

Alternative numerical methods to solve equations of motion for a charged particle in a magnetic field

Exploring the suitability of Discrete Event driven Quantized State Systems (QSS) methods

Nicolás Bruno Ponieman¹
Lucio Santi², Rodrigo Castro²
Soon Yung Jun³, Krzysztof Genser³

¹Department of Physics, University of Buenos Aires

²Computer Science Department, University of Buenos Aires

³FNAL



universidad de buenos aires - exactas
departamento de Física
Juan José Giambiagi



UBA
Universidad de Buenos Aires
Argentina virtus robor et studium



**DEPARTAMENTO
DE COMPUTACION**

Facultad de Ciencias Exactas y Naturales - UBA

- 1 Transportation Chain in Geant4
 - Introduction
 - Parameters Involved
 - Runge-Kutta
- 2 Quantized State System
 - Introduction to QSS
 - QSS Explained
 - QSS vs Runge-Kutta
- 3 Exploring Geant4 Parameters
 - Introduction
 - Varying epsilon
 - Varying stepMax
 - Varying deltaChord
- 4 Exploring QSS Parameters
- 5 Geometry crossings - Performance Comparison

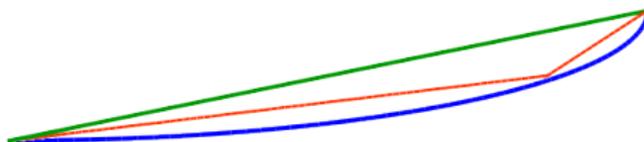
Outline

- 1 Transportation Chain in Geant4
 - Introduction
 - Parameters Involved
 - Runge-Kutta
- 2 Quantized State System
 - Introduction to QSS
 - QSS Explained
 - QSS vs Runge-Kutta
- 3 Exploring Geant4 Parameters
 - Introduction
 - Varying epsilon
 - Varying stepMax
 - Varying deltaChord
- 4 Exploring QSS Parameters
- 5 Geometry crossings - Performance Comparison

Transportation Chain in Geant4

Introduction

- To propagate a charged particle inside a magnetic field, we solve its equation of motion using numerical methods.
- Geant4 uses the numerical methods of Runge-Kutta class to solve the Ordinary Differential Equations (ODE).
- Each step may be divided in smaller steps.
- Geant4 breaks up the curved path into linear chord segments.
- Chord segments are determined so that they closely approximate the curved path (if the current error is large, the path uses more segments) in order to control the accuracy of volume intersection.

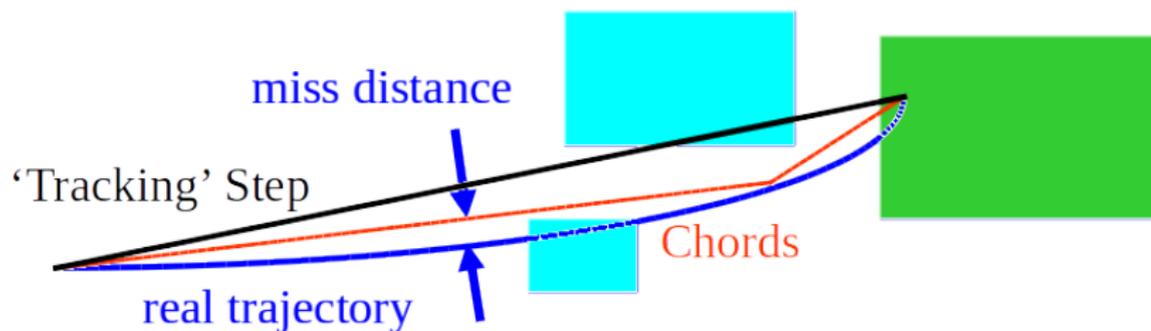


Images to illustrate the parameters effect were taken from: "Geant 4, Detector Description: EM Fields", J.Apostolakis

Transportation Chain in Geant4

Visual Analysis

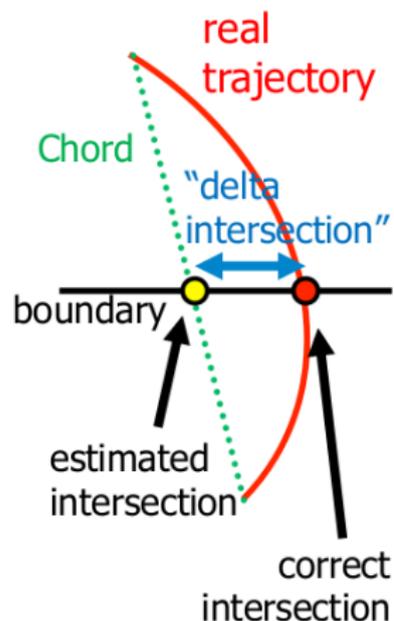
- Using the chords, G4Navigator is interrogated to check if a volume boundary was crossed.
- User can set the accuracy of the volume intersection by setting the *deltaChord* parameter, which is to be compared with the *missed distance*.
- If the missed distance is greater than *deltaChord*, the path is broken into more linear chords.



Transportation Chain in Geant4

Parameters Involved

- *deltaInt* controls the accuracy of the intersection with a volume boundary. It limits a bias on the algorithm (intersection point always *inside* of the curve).
- *deltaOneStep* ($\delta_{oneStep}$) controls the accuracy for the endpoint of an ordinary integration step (not crossing any boundary).
- *stepMax* controls the maximum step length.
- *epsilon* (ϵ) controls the error after a substep due to numerical integration. $\epsilon = \frac{\delta_{oneStep}}{h}$ ($h \leq stepMax$ is the current step length).
- *epsilonMin* and *epsilonMax* may override ϵ .



Transportation Chain in Geant4

Runge-Kutta

We describe here the Runge-Kutta method of 4th order, used by Geant4.

- Let $\dot{y} = f(t, y)$ and $y(t_0) = y_0$ be our initial problem.
- A step size $t_h > 0$ should be picked, which defines
$$y_{n+1} = y_n + \frac{t_h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$
$$t_{n+1} = t_n + t_h$$
- The k_i represent the increment based on the local slope (derivative) which is given by f , calculated at different points. They can be explicitly calculated.

Outline

- 1 Transportation Chain in Geant4
 - Introduction
 - Parameters Involved
 - Runge-Kutta
- 2 Quantized State System
 - Introduction to QSS
 - QSS Explained
 - QSS vs Runge-Kutta
- 3 Exploring Geant4 Parameters
 - Introduction
 - Varying epsilon
 - Varying stepMax
 - Varying deltaChord
- 4 Exploring QSS Parameters
- 5 Geometry crossings - Performance Comparison

Introduction to QSS

Motivating Example

- Consider the following second order system:

$$\begin{aligned}\dot{x}_1(t) &= x_2(t) \\ \dot{x}_2(t) &= -x_1(t)\end{aligned}\tag{1}$$

- We can introduce an approximation by state quantization:

$$\begin{aligned}\dot{x}_1(t) &= \text{floor}(x_2(t)) = \lfloor x_2(t) \rfloor = q_2(t) \\ \dot{x}_2(t) &= -\text{floor}(x_1(t)) = -\lfloor x_1(t) \rfloor = -q_1(t)\end{aligned}\tag{2}$$

- $x_1(t)$ and $x_2(t)$ correspond to the original state variables, whereas $q_1(t)$ and $q_2(t)$ correspond to the quantized state variables.

