PRECISION SPIN TRACKING FOR ELECTRIC DIPOLE MOMENT SEARCHES

M. Gaisser[1,2,*], S. Hacıömeroğlu[1,2], Y.I. Kim[1,2], S. Lee[1,2], Y.K. Semertzidis[1,2,3]
[1]Center for Axion and Precision Physics (CAPP), Daejeon, South Korea
[2]Institute for Basic Science (IBS), Daejeon, South Korea
[3]Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea

Abstract

Several proposed storage ring electric dipole moment (EDM) searches as well as muon g-2 experiments require a precise understanding of the evolution of the spin during the experiment, both theoretically as well as experimentally in order to understand systematic errors. Here we present the computational challenges of these experiments, our way of dealing with them and a comparison of analytical benchmarking cases with simulation results. In the end we give a short overview of future improvements of our program.

INTRODUCTION

Spin is probably the least used particle property in accelerator physics although it offers paths to precision physics that can yield information about physics beyond the Standard Model. For both, muon g-2 as well as storage ring electric dipole moment (EDM) experiments, the relevant physical information is encoded in the speed and direction of the spin rotation and proper beam preparation, beam control and polarization measurements are therefore crucial for the measurements. Furthermore it is essential to understand all unwanted sources of spin rotation (systematic errors), their magnitude and possible ways of their elimination. Because of the complexity of the equations, this can in general only be achieved with simulations and it has to be shown that their results are reliable.

SPIN EVOLUTION

The spin evolution of a particle with magnetic moment $\vec{\mu}$ and electric dipole moment $\vec{d}$ in the particles rest frame is given by

$$\frac{d\vec{S}}{dt} = \vec{\mu} \times \vec{B}^* + \vec{d} \times \vec{E}^*,$$

where $\vec{B}^*$ and $\vec{E}^*$ are the magnetic and the electric field in the particles rest frame. Using Lorentz transformations for the fields and taking the acceleration of the rest frame into account, one arrives at the famous T-BMT equation which in accelerator coordinates parametrized by the arc-length s looks as follows:

$$\begin{aligned}
\frac{d\vec{S}}{ds} &= \left(\frac{1}{\dot{s}}\left(\vec{\Omega}_\mu + \vec{\Omega}_d\right) - \vec{\kappa} \times \vec{e}_s\right) \times \vec{S} \\
\vec{\Omega}_\mu &= \frac{q}{m}\left(\left(G + \frac{1}{\gamma}\right)\vec{B} - \frac{G\gamma}{\gamma+1}(\vec{\beta}\cdot\vec{B})\vec{\beta}\right) \\
&\quad - \frac{q}{m}\left(G + \frac{1}{\gamma+1}\right)\frac{\vec{\beta}\times\vec{E}}{c} \\
\vec{\Omega}_d &= \frac{\eta q}{2m}\left(\vec{\beta}\times\vec{B} + \frac{\vec{E}}{c} - \frac{\gamma}{\gamma+1}\frac{\vec{\beta}\cdot\vec{E}}{c}\vec{\beta}\right)
\end{aligned}$$

Almost every quantity depends on the position and/or time: The fields $\vec{E}$ and $\vec{B}$, the relativistic factors $\vec{\beta} = \vec{v}/c$ and $\gamma = 1/\sqrt{1-\beta^2}$ (only in electric fields) as well as the curvature $\vec{\kappa} = (1/R(s), 0, 0)$ (for flat rings of bending radius $R(s)$) and the longitudinal unit vector $\vec{e}_s$. $G = (g-2)/2$ is the anomalous magnetic moment while the dimensionless parameter η is given by $\vec{d} = \eta\frac{q}{2mc}\vec{S}$ in analogy to the g-factor for the magnetic dipole moment. Its magnitude is about $\eta \approx 2 \cdot 10^{-15}$ for an assumed proton EDM of 10^{-29}e · cm. $1/\dot{s} = 1/(ds/dt) = dt/ds$ is the inverse of the longitudinal velocity of the particle. The fields have to be evaluated at the respective position of the particle which introduces an additional dependence on the transversal coordinates and closely couples this equation to the equation of motion. All of that, together with a possible explicit time-dependence for rf-fields, makes this a very complicated equation which in general cannot be solved analytically.

