Modeling Relativistic Muons in Electromagnetic Storage Rings via Object Oriented Techniques

Tomasz Malisiewicz
ERULF Program
Rensselaer Polytechnic Institute
Brookhaven National Laboratory
Upton, New York

August 2, 2002

Prepared in partial fulfillment of the requirements of the Department of Energy Research Undergraduate Laboratory Fellowship under the direction of Yannis Semertzidis in the Physics Department at Brookhaven National Laboratory.

Contents

Abstract

The Muon g-2 Experiment at Brookhaven National Laboratory consists of the world's largest super-conducting magnet which is used as a muon storage ring. Since it is impossible to control the momentum and location of each muon at the time of injection into the ring, the configuration of magnetic and electric fields has to be determined in such a way as to allow for maximum orbital stability throughout the experiment. By adhering to the tenets of object oriented programming, a highly flexible program was developed in C++ in order to study the effects of different electromagnetic ring configurations and muon initial conditions. The use of abstract base classes, inheritance, and polymorphism allowed the easy incorporation and testing of several different algorithms for solving the necessary differential equations. The program's results demonstrated the dependence of Coherent Betatron Oscillation amplitude on the muon initial conditions and a dependence on CBO frequency on weak focusing field intensity. A study showed first order methods for solving differential equations inadequate for the simulation and the Runge-Kutta fourth order method as suitable for studying particle dynamics. The program showed OOP to be an effective mechanism for analyzing different numerical algorithms and simulating particle dynamics.

Research Category: Computer Science/Physics
School Author Attends: Rensselaer Polytechnic Institute
DOE National Laboratory Attended: Brookhaven National Laboratory
Mentor's Name: Yannis Semertzidis
Phone: 3881
E-mail Address: Yannis@bnl.gov

Presenter's Name: Tomasz Malisiewicz
Mailing Address: 260 Bay Avenue
City/State/ZIP: Patchogue, NY 11772
Phone: (631) 758-7197
E-mail Address: malist@rpi.edu
Is this being submitted for publication? Yes

DOE Program: ERULF

1 Introduction/Problem Description

Large storage rings have been playing an ever increasing role in the world of particle and high energy physics, however the complex geometries of the electric and magnetic fields do not always allow the scientists to understand the particle dynamics 'a priori.' Without the possibility of obtaining analytic solutions to the equations of motion for high energy particles, the scientists has to rely on numerical simulations. Since there are many different algorithms in the field of numerical computing and proper software engineering practice yields reusable code, it is worthwhile to utilize object oriented programming(OOP) in order to test the performance of various algorithms and attain the highest degree of flexibility.

An effective way to help many scientists who are interested in numerical solutions is to engineer highly-flexible computer software which simulates the dynamics of relativistic particles. An object oriented framework allows the greatest degree of flexibility by allowing experimentalists to easily fine tune experimental apparatus while running 'virtual experiments,' and still provide theoreticians with a robust environment for theory validation by altering the computational foundations. OOP, which is the backbone of proper software engineering practice, facilitates the development of customizable software which is ideal for the modeling of physical systems. Because computer simulations produce a great deal of data which can be further analyzed, it is important to test the effectiveness of the simulation by modeling the conditions of an already existing experiment and comparing the computational(virtual experiment) results with the empirical(real experiment) results. Since data collection via experiments is the main tool of science for validating theories, computer simulations serve a critical role in the development of modern science. "The classical scientific method depends upon theory formation followed by experimentation and observation in order to provide a feedback loop to validate, modify and improve the theory." (Zelkowitz, 1997) Clearly, by part-taking in this feedback loop, computer science has the potential to rise as a bold advance in the classical scientific method. Even though computer science can serve as an all powerful tool for any science, the physics community has the most to gain by adherence to a proper software engineering practice due to the rising complexity in physical theories.

1.1 Goals of paper and overview of contents

The long term goal of this project was to develop an object oriented framework in C++ which simulated the Muon g-2 Storage Ring at Brookhaven National Laboratory. Shorter goals included developing various algorithms for solving the necessary differential equations and testing their respective efficiencies. In the first section, the physics behind the Muon g-2 experiment, the importance of such an experiment, and the physics behind the measurement process are described. In the Methods and Materials section, an overview of the methodology of software engineering is described, how it varies from traditional programming techniques, and the overall program design. Next, the dynamics of relativistic particles are introduced and how an OO framework in C++ was used to solve the differential equations. Finally the paper concludes with a short analysis of the various numerical algorithms used for the particle simulation program, their ability to model actual g-2 data, and the successes and failures of applying object oriented techniques to real world problems in physics.

1.2 The g-2 value

In the Muon g-2 experiment at Brookhaven National Laboratory, scientists are trying to measure the muon anomalous moment with unprecedented accuracy. "The goal of the ongoing experiment at Brookhaven National Laboratory is to improve the accuracy of the CERN measurement by a factor of 20, or to .35 ppm." (Hughes, 2002) This measurement is very important for physics because the muon anomalous moment can be determined to a high degree of precision using both theoretical calculations and experimental methods. Since it is possible to ascertain both the experimental and theoretical values, the g-2 value serves as a good mechanism for testing the Standard Model. "The magnetic moments and g-values of particles have played a central role in the development of modern physics, including quantum electrodynamics, nuclear physics, and particle physics." (Hughes, 2002) The result from the 2000 data analysis, published in July 2002, was $a_{\mu+} = 11659204(7)(5) \times 10^{-10}$ (0.7 ppm), which stands in excellent agreement with the 1999 result, but more experimental runs are needed to reduce the error to the desired accuracy, 0.35 ppm. (Bennet, et. al. 2002)