Introduction to QSS

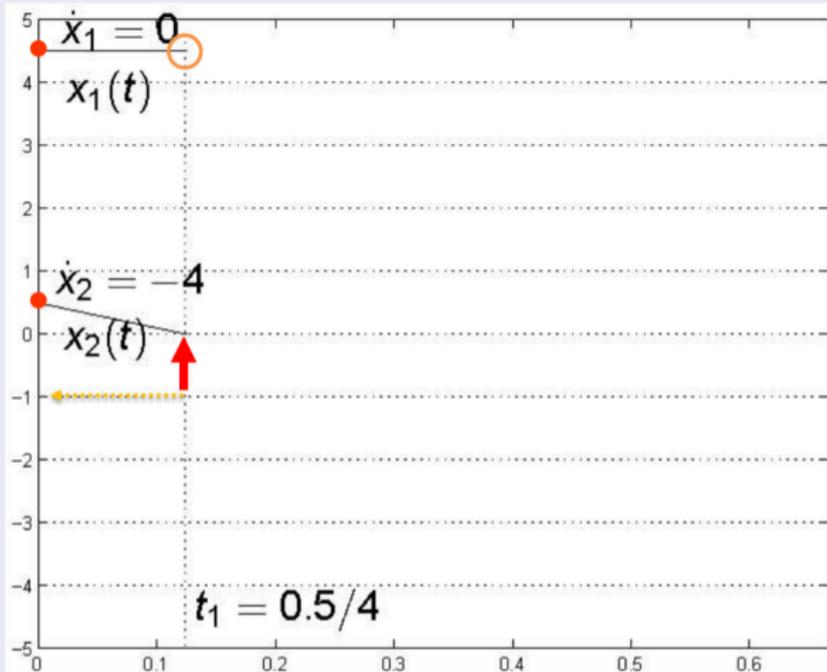
Simulation Explained Intuitively

Equation

$$\begin{aligned}\dot{x}_1(t) &= \lfloor x_2(t) \rfloor = q_2(t) \\ \dot{x}_2(t) &= -\lfloor x_1(t) \rfloor = -q_1(t)\end{aligned}$$

Initial Conditions

$$\begin{aligned}x_1(0) &= 4.5 \\ x_2(0) &= 0.5\end{aligned}$$



Introduction to QSS

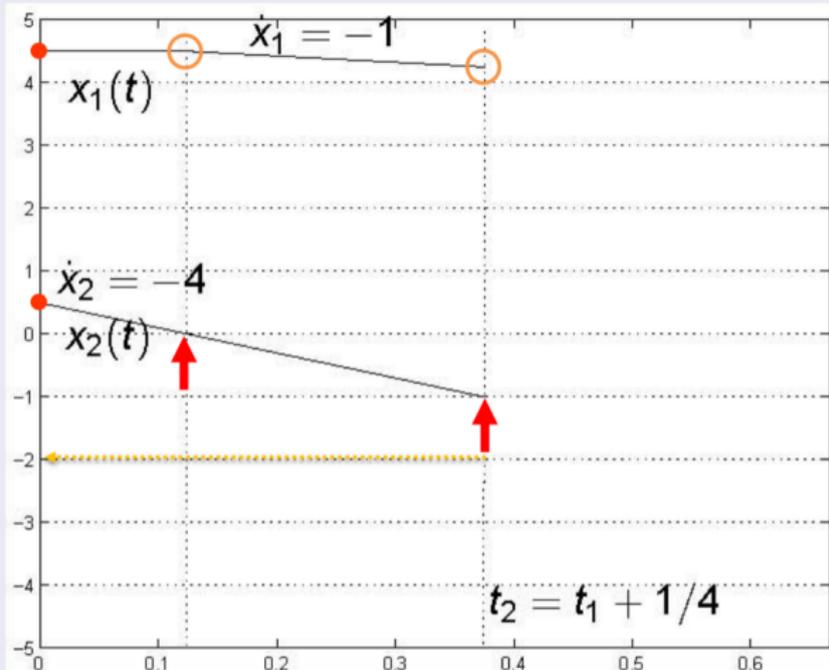
Simulation Explained Intuitively

Equation

$$\begin{aligned}\dot{x}_1(t) &= \lfloor x_2(t) \rfloor = q_2(t) \\ \dot{x}_2(t) &= -\lfloor x_1(t) \rfloor = -q_1(t)\end{aligned}$$

Initial Conditions

$$\begin{aligned}x_1(0) &= 4.5 \\ x_2(0) &= 0.5\end{aligned}$$



Introduction to QSS

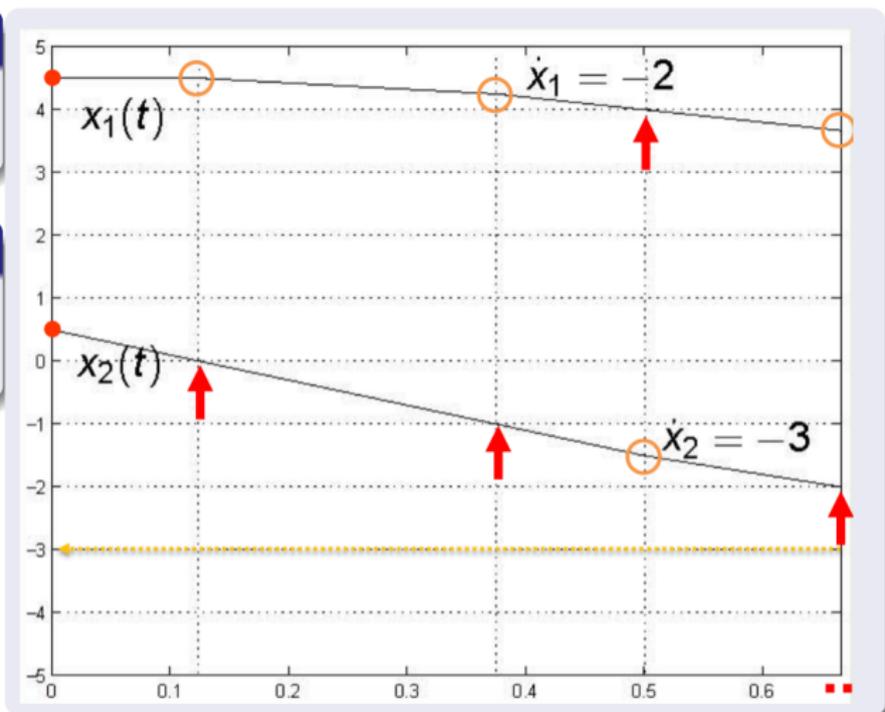
Simulation Explained Intuitively

Equation

$$\begin{aligned} \dot{x}_1(t) &= \lfloor x_2(t) \rfloor = q_2(t) \\ \dot{x}_2(t) &= -\lfloor x_1(t) \rfloor = -q_1(t) \end{aligned}$$

Initial Conditions

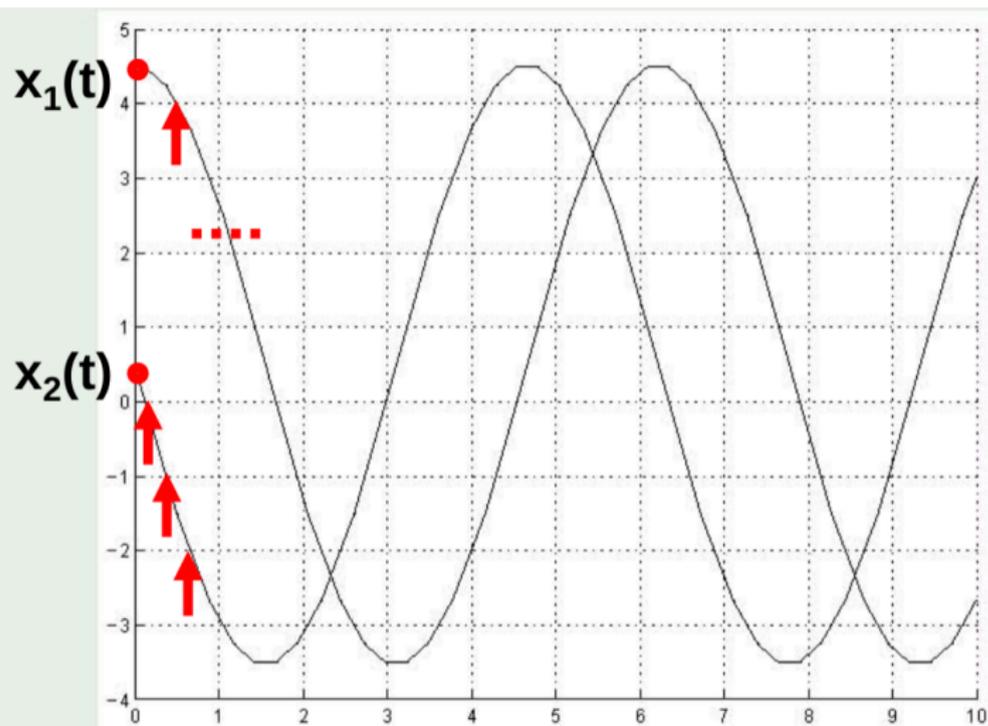
$$\begin{aligned} x_1(0) &= 4.5 \\ x_2(0) &= 0.5 \end{aligned}$$



Introduction to QSS

Simulation Explained Intuitively

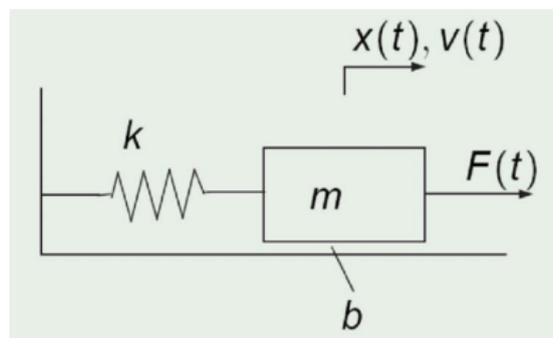
- Solution for the Quantized System



- Quantized State System (QSS) (Kofman E., 2001)
 - Family of numerical methods based on the principle of state quantization.
 - Quantize the state variables with discrete quanta in place of partitioning time in discrete steps.
 - Implemented in several general purpose Modeling and Simulation (M&S) tools (most advanced: PowerDEVS).
 - Recently also as a stand-alone QSS numerical solver (which we have already started using!).
 - Step is now a misleading term: QSS are inherently asynchronous and step-less methods.