SYSTEMATIC ERRORS IN PROTON EDM EXPERIMENT

The proton EDM experiment is proposed for an all-electric ring with the proton momentum at its "magic" value $p = \frac{mc}{\sqrt{G}} \approx 0.7$ GeV/c such that there are no spin rotations due to the magnetic dipole moment in the rest frame of the ideal particle [1]. A possible electric dipole moment will cause a spin rotation out of the horizontal plane with an expected rate of a few nrad/s for an assumed value of $d = 10^{-29}$ e · cm. A net radial magnetic field of only a few aT would cause a similar effect. Shielding to this level is not possible and additional strategies have been developed to deal with the issue, most notably the use of counter-rotating beams with a low and modulated vertical tune such that the oscillating vertical beam separation caused by the radial B-field could be measured with SQUID-based beam position monitors. In case of strong local spin rotations, e.g. due to misalignments

* gaiwa84@ibs.re.kr

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-136-6

or field errors, that may almost average out along the ring, one also has to consider the "geometric phase effect", i.e. the fact that rotations do not commute and hence a sequence of rotations around different axes generally does not cancel completely. Other systematic errors might be caused by a possible polarization profile of the beam or an unsuitable extraction scheme.

OUR APPROACH

Several codes based on different ideas like mapping codes using differential algebra or truncated power series algebra, kick-bend algorithms or Romberg integration are in use around the world. Some of these codes have difficulties with explicitly time-dependent fields or difficult field configurations. In general every code has to make various approximations and this typically originates either in a difficulty to model the real world or a trade-off between accuracy and speed.

In order to force as little approximations as possible, we choose a very simple but general approach and integrate the equation of motion as well as the T-BMT equation numerically. To this end many different algorithms are available with different characteristics. In our program we implement several of them with the aim of testing and benchmarking them with respect to accuracy and speed. Standard algorithms like the fourth order Runge-Kutta algorithm will be compared to newer ones and great emphasis is placed on the modular implementation in C++ for maximal flexibility. VexCL [2] (Vector Expression Template Library for OpenCL/CUDA) is used for parallelization with either CUDA or OpenCL as backends for a wide range of different hardware. VexCL is a header-only library that strives to reduce boilerplate code and supports multi-device and multi-platform computations. OpenCL has the advantage of being able to run on different hardware from various manufacturers while CUDA may be faster but depends on NVIDIA GPUs. Furthermore, VexCL is easy to learn and was demonstrated to work with Boost-Odeint [3, 4].

Our program also comes with the option of changing the data type for all floating point operations and allows the use of arbitrary precision numbers via the Boost-Multiprecision library with various backends in order to investigate errors caused by limitations of the machine precision.

CHOICE OF ALGORITHMS

Currently our program uses several algorithms from the Boost-Odeint library, some of which can be used with step size control or as dense output steppers meaning that a large step size can be used for the calculation while output at intermediate points is obtained via interpolation. Both concepts can help to speed up the calculation and improve the accuracy. Not all solvers from the library can be used however, since we currently use the hard edge approximation for fields which causes discontinuities that cannot be dealt with by some algorithms. This excludes for example the class of multistep solvers which can be very fast and accurate because they require very few (possibly costly) evaluations of the right hand side of the differential equation while they may still be of high order. Fortunately, A. Nordsieck discovered another way of expressing the same concept which solves all the problems associated with multistep methods [5]. This may become implemented in our program in the future.

So far also the symplectic solvers have to be excluded since our equation of motion is based on the Lorentz force equation and the physical fields, while symplectic solvers make use of the Hamiltonian approach with the potentials of the fields. This issue will certainly be addressed in the near future and both ways of expressing the differential equation may coexist within the code.