For a particle, the magnetic moment $\vec{\mu}$ is

$$\vec{\mu} = \frac{e\hbar}{2mc} g\vec{s} \tag{1}$$

where $\vec{s}$ is the spin angular momentum, and g is the gyro-magnetic ratio. If g does not equal 2, then the anomaly is defined as

$$a = \frac{g-2}{2} \tag{2}$$

It is this anomaly of the muon which is measured in the g-2 experiment. According to Vernon Hughes, a deviation of a_μ(expt) from a_μ(SM) indicates new physics, such as lepton structure, W anomalous magnetic moment, supersymmetry, leptoquarks, new particles, or extra dimensions. (Hughes, 2002) Evidently, it is important to determine whether the experimental value agrees with theory, and in order to do this efficiently, unprecedented accuracy must be attained. From the 2000 data analysis, the difference between $a_\mu(SM)$ and $a_\mu(expt)$= 1.6 to 2.6 times the combined theoretical and experimental uncertainty. (Bennet, et. al. 2002)

1.3 The Experimental Setup

The Alternating Gradient Synchtron, AGS, is the source of protons which ultimately leads to the production of muons. After the protons are accelerated in the AGS, they collide with a target and produce pions, weighing approximately $\frac{1}{6}$ of a proton. The method of separating muons from pions is accomplished via a pion 'decay channel' with focusing elements. Since pions decay along the momentum, a bending magnet is used to select the forward decay muons, namely the muons with momentum only 1-2 % less the pions. (Farley, 1990) After this selection process takes place, this beam of longitudinally polarized muons is injected into the Storage Ring via an inflector magnet. This super-conducting inflector substantially cancels the 1.45 T magnetic field produced by the storage ring magnet which allows the muon beam to enter the storage ring approximately parallel to the central orbit but 77mm farther out in radius. (Hughes, 2002)

The Storage Ring consists of the world's largest super-conducting magnetic, with a radius of 7.112 meters, an aperture diameter of 9cm, and a mean magnetic field of 1.45 Tesla. For this experiment the the magnetic field is extremely homogeneous and an electric quadrupole field is provided in order to provide weak focusing. (Hughes, 2002) The electric quadrupoles are responsible for keeping the particle in orbit and for reducing the amount of scraping present during the muon injection. (Hughes, 2002) During its orbit in the storage ring, each muon executes cyclotron motion with angular frequency ω_c in the horizontal plane and Coherent Betatron Oscillations(CBO) are produced by the focusing field in both the horizontal and vertical planes. Concurrently, the muon spin precesses with angular frequency ω_s. The difference $\omega_a = \omega_s - \omega_c$, together with knowledge of the mean magnetic field, allows physicists to determine the muon anomaly, a_μ. (Hughes, 2002)

$$\vec{\omega_a} = \frac{e}{mc}(a_\mu \vec{B} - [a_\mu - \frac{1}{\gamma^2 - 1}]\vec{\beta} \times \vec{E}) \tag{3}$$

The key to the success of the g-2 experiments is the selection of the initial muon momentum. γ is chosen in order to cancel out the effects of the $\vec{E}$ field on ω_a, namely at the 'magic' γ=29.3. This 'magic' gamma is equal to approximately 99.94 % the speed of light.

1.4 The Measurement process

In order to determine a_μ, scientist use the muon's spontaneous decay to their advantage. After 2.2 μs in the muon rest frame (64.4 μs in the storage ring rest frame), each muon decays according to Equation (4).

$$\mu^+ \rightarrow e^+, \nu_e, \nu_\mu \tag{4}$$

"Parity violation leads to a preference for the highest-energy decay electrons to be emitted in the direction of the muon spin." (Hertzog, 4) The storage ring is lined with 24 photomultiplier detectors (lead/scintillating fiber calorimeters), which measure the time and the energy of the decay product. The muon spin direction is determined when looking at the number N(t) of decay positrons with energies above a selected threshold. In order to measure the magnetic field, $\vec{B}$, nuclear magnetic resonance (NMR) is used with a standard spherical probe of H_2O. 17 NMR probes were mounted on a trolley and traveled through the ring every 2 or 3 days, while approximately 150 probes were fixed on the bottom and top walls of the vacuum chamber. The total systematic error on ω_p was reduced by .24ppm in the 2000 run. (Bennet, et. al. 2002)

2 Materials and Methods

2.1 Particle Dynamics

The equation for motion in the presence of Electric($\vec{E}$) and Magnetic($\vec{B}$) fields:

$$\frac{d\vec{\beta}}{dt} = \frac{e}{\gamma mc}[\vec{E} + (\vec{v} \times \vec{B}) - \vec{\beta}(\vec{\beta} \cdot \vec{E})] \tag{5}$$

The equation for the spin in the presence of Electric($\vec{E}$) and Magnetic($\vec{B}$) fields:

$$\frac{d\vec{s}}{dt} = \frac{e}{m}\vec{s} \times [(a + \frac{1}{\gamma})\vec{B} - \frac{a\gamma}{\gamma + 1}(\vec{\beta} \cdot \vec{B})\vec{\beta} - (a + \frac{1}{\gamma + 1})(\vec{\beta} \times \vec{E}] \tag{6}$$

These vector differential equations must be solved numerically in order to produce an effective particle simulation.