QSS Explained

More Complex Example



ODE

$$\begin{aligned}\dot{x}(t) &= v(t) \\ \dot{v}(t) &= -\frac{k}{m}x(t) - \frac{b}{m}v(t) + \frac{1}{m}F(t)\end{aligned}$$

QSS

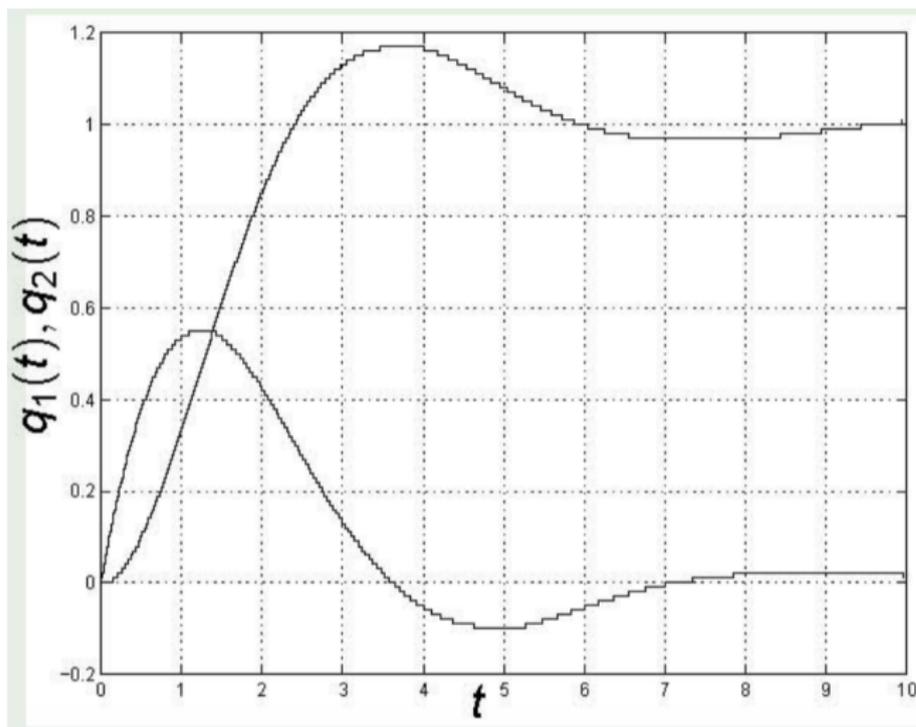
$$\begin{aligned}\dot{x}_1(t) &= q_2(t) \\ \dot{x}_2(t) &= -\frac{k}{m}q_1(t) - \frac{b}{m}q_2(t) + \frac{1}{m}F(t)\end{aligned}$$

For the simulation, we take $m = b = k = 1$.

QSS Explained

More Complex Example

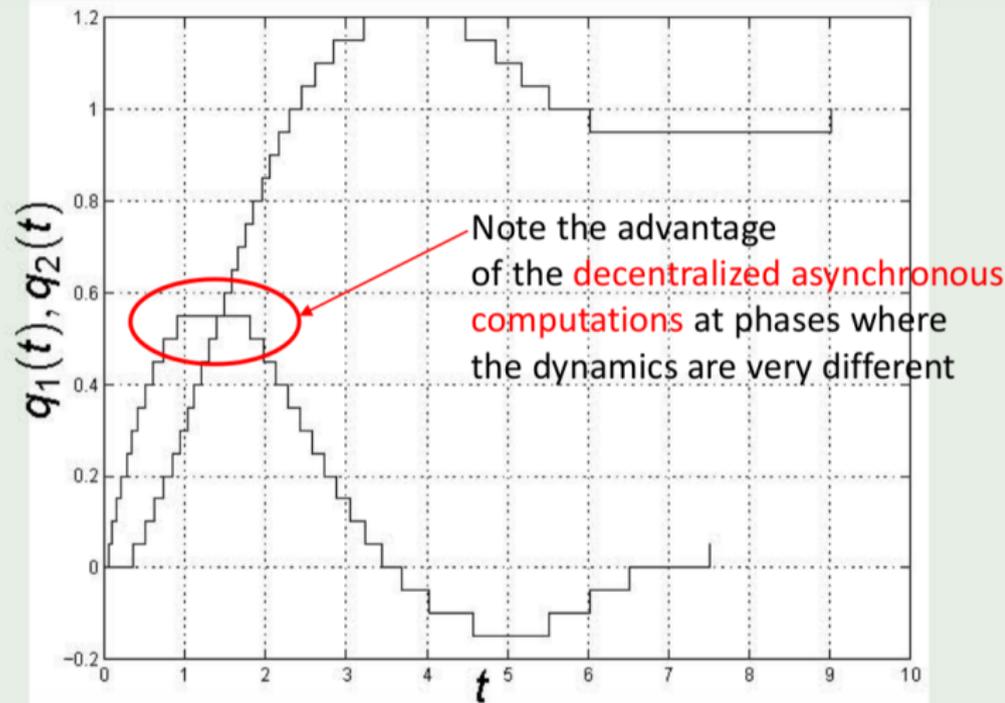
- QSS1 Solution with quantum size $\Delta Q = 0.01$



QSS Explained

More Complex Example

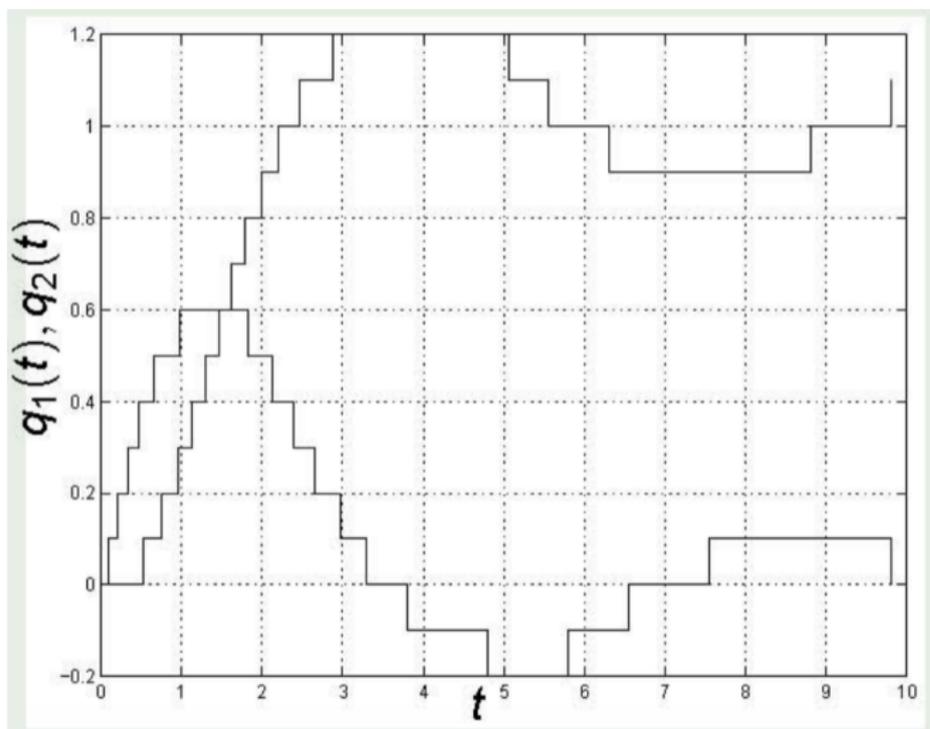
- QSS1 Solution with quantum size $\Delta Q = 0.05$