SYMPLECTICITY

All of our currently implemented algorithms are not symplectic meaning they neither conserve energy nor angular momentum. For short term simulations this is not a real concern since one can enforce approximate correctness with smaller step sizes. It however seriously limits the capabilities for precise long term tracking since the errors add up over time and small step sizes make the program slow. On the other hand it offers an additional path to estimating the accuracy of the result by looking at the energy or angular momentum drift.

The issue generally originates in the formulation of these algorithms where a new state vector $\vec{x}_{n+1}$ is calculated from an old one $\vec{x}_n$ by a scheme like

$$\vec{x}_{n+1} = \vec{x}_n + \Delta\vec{x}, \tag{1}$$

whereas mathematically the evolution is described by a rotation $\mathbf{R}$ in a suitable space such that a new point of the solution is obtained via a scheme like

$$\vec{x}_{n+1} = \mathbf{R} \cdot \vec{x}_n. \tag{2}$$

In eq. 1 the numerical errors will grow over time while they will average out for the symplectic case in eq. 2. It should be noted however that symplectic methods only guarantee that the length of a vector remains constant while its direction may still be off. Nonetheless, for long term simulations symplectic methods seem to be a crucial requirement since the spin evolution is tightly tied to the orbital motion. Errors in the solution of the phase space coordinates will therefore directly radiate into the spin part and the nonlinear behavior there may make things even worse.

PERFORMANCE

Some previous integration codes made use of cartesian coordinates with time as independent parameter, see e.g. [6]. This required very small step sizes of $dt \leqslant 10^{-11}$ s for precise results. It is however known that the arc length parametrization (of the individual particle) has much more favorable properties [7] and one may expect that the accelerator coordinates come close to this. Our new program employs these coordinates and tests show that generally a step size

ISBN 978-3-95450-136-6

Copyright © 2015 CC-BY-3.0 and by the respective authors

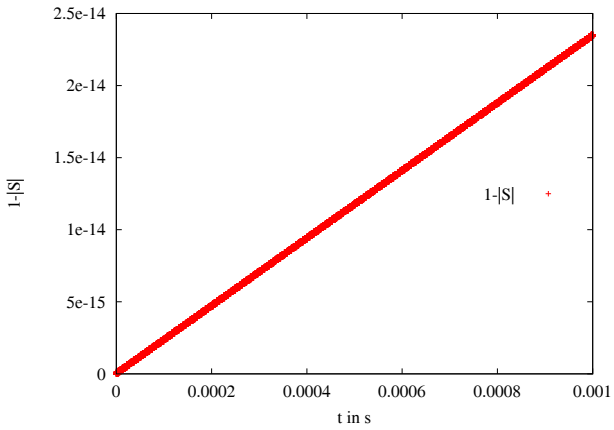

Figure 1: Loss of spin of a muon during the simulation of one millisecond with the 8th order Runge-Kutta algorithm with a step size $dt = 10^{-8}$ s and data type *long double* in the Fermilab muon g-2 ring. For a step size of $dt = 3 \cdot 10^{-9}$ s (not shown) the error remains below $3 \cdot 10^{-16}$ for the complete time interval.

corresponding to $dt = 10^{-8}$s yields very good results, see Fig. 1. This almost directly corresponds to a performance gain of a factor of thousand. With this parametrization it generally takes a few seconds to simulate a particle for one millisecond when a native C++ data type like *double* or *long double* is used. Using arbitrary precision data types makes the code several times slower, e.g. by approximately a factor of 4 for data type *float128* (quad precision) and a factor of 10-15 for *mpfr<N>* data types with N significant digits.

The parallel version of the program cannot use arbitrary precision data types yet and is far from optimal performance. It is however still possible to simulate 10.000 particles for a millisecond on two Intel Xeon E5-2630 hexacore processors in the course of a few hours. Significant improvement on this can be expected from using VexCLs kernel generation capability and the use of GPUs as accelerators.