2.2 Program Design

OOP is more than a style of programming, it is highly abstract level of thinking which is crucial to the development of successful software according to the software engineering paradigm. Object oriented languages were designed with the software development process in mind. "Overall, object-oriented software development confronts corrections, modifications, and improvements as fundamental parts of the programming process, essential to producing quality software. The basic idea of object-oriented programming is to decompose a software system on the basis of objects - entities characterized by actions - instead of on the basis of functions or data, as is done in more traditional design methodologies." (Barton, 1994) Instead of thinking about a program in terms of computations, OOP allows programmers to think in much more familiar concepts, namely objects. Since one key goal of OOP is to aid the evolution of a program through its development stage, it serves as the mechanism responsible for promoting software development into an engineering discipline. According to Barton, the use of objects in OOP allows programs to be engineered in the same way that airplanes, houses, and computers are designed. (Barton, 1994)

What is an object? "An entity with a well-defined boundary and identity that encapsulates state and behavior. State is represented by attributes and relationships, behavior is represented by operations, methods, and state machines. An object is an instance of a class." (Winter, 1998)

As for the the programming language, C++ was chosen due to its widespread use. The reasons why C++ has dominated the programming community as an OO language are very clear. C++ is widely available because of its wide popularity and portability, and C++ catches many errors during the early stages of the development process, thus increasing productivity. C++ can call C functions and FORTRAN subroutines, it can interface with graphics packages such as OpenGL, and it comes standard with many operating systems. One interesting quality of C++ is that due to its implementation as a superset of C, is that unlike Java it allows the easily integration of systems programming into its OO applications. According to Barton, "If C++ is not the perfect language, it is far ahead of the FORTRAN, C, and PASCAL family." When software lifetime is considered, it is difficult to not choose the mainstream since much of the world-wide development in OO currently is based on C++ and tools of all kinds are mostly only available for that language. (Bos, 1998)

When this approach is applied to the muon g-2 experiment, classes such as Particle, Muon, and StorageRing are developed as no surprise since they already resemble their physical counterparts. However, the challenge comes from objectifying the internal mechanism responsible for solving the necessary differential equations. Not only does this approach allow easy testing/debugging, it is necessary for the project to make successful transitions between

programmers. Also, by encapsulating the differential equation solver, more advanced algorithms can later be integrated into the software without any difficulty.

2.3 Functional Objects

Motivated by OOP was the encapsulation of functions since C++ does not provide the programmer with a built-in way of treating functions like objects. "C++ built-in functions are not C++ objects: They cannot be copied, assigned, or altered. C++ does provide pointers to functions, and these pointers can be used to build objects." (Barton, 1994) According to SGI whose Standard Template Library uses Functional Objects , "A Function Object, or Functor, is simply any object that can be called as if it is a function." (SGI) Clearly Functors are an advancement over regular functions, since by becoming objects, they are now allow to part-take in advanced features like inheritance and polymophism.

```
template<class T, class G>
class Function
{
 public:
  virtual T operator()(T,G)=0;
};
template <class T, class G>
class GenericFunction:
public Function<T,G>
{
 protected:
  T (*func)(T,G);
 public:
  GenericFunction(T (*f)(T,G));
  virtual T operator()(T y, G t);
};
```

The Function interface describes how all Functional Objects should behave, namely they should have a two parameter functional evaluator called operator(). It is this abstract base class that is used to construct more advanced Functors. GenericFunction allows already existing functions to treated as Function objects, thus allowing the most general algorithms to work on Functions. The Function class allows an easy representation of Equation (5) and (6), by allowing hidden parameters such as $\vec{E}$ and $\vec{B}$ to reside as private members.

```
class DBDT: public Function<vec,double>
{
 private:
  Particle* my_particle;
  StorageRing* my_ring;

 public:
  DBDT(Particle* part, StorageRing* str): my_particle(part), my_ring(str) {}
  virtual vec operator()(vec beta,double t)
```

```
  {
       return (my_particle->getCharge()) * E_CHARGE /
       (my_particle->getMass() * C_LIGHT * gamma(beta)) * (
       my_ring->E(my_particle->getDisplacement()) +
       (C_LIGHT*beta) / my_ring->B(my_particle->getDisplacement()) -
       beta * (beta * my_ring->E(my_particle->getDisplacement())));
  }
};
```

2.4 Integrator Interface

The added flexibility created with the introduction of Functional Objects was necessary in order to develop a generalized differential equation solving class. Each doubly templated Integrator class encapsulates a pointer to a doubly templated Functional Object, a pointer to the independent variable, and a pointer to the dependent variable. Also, the Integrator class specifies two purely virtual functions, step(), and getName() which represent the requirements of every class derived from Integrator. Since Integrator is an abstract base class, new classes(Euler,RungeKutta,Heun,etc.) were designed that work according to Integrator's specifications and itegrated into the program via the OO technique known as polymorphism.

```
template <class T, class G>
class Integrator
{
 protected:
  Function<T,G>* g;
  T* y;
  G* x;

 public:
  Integrator(Function<T,G>* func=0,T* v=0,number* t=0);
  virtual ~Integrator();

  virtual void step()=0;
  virtual string getName()=0;
};
```

The Integrator Class allows for the numerical integration of any two parameter functions, such as Equation (6), and (5). Although vector valued differential equation of motion and spin were used throughout the program, the generality of the Integrator class allows for the solutions of scalar valued functions, vector valued functions, complex valued functions, etc.

2.5 The Numerical Algorithms

Due to the careful implementation of the Integrator class, it was very easy to develop new algorithms to integrate first order differential equations. Several algorithms were used from

little complexity (first order single step explicit) to more advanced (multi step predictor corrector). The program was designed so that it would implement any algorithm derived from the Integrator class, without worrying on the algorithm implementation. There were 4 main algorithms being tested, even though many slight variations were developed along the way.