QSS Explained

More Complex Example

- QSS1 Solution with quantum size $\Delta Q = 0.1$



QSS Explained

Advantages and Disadvantages

- We have presented the first order accurate QSS (QSS1)
- Advantages:
 - Stability and Global Error Bound (at least for linear systems)
 - Decentralized dynamics (decoupled, independent computation of changes in state variables) which makes it naturally asynchronous.
 - Efficiently simulates heavily discontinuous systems
 - Dense polynomial output, with trivial detection and handling of discontinuities.
- Disadvantages
 - Oscillations may appear. Problems with stiff systems
Solution: Linearly Implicit LIQSS family methods
 - The Quantum must be chosen
Solution: Logarithmic (adaptive) Quantum controls the relative error automatically
 - The # of computational steps grows linearly with precision
Solution: Higher order methods QSS2, QSS3, QSS4

QSS Explained

Simple Realistic Model for Exploration

- To start studying the numerical methods, we focus on a simple example with a well defined trajectory and where the analytical solution is known.
- This will enable us to have a better intuition of the results.
- It will also enable us to compare our simulations with the real solution.
- Model: charged particle in a static, homogeneous magnetic field in \hat{z} .

ODE

$$\begin{aligned}\dot{\vec{x}}(t) &= \vec{v}(t) \\ m \frac{d}{dt} (\gamma \frac{d\vec{x}}{dt}) &= q(\vec{v} \times \vec{B})\end{aligned}$$

QSS

$$\begin{aligned}\dot{x}(t) &= q_3(t) & \dot{v}_x(t) &= \frac{qB}{m\gamma} q_4(t) \\ \dot{y}(t) &= q_4(t) & \dot{v}_y(t) &= -\frac{qB}{m\gamma} q_3(t)\end{aligned}$$

Simple Model for Exploration

Analytical solution: a circle in the x-y plane.

$$\vec{x}_0 = (0, 0, 0)$$

$$\vec{v}_0 = (0.999c)\hat{x}$$

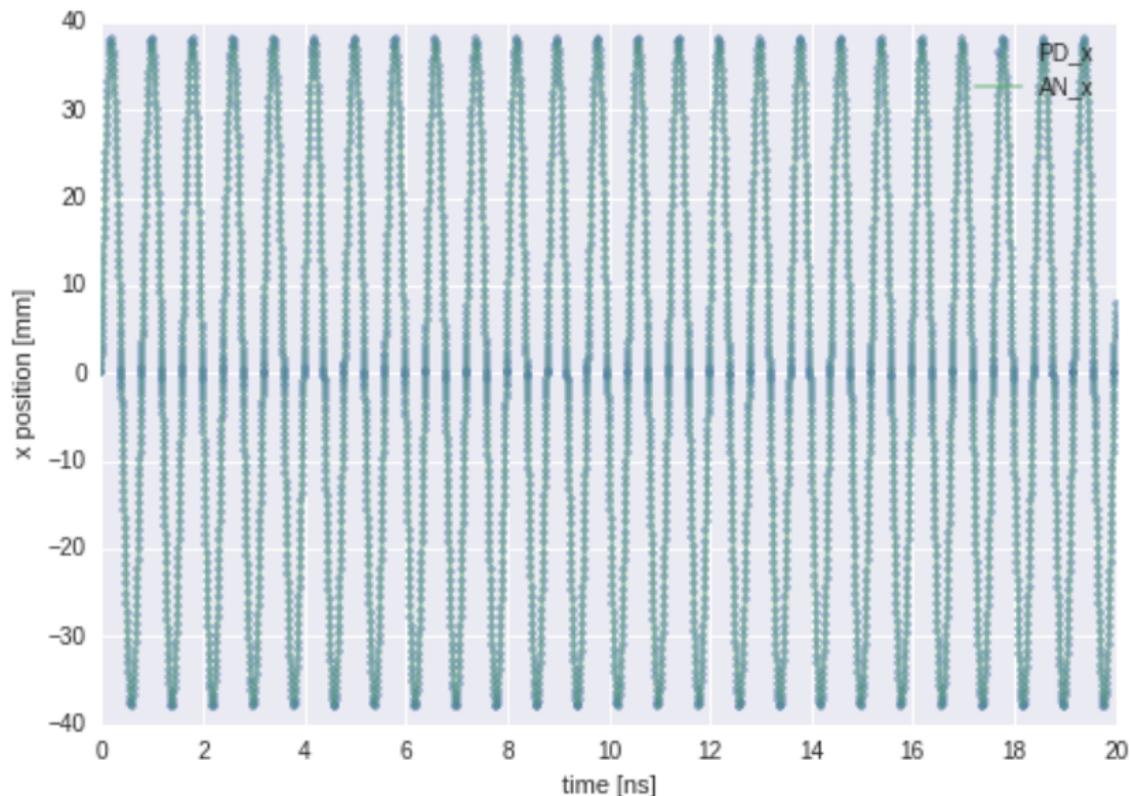
$$\vec{B} = B\hat{z} = (1.0\text{tesla})\hat{z}$$

$$\text{Radius} = 38.085\text{mm}$$

$$\text{Period} = 0.799\text{ns}$$

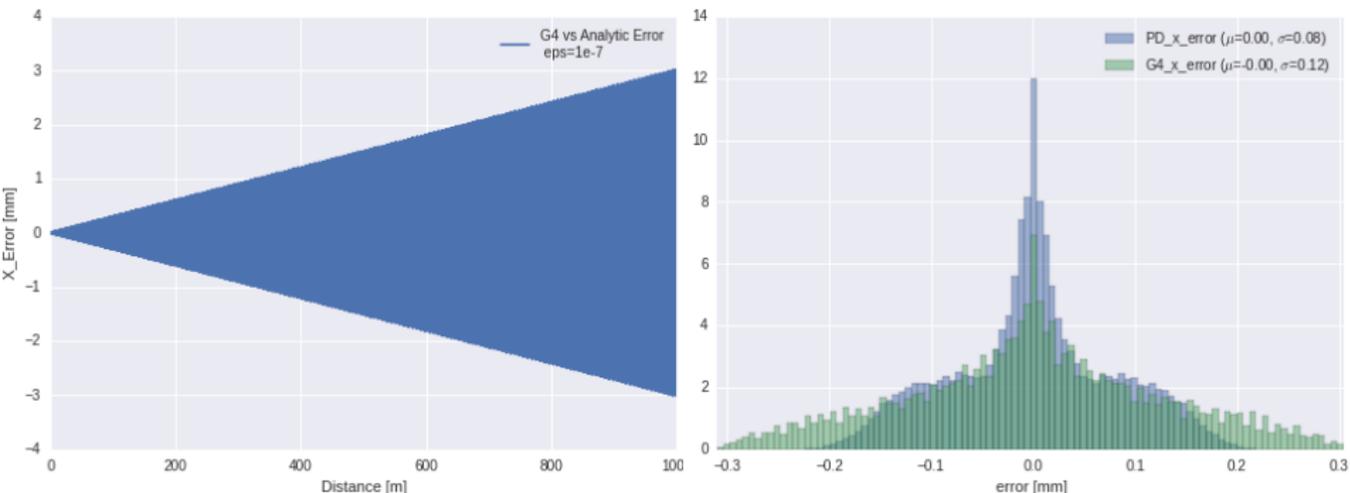
QSS Explained

Simple Realistic Model for Exploration - Simulation Results



QSS Explained

Simple Realistic Model for Exploration - Simulation Results



QSS

- Error is globally bounded.
- Dense output (with polynomials of n^{th} order).
- Only two parameters to tune.
- Naturally asynchronous, potentially good for parallelization.

Runge-Kutta

- Error is locally bounded.
- Non dense output.
- Multiple parameters to tune. Not so easy to understand how each parameter will affect the simulation.
- Parallelization requires changing the algorithm considerably.