BENCHMARKING

Every precision tracking program has to be tested very accurately. A recent paper [8] describes several high precision benchmarks with effects that can be calculated analytically which makes them ideal test cases for precision codes. One of the most precise tests of the correctness of an algorithm solving the T-BMT equation is the so-called pitch effect. This is a frequency change of the g-2 precession of the spin in a magnetic field when the particle undergoes vertical betatron oscillations, i.e. when there is a component of the B-field parallel to the velocity. The effect was first considered in [9] for a muon g-2 experiment and it was shown in [8] that our tracking programs are so precise that second order terms have to be included in the analytical calculation and the obtained results are matched to sub-ppb level, see Table 1.

Table 1: Comparison of Pitch Correction: in ppb between analytical estimates and tracking results for a muon with $\gamma = 29.3$ in a magnetic ring of radius $r = 7.112$ m with vertical focusing index n and maximal pitch angle $\theta_0 = 0.5$ mrad.

n	Estimated (ppb)	Tracking (ppb)
0.01	1.1	1.0
0.02	2.4	2.4
0.03	3.7	3.6
0.05	6.4	6.4
0.08	10.7	10.8
0.10	13.7	13.7
0.137	19.7	19.9
0.237	38.7	38.8

Table 2: Comparison of Tracking Results and Analytical Calculations of the EDM Signal and the Systematic Error (both in rad/s) Due to a Misalignment of the RF Wien Filter of an Angle $\theta = 0.1$ mrad for the Deuteron Case

p [GeV/c]	EDM tracking	EDM analytical	Syst. error tracking	Syst. error analytical
0.7	-1.00	-1.00	0.41	0.41
1.4	-0.74	-0.73	0.175	0.17
2.1	-0.50	-0.51	0.096	0.097
2.8	-0.36	-0.35	0.063	0.06

Another test described in detail in [10] is a systematic error estimation for a magnetic ring with an rf Wien filter that is operated at the $g - 2$ frequency and tilted by a small angle versus the ideal vertical direction. With perfect alignment, the Wien filter produces a vertical magnetic field and a radial electric field such that the Lorentz force cancels. In the presence of an EDM it would however cause an approximately linear build-up of a vertical component of the spin that is proportional to the EDM. In the non-ideal case when the Wien filter is misaligned with respect to the vertical direction, the B-field produced by the Wien filter also has a radial component which also causes a spin rotation out of the plane. Both effects were estimated analytically and were also simulated for the proton and deuteron case. Table 2 shows the results for the deuteron case with an estimated EDM of $d = 10^{-18}$ e · cm and an angular misalignment of $\theta = 0.1$ mrad of the rf Wien filter of 0.1 m length and 30 MV/m electric field strength.

FUTURE WORK

There are great perspectives for the new program described above but there also remains a lot of work to be

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-136-6

done in the future. Conceptually one of the most important improvement is the inclusion of geometric algorithms to enforce symplecticity. Algorithms based on the Magnus expansion [11] appear to be very efficient and precise at the same time and can be constructed of high order. These will be implemented in the near future.

Another very interesting option for precise long term tracking is the use of algorithms with global error estimation or control. So far, only algorithms are used that can adjust their step size such that a local error requirement set by the user is met. This feature can significantly increase the execution speed of the program but the local errors still build up over time. Recent developments [12–14] show that significantly better results may be obtained with different classes of algorithms when global error control algorithms are used in step size adjustment schemes. Although this decreases the speed of the method it may prove very useful for our purpose and will be implemented later.

A third issue that requires some work will be the optimization of the parallel program version with the help of VexCLs kernel generation facility. Currently each command is compiled during the execution of the program into a so-called compute kernel. Together with the cost of starting these kernels this is a major source of overhead. VexCL offers however the possibility to record a sequence of commands and build a kernel from the combined command sequence which then can be run with much less total overhead. This may accelerate the parallel program version significantly although the approach works only when there are no data dependent conditions. One therefore needs a smart way of implementation or a split of a command sequence into different kernels with the if-condition in between.