- Forward Euler Method (First-Order Explicit)

$$x(t+h) = x(t) + hf(x(t), t) \tag{7}$$

  1. Very easy to implement and very fast (one functional evaluation per step)
  2. Unfortunately only accurate to first order

- Heun's Method (Second-Order Explicit)

$$\begin{aligned} x^* &= x(t) + \frac{h}{2} f(x(t), t) \\ x(t+h) &= x(t) + hf(x^*, t + \frac{h}{2}) \end{aligned} \tag{8}$$

  1. Only two functional evaluations per step
  2. Much more accurate than Euler First Order Forward Method

- Fourth Order Runge-Kutta Method (Fourth-Order Explicit)

$$\begin{aligned} F_1 = f(x,t), F_2 = hf(x + \frac{F_1}{2}, t + \frac{h}{2}), \\ F_3 = hf(x + \frac{F_2}{2}, t + \frac{h}{2}), F_4 = hf(x + F_3, t + h) \\ x(t+h) = x(t) + \frac{1}{6}(F_1 + 2F_2 + 2F_3 + F_4) \end{aligned} \tag{9}$$

  1. Very popular
  2. Four functional evaluations per step

- Adams-Bashforth-Moulton Predictor Corrector (Four-step Predictor and Three-step Corrector)

$$f_n = f(x(t+nh), t+nh)$$

$$Predictor:$$

$$x(t+h) = x(t) + \frac{h}{24}(55f_0 - 59f_{-1} + 37f_{-2} - 9f_{-3})) \tag{10}$$

$$Corrector:$$

$$x(t+h) = x(t) + \frac{h}{24}(9f_1 + 19_0 - 5f_{-1} + f_{-2}) \tag{11}$$

  1. Two functional evaluations per step
  2. Not self-starting due to the use of previous points. Until enough previous solutions are determined, another method (ie. Runge-Kutta) method must be used.

2.6 Documentation

Any project is incomplete without proper documentation. Documentation is not only very important for the programmer during the development stage, but it is critical for extending the lifetime of a software project. A poorly documented software project will not allow for an easy transition from one developer to another, thus drastically limiting its lifetime. Documentation was easily created via the help of project Doxygen written by Dimitri van Heesch. By following one of a few supported commenting styles (Javadoc style), Doxygen scanned every single source file and created a beautiful set of html web pages describing every detail of the program. The online documentation and the simulation program can be both found at

`http://www.rpi.edu/~malist/bnl/`

3 Results

3.1 Coherent Betatron Oscillation

Coherent Betatron Oscillations (CBO) were observed both in the XY (horizontal) and YZ (vertical) planes. Just as expected, CBO amplitudes were minimized when injection was tangential to the central orbit of the storage ring ($\vec{L}$=Location. $\vec{L}_{injection} =< 7.112, 0, 0 >$ and $\vec{\beta}_{injection} = \vec{\beta}_{magic}$). However, since this ideal situation would seldom occur in the experimental run, various different initial phase space conditions were tested. First, $\vec{\beta}_{injection} = \vec{\beta}_{magic}$ was kept constant and injection location was allowed to vary. These results showed that vertical CBO amplitude was proportional to L_z and horizontal CBO amplitude was proportial to L_x, keeping in mind that the ideal $\beta_{injection}$ lies along the Y axis. Second, the injection location was kept constant and the injection angle in phase space was varied. First, the angle the between the Y axis and $\vec{\beta}$ in the YZ plane was varied. As the angle increased, the amplitude of the vertical CBO also increased linearly until the maximum angle of approximately .0023 radians, which produced vertical oscillations with an amplitude of 4.5cm (the aperture radius). Second, keeping β_z=0, the angle in the XY plane between $\vec{\beta}$ and the Y axis was varied. This angle was found to be directly proportional to the amplitude of the horizontal CBO, with a maximum amplitude of 4.5 cm at .006 radians. Actually, since the accepted angles were so small and $\sin(\theta) \approx \theta$ when $\theta \approx 0$, the current study was unable to discern from a linear relationship between θ or a linear relationship between $\sin(\theta)$. The phase space initial condition tests concluded that frequency of the CBO effect was independent of the muon initial conditions, and that CBO amplitude is sensitive to injection locations and angles in phase space. The next test involved the weak focusing field. Since the electric quadrupoles are necessary for vertical focusing, two slightly different quadrupole configurations were tested. In the first case, the electric quadrupoles were placed at their respective locations as in the g-2 ring, and in the second case there was assumed a electric field with an intensity 43% of each single quadrupole along the entire storage ring. With the electric quadrupoles placed in their respective locations, the CBO graph showed a some 'non-smooth' linearity near the oscillation peak. In the case of the electric field covering the

entire storage ring region, the betatron oscillations showed more curvature near the peak, but this slight anomaly proved both scenarios to be effective in modeling the CBO effect.

3.2 OOP

The OO design of the program allowed easy testing of separate objects before it would be nearly impossible to find bugs. This flexibility allowed the Integrator classes to be tested independently from the particle simulation, thus decreasing development time. Not only was development time shortened by adherence to an OO methodology, but the design allows for future re-usability of many objects developed for this project. The Integrator classes can be incorporated into any project which requires the solution of a differential equation, and Particle classes will by handy for any type of particle system simulation.

3.3 Algorithmic Performance

The Euler first order method proved to be inadequate in most cases. The step-size required for the Euler algorithm to succeed in tracking the particle for the first microsecond was very low, thus taking an exorbitant amount of time. However, there was a surprise in the effectiveness of the Heun second order method. When $h \approx 1e-11$ Heun's algorithm started behaving very much like Runge-Kutta second order algorithm. Surprisingly, the constant step size Adams-Moulton Predictor Corrector algorithm failed to model the CBO.