Outline

- 1 Transportation Chain in Geant4
 - Introduction
 - Parameters Involved
 - Runge-Kutta
- 2 Quantized State System
 - Introduction to QSS
 - QSS Explained
 - QSS vs Runge-Kutta
- 3 Exploring Geant4 Parameters
 - Introduction
 - Varying epsilon
 - Varying stepMax
 - Varying deltaChord
- 4 Exploring QSS Parameters
- 5 Geometry crossings - Performance Comparison

Exploring Geant4 Parameters

Introduction

- We identified and focused on a set of Geant4 parameters which can be controlled and should impact the **simulation time** and **numerical error**.
- We ran some experiments to verify if the impact of these parameters in our simple example is the expected one.
- Understanding how the value of these parameters impacts on the simulation error and time will enable us to **compare** with QSS in a **fair way**: compare one setup for each method where the error is similar, and see which one performs better in terms of time.

Fixed Parameters

- $\text{deltaOneStep} = 1.0E-2\text{mm}$
- $\text{deltaInt} = 1.0E-5\text{mm}$

Experiments Varying Parameters

- epsilon
- stepMax
- deltaChord

Varying epsilon

Fixed Parameters

- $\text{deltaChord} = 0.25\text{mm}$
- $\text{stepMax} = 20\text{mm}$
- $\text{epsilonMin} = \text{epsilonMax} = \text{epsilon}$
- $r_RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^N (x_i - x_{pred})^2 + (y_i - y_{pred})^2}$
- r_RMSE is the **Root Mean Squared Error** for the position.

Experiment

- Vary epsilon from $1E-3$ to $1E-12$ and measure the impact on the simulation time and simulation error.
- We expect the simulation time to increase when epsilon becomes smaller (searching for more accuracy), whereas the error should decrease.

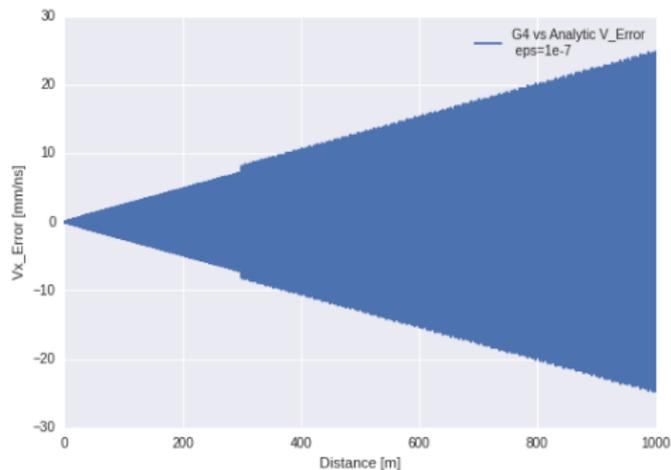
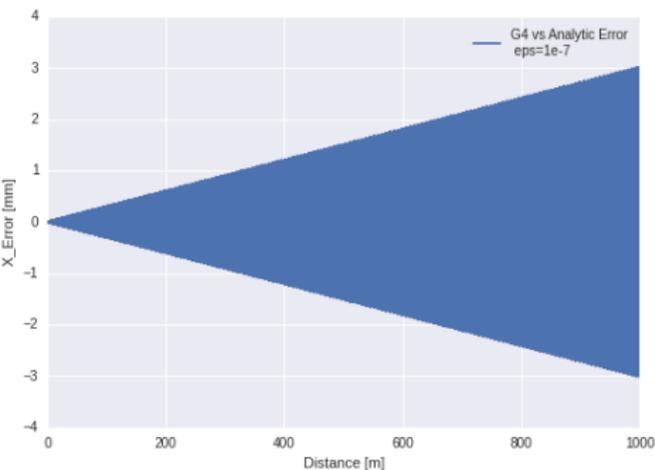
Varying epsilon

Results

epsilon	Simulation Time [s]	RHS evaluation steps	Output Steps	Ratio	r_error[mm]				
					1 km	100 m	10 m	1 m	100 mm
1.0E-3	13.72	1.65e7	500001	33	1.73E+0	1.73E-1	1.73E-2	1.72E-3	1.26E-4
1.0E-4	13.72	1.65e7	500001	33	1.73E+0	1.73E-1	1.73E-2	1.72E-3	1.26E-4
1.0E-5	13.72	1.65e7	500001	33	1.73E+0	1.73E-1	1.73E-2	1.72E-3	1.26E-4
1.0E-6	21.81	3.85e7	500001	76	1.74E+0	1.74E-1	1.74E-2	1.73E-3	1.30E-4
1.0E-7	27.98	6.05e7	500001	121	1.74E+0	1.74E-1	1.74E-2	1.74E-3	1.30E-4
1.0E-8	41.05	1.04e8	500001	207	1.74E+0	1.74E-1	1.74E-2	1.74E-3	1.30E-4
1.0E-9	55.83	1.59e8	500001	318	1.74E+0	1.74E-1	1.74E-2	1.74E-3	1.30E-4
1.0E-10	85.94	2.69e8	500001	537	1.74E+0	1.74E-1	1.74E-2	1.74E-3	1.30E-4
1.0E-11	139.99	4.56e8	500001	917	1.74E+0	1.74E-1	1.74E-2	1.74E-3	1.30E-4
1.0E-12	230.22	7.86e8	500001	1572	1.74E+0	1.74E-1	1.74E-2	1.74E-3	1.30E-4

Varying epsilon

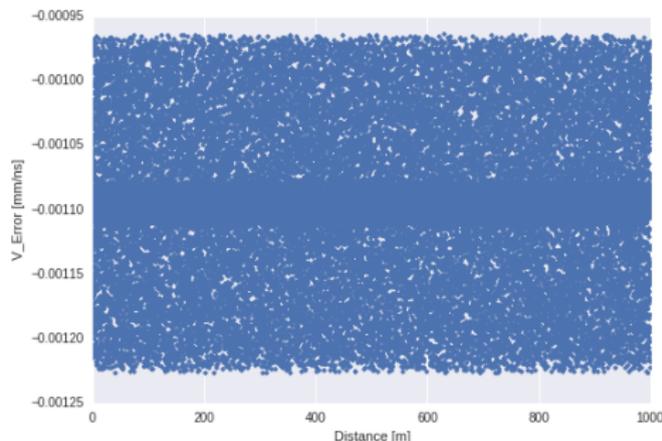
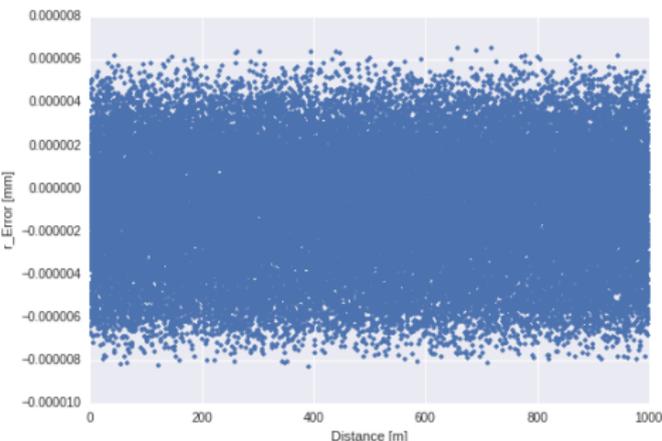
x_Error and v_x_Error vs distance



Varying epsilon

r_Error and v_Error vs distance

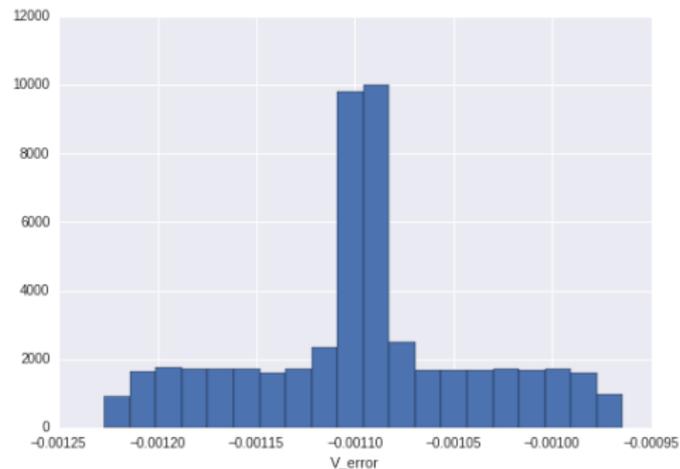
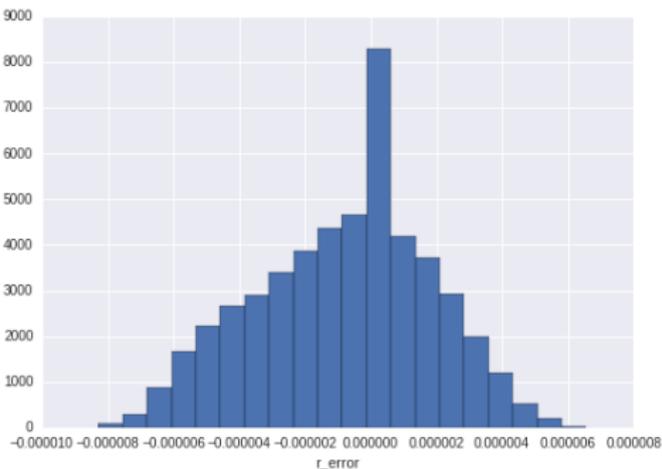
- We define r_Error as the distance between the simulated position and the circle.
- Similarly, we define v_Error as the difference between the simulated velocity and the theoretical (which should be constant).



