ATLAS: FASTER MATRIX ALGEBRA

David Nicolas Lopez

Introduction:

The experiment ATLAS has achieved unprecedented milestones in the quest to understand the makeup of our universe. By observing high energy collisions inside the Large Hadron Collider, the Higgs Boson and other fundamental particles have been discovered. ATLAS computes a plethora of data while looking for patterns through ingenious algorithms with the help of Eigen.

Eigen is an open source linear algebra library that offers a dynamic and reliable platform for many state of the art projects. However, ATLAS relies on 5x5 covariance symmetrical matrices to describe the position of tracks in magnetic fields, and in larger symmetric matrices for algorithms in pattern recognition. The Eigen Library does not yet support symmetric matrices, and if it would, computation and storage in the ATLAS experiment would dramatically decrease.

Who am I?

I am an Illinois Wesleyan Physics and Computer Science double major with the dream of creating and understanding. I am in riveted by coding, and recently I won the Illinois Wesleyan Entrepreneurial Fellowship because of an app I developed. Presently, I am a Physics Teacher Assistant and an assistant at the Mark Evans Observatory and have taken object-oriented programming classes in C++ as part of my quest to further my involvement and understanding of physics and its applications. Experiments like ATLAS rivet me because they challenge our understanding of our universe and contributing to research is the path I have chosen for my career. I am completely ready to jump in and spend my whole summer coding with the ATLAS community.

Goals:

The Goal of this project is to implement robust and efficient symmetric matrix class to the Eigen library. Robustness and efficiency will be the focus when programming because it will guarantee reusability and will assure an effective integration into ATLAS' algorithms. Therefore, documenting the new class will prioritized to guarantee robustness, and creating methods for ATLAS' use for symmetric matrices will also be focused. The new class will be ready for immediate integration after the Google Summer of Code finishes. Additionally, I will develop repeatable test to check the runtime improvement before and after the integration of the new symmetric matrix class. All of this will allow ATLAS to perform more efficient and complex calculations to keep studying the fabric of our universe.

Deliverables:

- A robust and efficient Symmetric Matrix Class.
- Properly updated documentation
- Runtime tests to show computational improvement

If time allows:

- Interfacing and adding more methods to the class

Timeline:

All weekly tasks include developing unit tests and debugging the code.

Dates	Objectives
April 23 – May 14	-Community Bonding: -Contacting mentors and interacting with Main Eigen Developers -Perfecting Eigen style -Contributing to the Eigen opensource community
May 14 – May 19	-Creating Constructors and Destructors
May 20 – May 27	-Algebra Methods between Symmetrical Matrices: Addition/Subtraction
May 27 – June 3	-Algebra Methods between Symmetrical Matrices and Eigen:: Matrix: Addition/ Subtraction
June 3 - June 10	-Algebra Methods between Symmetrical Matrices: Multiplication **-**Running runtime tests -Integration testing
June 11	**FIRST EVALUATION**
June 10- June 17	-Algebra Method between Symmetric Matrices and Eigen::Matrix : Multiplication
June 17 – June 24	-Symmetric Matrix Methods: Interfacing with Eigen::Inverse
June 24- July 1	-Symmetric Matrix Methods: Decomposition
July 1 – July 8	-Review -Update Documentation -Integration testing -Running runtime tests
July 9	**SECOND EVALUATION**

July 8 – July 15	-Symmetric Matrix Methods: The Determinant
July 15 – July 22	-Symmetric Matrix Methods: interfacing with other methods (the symmetric LQ and QR methods, maybe?)
July 22 -July 29	-Final Review and Integration Testing -Preparing the class to be integrated
July 29 – August 5	-Final Documentation -Final Report on runtime improvement - Class Integration to Eigen
August 6	**FINAL EVALUTION**