4 Discussion and Conclusion

Runge-Kutta proved to be a very effective algorithm with Heun's algorithm in second and both Euler and the current version of the Adams-Moulton Predictor Corrector were insufficient for particle simulations. While Runge-Kutta and Heun performed very well at all levels, Heun's algorithm starting showing inconsistency only at the $h > 1e-11$ level. Since the current version of the Adams-Moulton Predictor Corrector does not incrporate any local error approximations nor a variable step size, these modifications will drastically boost the algorithm's performance. However, these improvements to the Adams-Moulton algorithm will als increase run time, and since Adams-Moulton took the longest time to run already, future tests will have to determine the effectiveness of this algorithm. In conclusion, the simulation program proved to be an effective mechanism for studing the CBO effect whose significance was under-estimated in the 1999 data analysis. Since the oscillations change the average radial position of the muon, this effects also modulates the recorded positron time spectra. Clearly, an effect on the positron time spectra affects the measurement of a_{mu}. According to the systematic error table of the 2000 data analysis, CBO related systematic errors account for a large portion of the error, thus showing the importance of understanding CBO effects in general. The systematic error for CBO increased by a factor of 4 from the 1999 data analysis to the 2000 data analysis.(Bennet, G. et.al. 2002)

4.1 Other scientific results

Since the development of OOP, the scientific community has gained a large edge on the development of scientific software. "Among all, the software development process based on the object oriented (OO) approach is considered to be a most promising technology these days." (Amako, 1998) There are many reasons why OO software engineering has been so successful. Many long term experiments have a fast turnover of collaborators, meaning that the software development team can change abruptly. Traditional programming methodology is insufficient in providing a mechanism which can handle this abrupt change, thus proving old methods as inadequate for the physics community. On behalf of the Moose project at CERN, Bos states that "the object oriented approach is well suited for the development of software for the LHC experiments." (Bos, 1998) Also, according to Amako, "Introduction of engineering discipline is absolute necessity in the construction of a software system in large scale future HEP experiments." (Amako, 1998)

4.2 So how good is C++?

Even though C++ will still be a popular choice of language for years to come, it is not a perfect language because of its intimate connection with C. Since C++ was developed as a superset of C, its OOP capabilities are a mere 'add-on,' allowing C++ developers to mix modern OO concepts and traditional C-style programming. Languages such as Java which were developed on an OO foundation are gaining popularity due to their inability to leave the OO methodology. "We have started looking at Java and were pleased to see that it is a much cleaner language than C++ because it was designed as an OO language from the start which C++ was not." (Bos, 2) In conclusion, the most efficient programming methodology is OOP and as for the methods and languages it is very important to choose something mainstream.

4.3 Future goals

As for the muon g-2 experiments at Brookhaven National Laboratory, the analysis of the 2001 data is underway. In 2001, μ^- were used and 3×10^9 decay electrons were observed. This measurement will improve the current experimental value of a_μ, as well as provide a sensitive test of CPT violation. (Bennet, G. et.al. 2002) Even after this analysis, more experimental runs will be needed to improve the accuracy of a_μ to the desired accuracy. As for future uses of the muon g-2 storage ring, scientists at BNL are proposing to search for the muon Electric Dipole Moment, which will test models boyond standard theory. As for modeling particles, object oriented software engineering still has lots of room for future growth. As mentioned in the results section, the Integrator class serves as a solid foundation for numerical solutions of differential equations, thus extending the class's scope beyond high energy particle simulations. Modifications of the existing program can yield more generalized particle accelerator development software, and the use of 3D programming APIs such as OpenGL can provide extensions for 3D visualization. Also, object oriented GUI development classes such as those provided by Trolltech (http://www.trolltech.com) can provide the program with invaluable extensions. Trolltech is responsible for the creation of Qt, the cross-platform C++ GUI

toolkit, which allows the easy development of graphical components, much like Swing in Java. In conclusion, an OO approach to software development in C++ has many distinct advantages. When applied to the development of software for the physics community, OOP is an invaluable resource whose use will only increase in the future.

5 Literature Cited

Amako, Katsuya. (1998) The software development process in worldwide collaborations. Nuclear Physics B 61B: 669-673.

Barton J., Nackman L.(1994). Scientific and Engineering C++: An Introduction with Advanced Techniques and Examples. Addison-Wesley Publishing, Yorktown Heights, New York.

Bennet, G. et.al. (2002) Measurement of the Positive Muon Anomalous Magnetic Moment to 0.7 ppm.

Bos, Kors. (1998). The Moose Project. Computer Physics Communications 10: 160-163.

Hertzog, David W. The BNL Muon Anomalous Magnetic Moment Measurement.

Hughes Vernon W., Sichternamm Ernst P. (2002). The Anomalous Magnetic Moment of the Muon. International School of Subnuclear Physics. August29-

Farley F.J.M., Picasso E. (1990). The Muon g-2 Experiments. Quantum Electrodynamics. World Scientific Publishing Co, New Jersey.

Winter, Mario. (1998). Managing Object-Oriented Integration and Regression Testing. euroSTAR98. Munich, Germany.

Trifomov, A.V. (2002). A New Precision Measurement of the Positive Anomalous Magnetic Moment of the Positive Muon. Diploma, Moscow State University. 2002.

Zelkowitz, et al. (1997). Experimental validation in software engineering. Information and Software Technology 39. 1997: 735-743.

6 Acknowledgments

I would like to thank the members of the scientific community that have helped me further develop my interests in computer science and physics. Their time and attention has helped pave the way on my intellectual journey through science. Most of all I would like to thank Dr. Yannis Semertzidis, my mentor. I would also like each member of the Muon g-2 collaboration. At last, I would like to thank the Department of Energy and Brookhaven National Laboratory for providing me with this remarkable opportunity to work with some of the world's brightest minds.