- Whereas both error oscillate similarly in all the trajectory around the same value, v_Error is always negative, i.e. the particle is travelling slower in our simulations!

Varying epsilon

r_Error and v_Error histograms



Varying epsilon

Conclusions

- The simulation time increases when *epsilon* becomes smaller, as expected.
- The simulated velocity is always smaller than the theoretical one.
- The error gets larger as the particle travels. This is expected for systems with exact solutions.
- The error when compared to the analytical solution is not affected considerably by the variation of *epsilon*. This invariance is counterintuitive.
- We made several experiments to try to explain this behavior, but did not succeed.
 - Runge-Kutta may be too accurate for this simple example, so that epsilon may not be controlling the steps at all.
 - TODO: measure the accuracy of the individual Runge-Kutta steps to validate the previous hypothesis.

Varying stepMax

Fixed Parameters

- $\text{deltaChord} = 0.25\text{mm}$
- $\text{epsilonMin} = \text{epsilonMax} = 1.0E-3$

Experiment

- Vary stepMax and measure the impact on the simulation time and simulation error.
- Values[mm]: 0.2, 1, 2, 5, 10, 20, 50, 100, 150, 200
- We expect the simulation time to increase when stepMax becomes smaller (as we may be forcing more steps), whereas the error should decrease.

Varying stepMax

Results

stepMax	Simulation Time [s]	RHS evaluation steps	Output Steps	Ratio	r_error[mm]				
					1 km	100 m	10 m	1 m	100 mm
0.2	814.81	5.50e8	?	?	1.74E+0	1.74E-1	1.74E-2	1.76E-3	1.67E-4
1	171.02	1.10e8	1000000	111	1.74E+0	1.74E-1	1.74E-2	1.76E-3	1.66E-4
2	88.19	5.50e7	500000	11	1.74E+0	1.74E-1	1.74E-2	1.76E-3	1.64E-4
5	33.12	2.20e7	200000	11	1.74E+0	1.74E-1	1.74E-2	1.75E-3	1.58E-4
10	21.90	2.20e7	100000	22	1.74E+0	1.74E-1	1.74E-2	1.74E-3	1.45E-4
20	13.79	1.65e7	50000	33	1.73E+0	1.73E-1	1.73E-2	1.72E-3	1.26E-4
50	8.31	1.32e7	20000	66	1.71E+0	1.71E-1	1.71E-2	1.66E-3	6.91E-5
100	7.17	1.32e7	10000	132	1.68E+0	1.68E-1	1.67E-2	1.56E-3	
150	6.84	1.32e7	66668	198	1.64E+0	1.64E-1	1.64E-2	1.58E-3	
200	6.38	1.26e7	50001	252	1.61E+0	1.61E-1	1.59E-2	1.36E-3	

Varying stepMax

Conclusions

- The simulation time increases when *stepMax* becomes smaller, as expected.
- The error gets larger as the particle travels. This is expected for systems with exact solutions.
- We observe the error changes when varying *stepMax*, but in a counterintuitive way: for smaller values of *stepMax* the error is larger.

Fixed Parameters

- $stepMax = 20mm$
- $epsilonMin = epsilonMax = 1.0E-3$

Experiment

- Vary *deltaChord* and measure the impact on the simulation time and simulation error.
- Values[mm]: 0.01, 0.05, 0.25, 0.5, 1.0
- We expect the simulation time to increase when *deltaChord* becomes smaller (as we may be forcing more steps), whereas the error should decrease.

Varying deltaChord

Results

deltaChord	Simulation Time [s]	RHS evaluation steps	Output Steps	Ratio	r_error[mm]				
					1 km	100 m	10 m	1 m	100 mm
0.01	3.9	6.60e6	? 50001	132	1.74E+0	1.74E-1	1.74E-2	1.74E-3	1.30E-4
0.05	2.3	3.30e6	50001	66	1.74E+0	1.74E-1	1.74E-2	1.74E-3	1.30E-4
0.25	1.5	1.65e6	50001	33	1.73E+0	1.73E-1	1.73E-2	1.72E-3	1.26E-4
0.5	1.3	1.10e6	50001	22	1.70E+0	1.70E-1	1.70E-2	1.68E-3	1.14E-4
1.0	1.3	1.10e6	50001	22	1.60E+0	1.60E-1	1.59E-2	1.52E-3	5.27E-5

Varying *deltaChord*

Conclusions

- The simulation time increases when *deltaChord* becomes smaller, as expected.
- The error gets larger as the particle travels. This is expected for systems with exact solutions.
- We observe the error changes when varying *deltaChord*, but in a counterintuitive way: for smaller values of *deltaChord* the error is larger.

Outline

- 1 Transportation Chain in Geant4
 - Introduction
 - Parameters Involved
 - Runge-Kutta
- 2 Quantized State System
 - Introduction to QSS
 - QSS Explained
 - QSS vs Runge-Kutta
- 3 Exploring Geant4 Parameters
 - Introduction
 - Varying epsilon
 - Varying stepMax
 - Varying deltaChord
- 4 Exploring QSS Parameters
- 5 Geometry crossings - Performance Comparison

Varying δQ_{Rel} and δQ_{Min}

Settings

- The simulation was done with the QSS3 method.
- $\delta Q_{Min} = \delta Q_{Rel} * 1E-3$
- These simulations were done using PowerDevs

Experiment

- Vary δQ_{Rel} (and δQ_{Min} accordingly) and measure the impact on the simulation time and simulation error.
- Values for δQ_{Rel} : $1E-3$, ..., $1E-7$
- We expect the simulation time to increase when δQ_{Rel} becomes smaller (as we are forcing a better accuracy). In fact, we expect it to increase with the cubic root of the decrease factor.
- We expect the error to decrease when we make δQ_{Rel} smaller.

