integrate QSS methods within Geant4 to make them available along the current Geant4 steppers, to enable more direct comparisons within the same application and to enable studies of more realistic test cases with particle-matter interactions turned on and using more complex detector geometries. This is work in progress, showing so far that a smooth coupling of Geant4 with QSS is algorithmically feasible. From an efficiency perspective, the introduction of physics processes would call for reinitializations of QSS variables at every interaction or decay point, bringing potential time performance penalties with respect to the speedups reported in this work.

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* (Secaucus, NJ, USA: Springer-Verlag New York, Inc.) ISBN 0387261028
[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] Fernández J and Kofman E 2013 *Proc. of RPIC 2013* (Bariloche, Argentina)
[8] Bergero F, Floros X, Fernández J, Kofman E and Cellier F E 2012 *9th Int. Modelica Conference, Munich Germany*
[9] Fritzson P 2004 *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1* (Wiley-Interscience, New York)
[10] 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.
[11] Kofman E 2006 *Latin American Applied Research* **36** 101–108