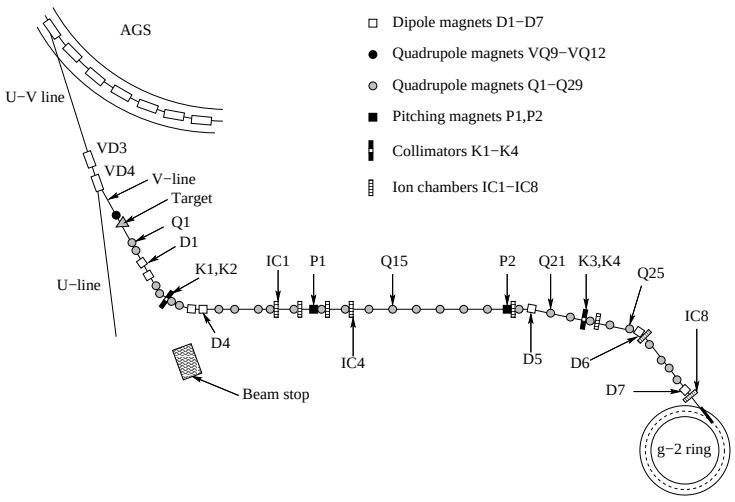


Figure 1: The beamline: The path of the muon from the AGS into the g-2 storage ring. (Trifomov, A.V. 2002)

7 Figures

8 Biography of author

Tomasz Malisiewicz was an intern at Brookhaven National Laboratory through the Department of Energy's ERULF program during the summer of 2002. He is currently a computer science and physics dual major at Rensselaer Polytechnic Institute. At BNL he analyzed various numerical algorithms for modeling relativistic particles and object oriented software design. His future plans include attending graduate school in theoretical physics and/or computer science in order study computational physics.

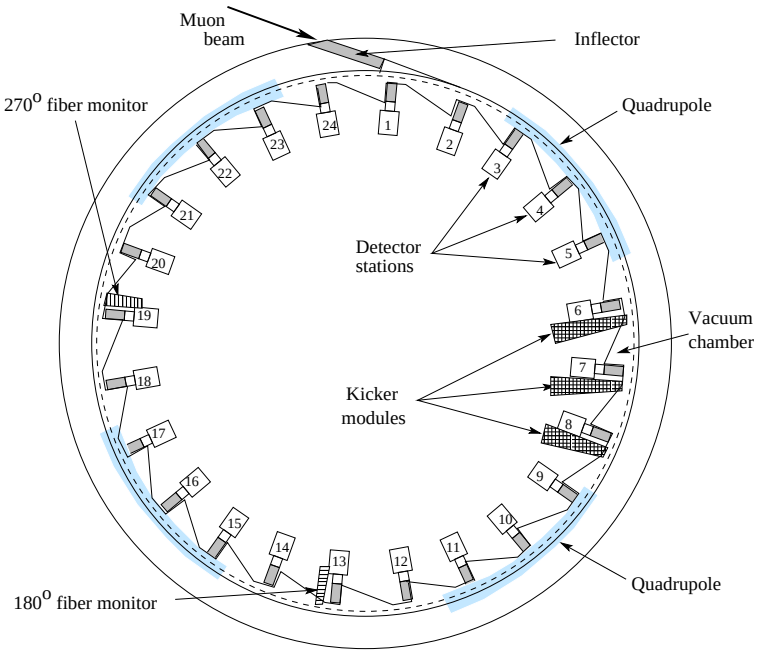


Figure 2: The Muon g-2 Storage Ring. (Trifomov, A.V. 2002)

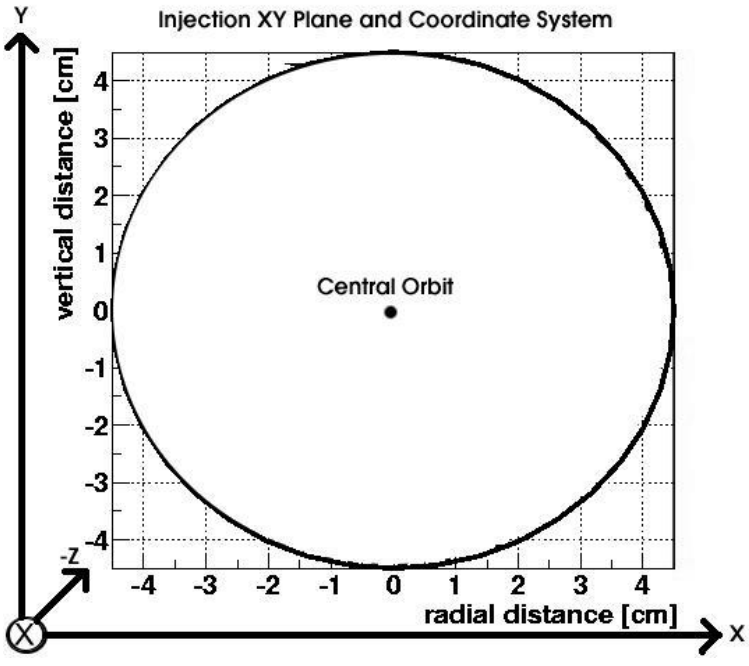


Figure 3: A slice in the XY plane of the coordinate system used showing the aperture with a radius of 4.5 cm, and the central orbit.

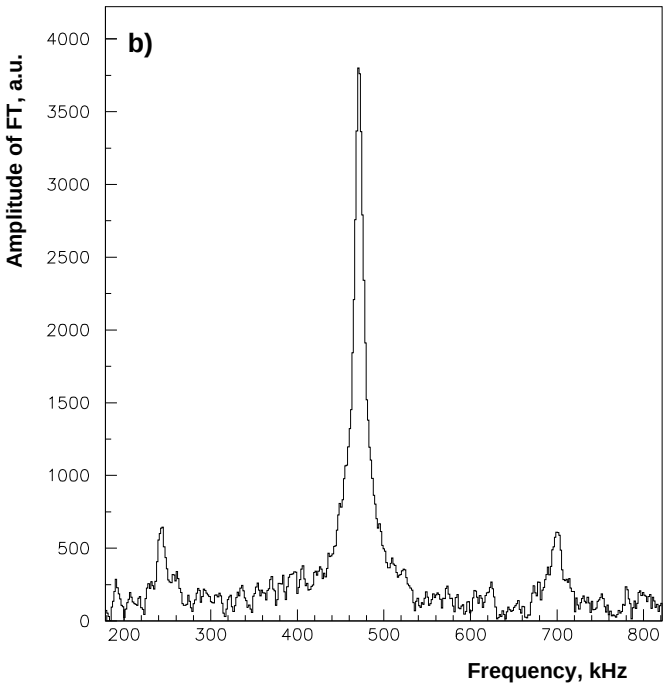


Figure 4: Fourier analyzed residuals from the 5-parameter fit showing a large peak at the CBO frequency. This demonstrates the degree of influence of the CBO effect on the data analysis. (Trifimov, A.V. 2002)