Last but not least, the program will be used for accuracy studies of the used methods and studies of systematic errors for muon g-2 and EDM experiments. Some simple initial tests show for example that the accuracy of the code depends on many details. In one test a solution for the simple initial value problem

$$dy/dx = 1.0 \cdot y, \quad y(0) = 1, \quad x \in [0, 10] \qquad (3)$$

with the analytical solution $y(x) = e^x$ was calculated with the Boost implementation of the 4th order Runge-Kutta algorithm with different optimization levels of the gcc compiler. The calculation was performed for different step sizes $dx = 10^{-k}$, $k = 1...10$ with *double* as the chosen data type. No difference of the solutions for optimization levels *O0* to *O3* could be observed but the option *Ofast* produces different results than all others, see Fig. 2. This may generally be not that surprising since the option *-ffast-math* is switched on but since the Runge-Kutta algorithm requires only the basic arithmetic operations $(+, -, *, /)$ for this problem, it actually is a surprise. For more complicated equations with trigonometric functions etc. significantly worse results may be expected. In this respect, a possible future symbolic math capability of compilers may prove very useful. Expression trees built with the use of expression templates could be simplified and optimized which would not only reduce execution time but could at the same time increase accuracy.

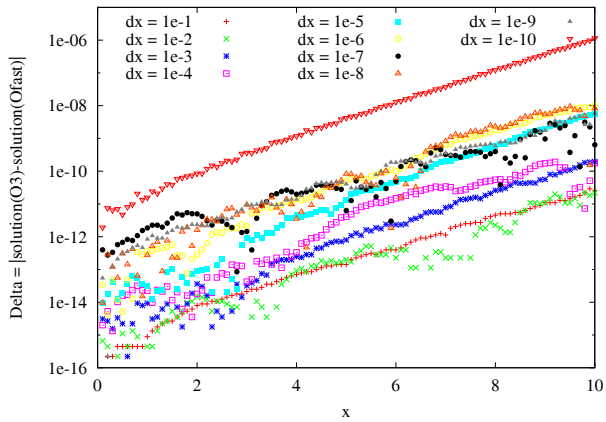

Figure 2: Difference of the solution of eq. 3 obtained with the 4th order Runge-Kutta algorithm from the Boost library when the code is compiled either with the *-O3* option or the *-Ofast* option.

REFERENCES

[1] V. Anastassopoulos et al., arXiv preprint, arXiv:1502.04317 (2015).

[2] VexCL website: `https://github.com/ddemidov/vexcl`

[3] Boost website: `www.boost.org`

[4] D. Demidov et al., SIAM Journal on Scientific Computing **35.5**, C453 (2013).

[5] A. Nordsieck, Mathematics of Computation **6.77**, 22 (1962).

[6] S. Hacıömeroğlu, Y.K. Semertzidis, Nucl. Instrum. Methods Phys. Res., Sect. A **743**, 96 (2014).

[7] E.B. Kuznetsov, Journal of the Franklin Institute, **344**, 658 (2007).

[8] E.M. Metodiev et al., Nucl. Instrum. Methods Phys. Res., Sect. A **797**, 311 (2015).

[9] F.J.N. Farley, Physics Letters B **42.1**, 66 (1972).

[10] W.M. Morse et al., Phys. Rev. ST Accel. Beams **16**, 114001 (2013).

[11] S. Blanes et al., Physics Reports **470**, 151 (2009).

[12] G.Yu. Kulikov, R. Weiner, J. Comp. App. Math. **233** 2351 (2010).

[13] G.Yu. Kulikov, Russ. J. Numer. Anal. Math. Modelling, **28.4**, 321 (2013).

[14] G.u. Kulikov, IMA J. Numer. Anal., **33**, 136 (2013).

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-136-6