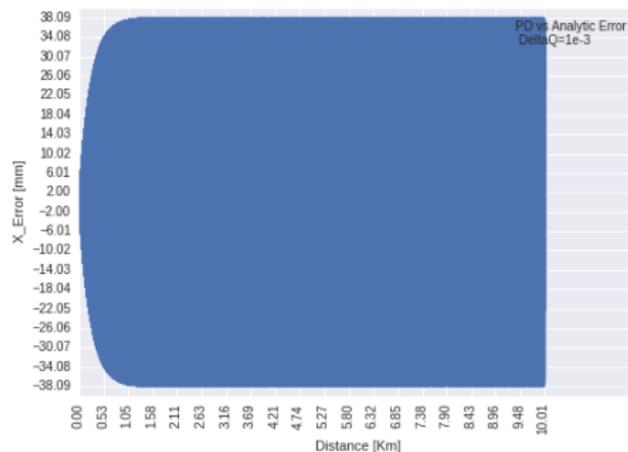
Varying deltaQRel and deltaQMin

Results

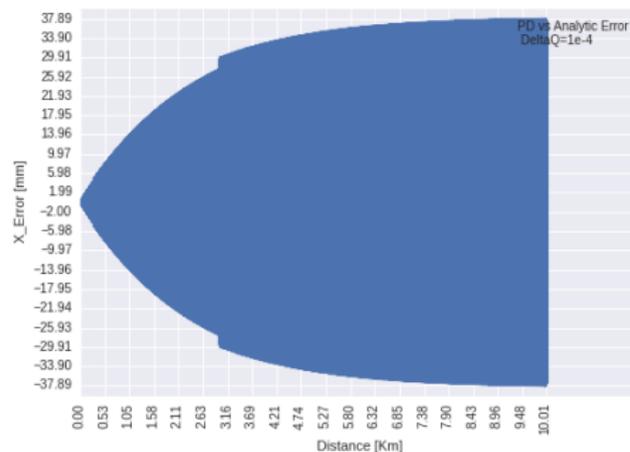
DeltaQ	Simulation Time [s]	PD evaluation steps	r_error[mm]				
			1 km	100 m	10 m	1 m	100 mm
1.0E-3	1.33	2.40e6	2.17E+1	6.11E+0	7.48E-1	8.17E-2	7.27E-3
1.0E-4	8.89	1.42e7	6.44E+0	7.89E-1	8.28E-2	8.65E-3	8.32E-4
1.0E-5	16.3	3.08e7	1.09E+0	1.13E-1	1.14E-2	1.18E-3	1.09E-4
1.0E-6	35.6	6.64e7	7.68E-1	7.69E-2	7.71E-3	7.64E-4	8.00E-5
1.0E-7	77.6	1.43e8	7.64E-1	7.65E-2	7.62E-3	7.67E-4	7.64E-5

Varying deltaQRel and deltaQMin

Error in the Position in \hat{x}



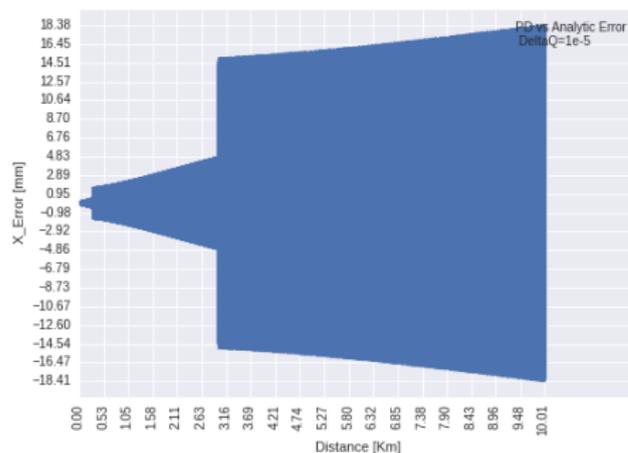
: (a) $\Delta Q_{Rel} = 1E-3$



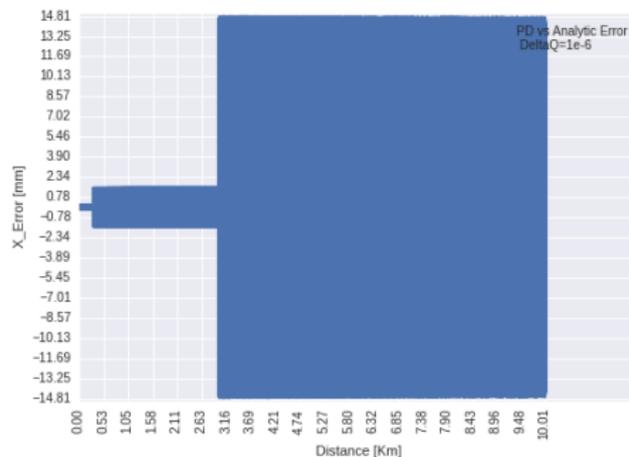
: (b) $\Delta Q_{Rel} = 1E-4$

Varying deltaQRel and deltaQMin

Error in the Position in \hat{x}



: (c) DeltaQRel=1E-5



: (d) DeltaQRel=1E-6

Varying δQ_{Rel} and δQ_{Min}

Conclusions

- The simulation time increases when δQ_{Rel} becomes smaller.
- It grows with the cubic root of δQ_{Rel} decrease factor, as expected.
- The error grows unbounded as the particle travels. This is expected when using numerical methods not specifically designed for marginally stable systems (such as our simple model or the armonic oscillator previously selected as the baseline case-study).
- We observe the error is reduced when we decrease the value of δQ_{Rel} , as expected.

Outline

- 1 Transportation Chain in Geant4
 - Introduction
 - Parameters Involved
 - Runge-Kutta
- 2 Quantized State System
 - Introduction to QSS
 - QSS Explained
 - QSS vs Runge-Kutta
- 3 Exploring Geant4 Parameters
 - Introduction
 - Varying epsilon
 - Varying stepMax
 - Varying deltaChord
- 4 Exploring QSS Parameters
- 5 Geometry crossings - Performance Comparison

Performance comparison for different numerical methods

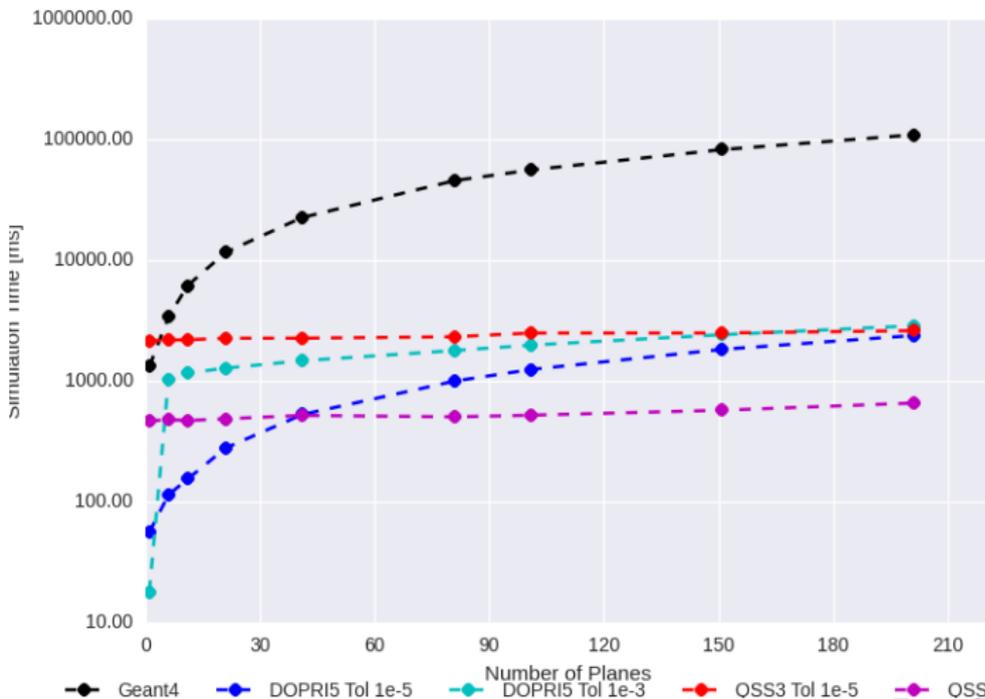
Varying number of planes crossed in a period

Settings

- We set equispaced rectangles which intersect with the particle trajectory in the \hat{x} plane.
- We used the G4VPReplica method in Geant4.
- We implemented the geometry crossing detection in QSS Solver.
- We will use 3 methods for our simulations: QSS3, Geant4, and DOPRI5. We will also change the parameter which controls the error in QSS3 and DOPRI5.
- DOPRI5 is a Runge-Kutta based implementation which works within the QSS Solver and which is of 5th order.
- We run simulations where the particle crosses a different number of planes per period: 0, 5, 10, 20, 40, 80, 100, 150, 200.
- The goal is to measure the time and error for each simulation.