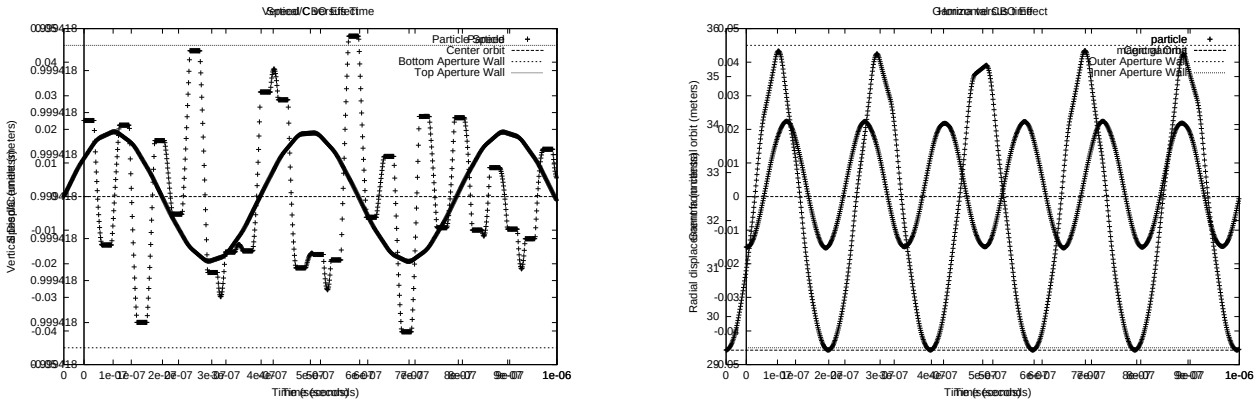


Figure 5: CBO with initial displacement $= < 7.112, 0, 0 >$ and initial $\beta =< 0, -.999417547271, .001 >$.

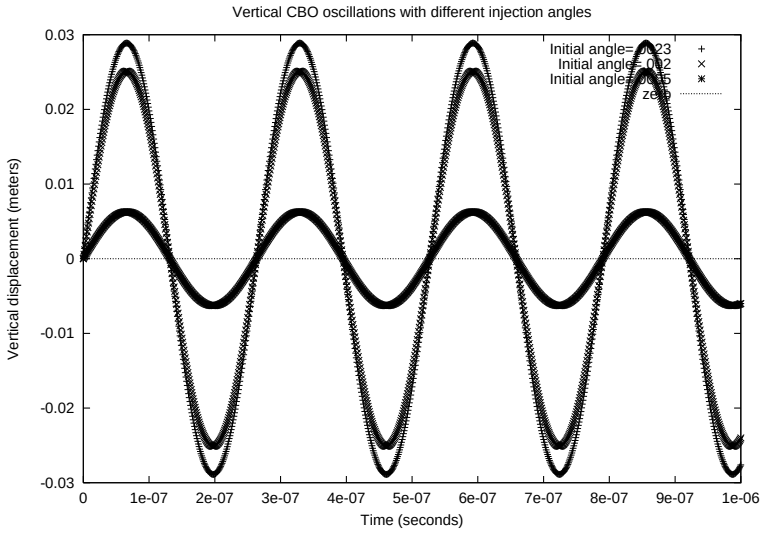
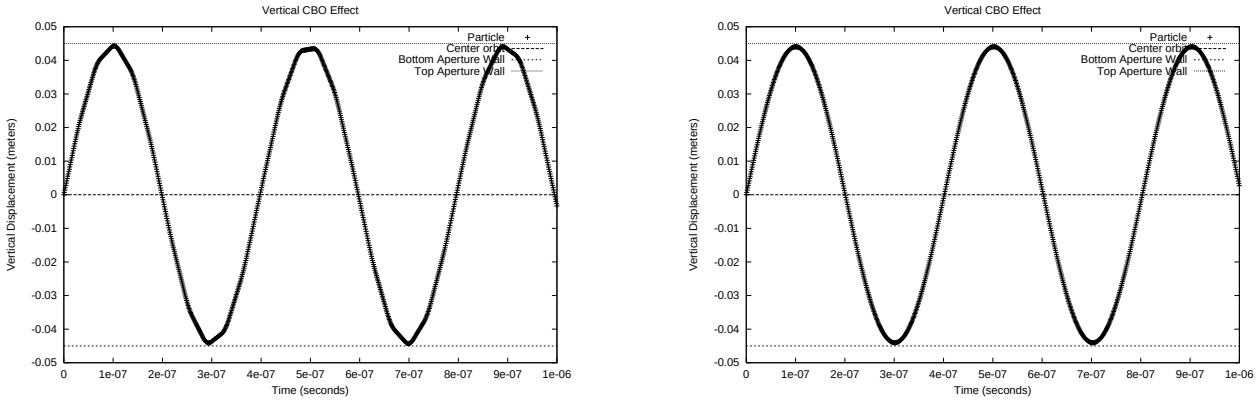


Figure 6: Clearly the frequency remains the same, while the amplitude changes as a function of injection angle. The injection angle is the angle between the vertical and XY plane velocity when $\beta = \beta_{magic}$



(a) (b)

Figure 7: Vertical CBO with (a) non-smooth quads, and (b) smooth quads. The non-smooth quads show a triangular shaped peak although the shape of the oscillations is very similar.

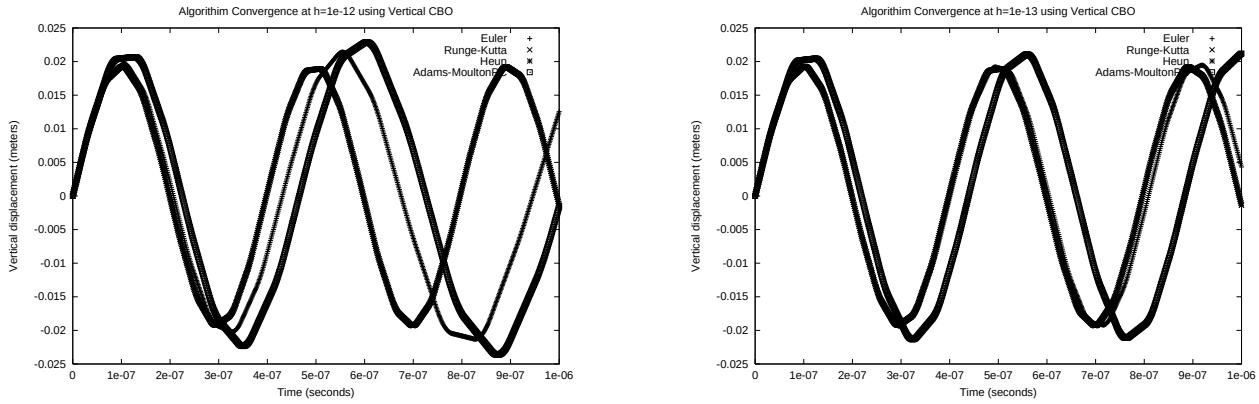


Figure 8: The divergence of the Euler and Adams-Moulton algorithms at h=1e-12 and h=1e-13 level. The two diverging algorithms do better as step decreases, but not good enough. RungeKutta and Heun both converge, hence they occupy the same area on the graph.

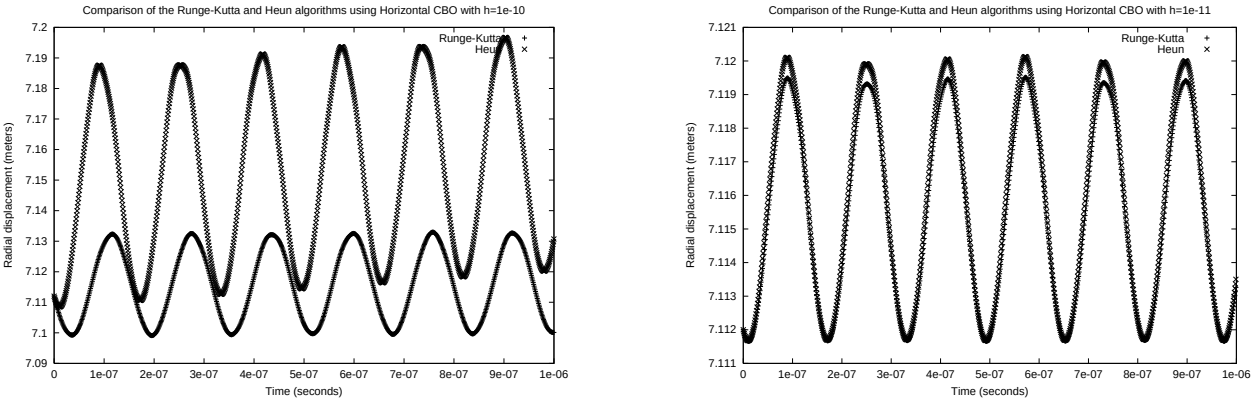


Figure 9: Comparison of Runge-Kutta and Heun algorithm at h=1e-10 and at h=1e-11. At this level, Euler and Adams-Moulton diverge, but Heun starts behing well at around h=1e-11.

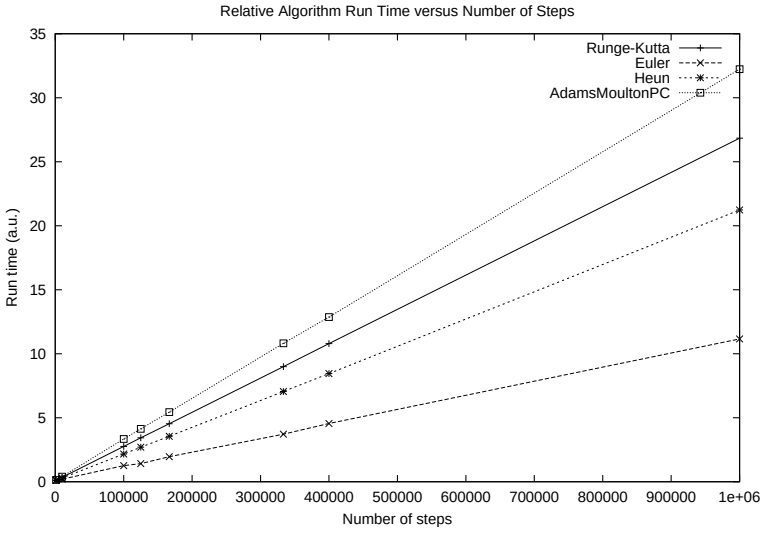


Figure 10: The Euler method is clearly the fastest, and the Adams-Moulton PC as the slowest. Note how the second order Heun's method and the fourth order Runge-Kutta method are very similar in run times.