Performance comparison for different numerical methods

Raw Simulation Times

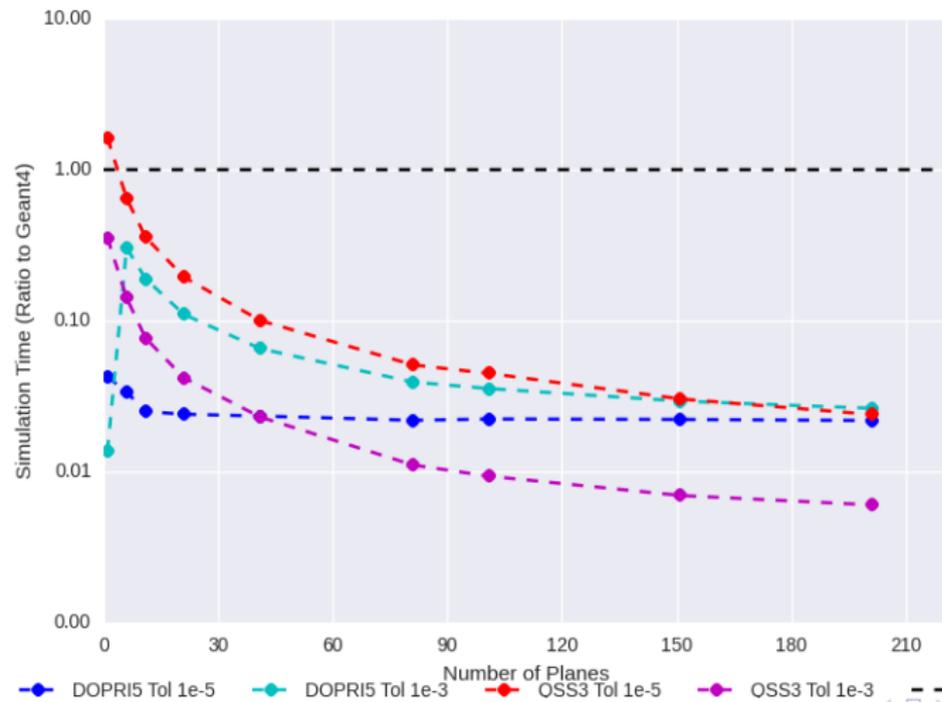


Conclusions

- Solver faster than Geant4
- QSS3 is better than DOPRI5 to handle planes. The latter is better than Geant4.
- How much better?

Performance comparison for different numerical methods

Simulation Times Ratio (Geant4 as baseline)

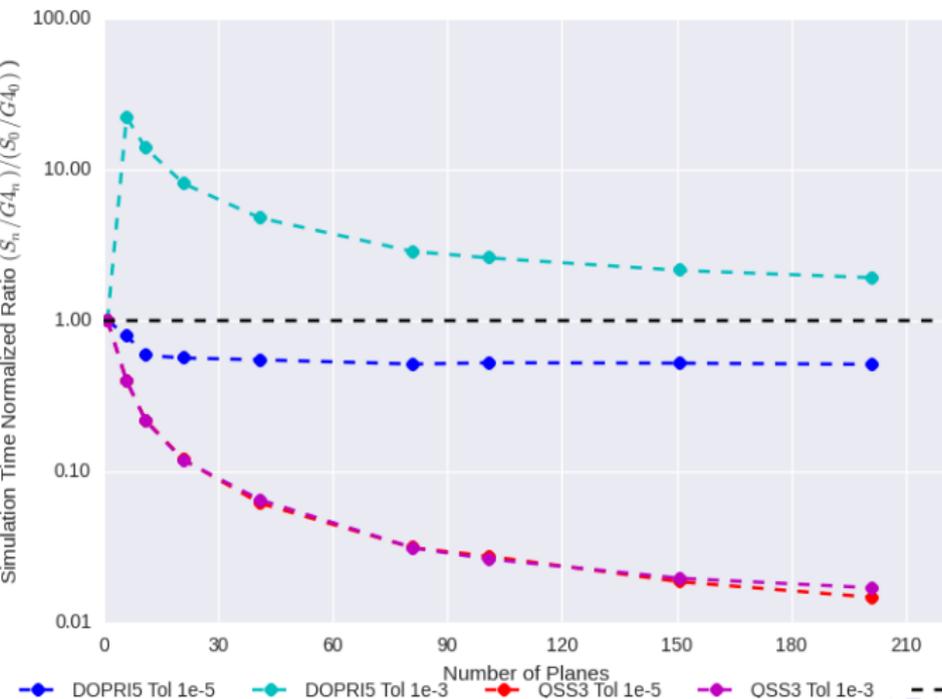


Conclusions

- QSS3 seems much more efficient for large number of planes
- QSS3 seems two orders of magnitude better than Geant4!!
- Is it a fair comparison?
- We may not be considering some overhead on Geant4.

Performance comparison for different numerical methods

Simulation Times Normalized Ratio $(S_n/G4_n)/(S_0/G4_0)$

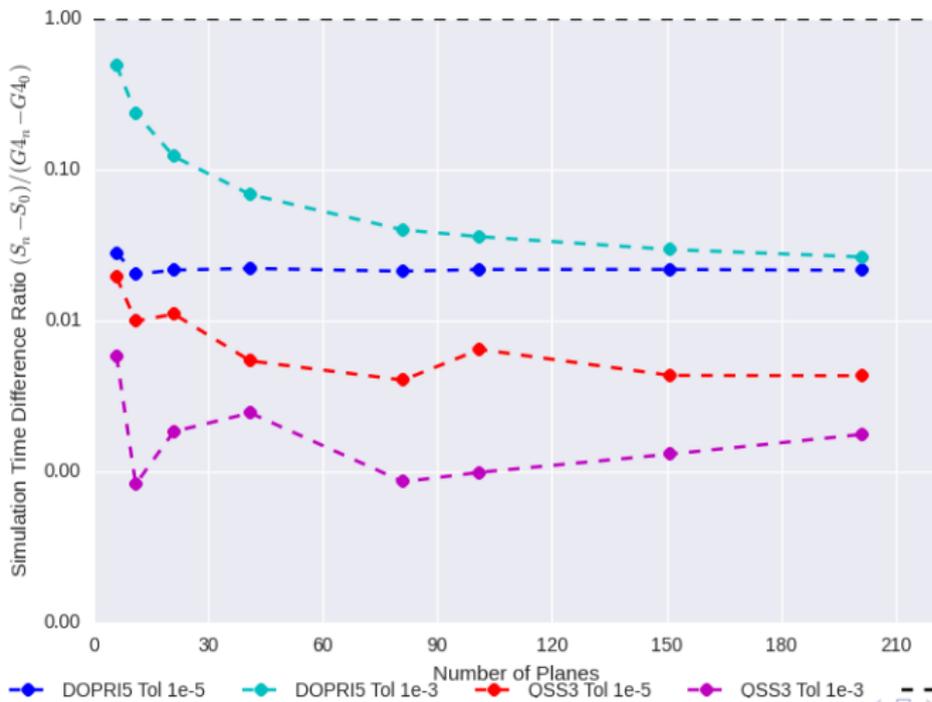


Conclusions

- QSS3 is still between one and two orders of magnitude better than Geant4!!
- If time is linear with the number of planes, why is the ratio not constant?
- Let $S_n = a_1 n + a_2$ and $G4_n = b_1 n + b_2$
- This plot is expected if $a_1 n \ll a_2$ which is true.

Performance comparison for different numerical methods

Simulation Times Difference Ratio $(S_n - S_0)/(G_{4n} - G_{4_0})$



Conclusions

- QSS3 is now at least two orders of magnitude better than Geant4!!
- The ratios look more similar to constants.
- TODO: statistic to get cleaner plots and errors.

Conclusions

- All the numerical methods used (modified Runge-Kutta in Geant4, QSS3 and Dopri5) can correctly simulate our simple model.
- In all cases, the predictive power of the methods gets worse with the distance travelled by the particle.
- We understand how to vary parameters in QSS3 to improve the error of the simulation. We do not know how to do this consistently in Geant4 yet.
- **QSS3** deals with geometry crossings in a **more efficient** way than **Geant4**. Our results show it is **two orders of magnitude** better.
- Short term objectives:
 - Study the impact of δt_{int} in Geant4, as it may explain the bad performance of Geant4 when crossing geometries.
 - Check how QSS3 would deal with other complex situations: varying magnetic field, particle decays, etc.
- Mid term objectives:
 - We would also like to **implement QSS3 within Geant4**.

Acknowledgments

- Lucio Santi, Discrete Event Simulation Group, Computer Science Department, University of Buenos Aires
- Rodrigo Castro, Discrete Event Simulation Group, Computer Science Department, University of Buenos Aires
- Soon Yung Jun, FNAL
- Krzysztof Genser, FNAL
- Daniel Elvira, FNAL
- John Apostolakis, CERN

Questions?



 J. Apostolakis.
Geant 4, Detector Description: EM Fields.

 F. Cellier, E. Kofman
Continuous System Simulation, Springer, New York, 2006.