Faster matrix algebra for ATLAS

(Project Proposal by STFC Rutherford Appleton Laboratory in 2018,http://hepsoftwarefoundation.org/gsoc/2018/proposal_ATLASEigen.html)

Introduction

The interactions due to collision of particle in the ATLAS detectors create an enormous flow of data. Complex data-acquisition and computing systems are then used to analyze the collision events recorded.

When the LHC is operating, 40 million packets of protons collide every second at the center of the ATLAS detector. Every time there is a collision, the ATLAS Trigger selects interesting collisions and writes them to disk for further analysis. The Event Data Model(EDM) which is intended to contain the detailed output of tracking the trajectories analyzes around **1PB** of Raw Data per year.

Background

To track these trajectory of charged particles and their collisions a large number of alignment parameters are required which can range from **10^4** to **10^5**. Several alignment algorithms as well as heavy CPU sonsumption is used to optimize these parameters. One such method requires the solution of a system of linear equations. To store these equations, a symmetric *n*-by-*n* matrix is used.Symmetric matrices make up a large fraction of the matrices used in track reconstruction. ATLAS Tracking Software makes heavy use of matrix algebra, implemented with the **Eigen** library. The CLHEP maths library was previously used throughout the ATLAS software, but after investigating potential alternatives, the decision was made to move to Eigen. Eigen was chosen since it offered the largest performance improvements for ATLAS.

Literature Survey

1. **ROOT Framework**

   ROOT is an object-oriented program and library developed by CERN. It was originally designed for particle physics data analysis and contains several features specific to this field, but it is also used in other applications such as astronomy and data mining.

   ROOT is optimized for describing small matrices and vectors and **It can be used only in problems when the size of the matrices is known at compile time**, like in the tracking reconstruction of physics experiments.

   The Matrix Class of ROOT**(SMatrix)**

   The template class **ROOT::Math::SMatrix** represents a matrix of arbitrary type with nrows x ncol dimension. Only limited linear algebra functionality is available for SMatrixThe class has following 4 template parameters, which define at compile time, its properties:

   1. type of the contained elements, T, for example float or double;
   2. number of rows;

3. number of columns;
4. representation type. This is a class describing the underlined storage model of the Matrix. Presently exists only two types of this class:
    1. **ROOT::Math::MatRepStd -** for a general *n* rows x *n* cols matrix.
    2. **ROOT::Math::MatRepSym** - for a symmetric matrix of size NxN. This class is a template on the contained type and on the symmetric matrix size N. It has as data member an array of type T of size N*(N+1)/2. It does not provide the complete linear algebra functionality but the basic matrix and vector functions such as matrix-matrix, matrix-vector, vector-vector operations, plus some extra functionality for square matrices, like inversion and determinant calculation.

Criticisms of ROOT include its difficulty for beginners, as well as various aspects of its design and implementation. Frequent causes of frustration include extreme code bloat, heavy use of global variables, and a perverse class hierarchy. But as of now, Root is mainly used for plotting and data analysis at CERN.

2. Issues like class structure issues, functionality issues, general design issues. Also, **not all matrix and vector operations are implemented, unlike Eigen or CLHEP.**
3. **The CLHEP project** - A Class Library for High Energy Physics .It is C++ library that provides utility classes for general numerical programming, vector arithmetic, geometry, pseudorandom number generation and specifically targeted for high energy physics simulation and analysis software. It contains vector and matrix classes for different dimensions, linear algebra functions, and geometry packages. All classes are connected to a generic interface for easy use.
4. The Event Data Model(which analyzes the Raw Data produced during a collision), as well as the algorithmic codes, were based on CLHEP. After thorough research, it was realized that CLHEP was one of the bottlenecks in the ATLAS software. So ATLAS switched to EIGEN algebra library.

   The Event Data Model(which analyzes the Raw Data produced during collision) as well as the algorithmic codes were based on CLHEP. After thorough research it was realized that CLHEP was one of the bottlenecks in the ATLAS software. So ATLAS switched to EIGEN algebra library.
5. **The Eigen Algebra Library** - Eigen is a C++ template library for linear algebra. It is a pure C++ library and does not have any dependencies beyond libstdc+ +. A separate compilation of Eigen is not necessary because it is composed of header files only and is then compiled with the application. Eigen provides different optimizations appropriate to the size of the matrices it is using. It supports SIMD(single instruction multiple data) vectorized instructions for basic operations such as 4 X 4 matrix multiplication. Unlike the ROOT Framework, Eigen can be used for dynamically sized matrices. It is used by e.g. the Google

Tensorflow, Ceres Large Survey Synoptic Telescope projects as well as ATLAS. Several performance profiling tests using PAPI, PIN Tool(for CPU time) e.t.c were performed on these frameworks and libraries. In case of the CPU time for multiplication of two four-dimensional matrices, Eigen performed best. These results combined with the results of the CPU performance comparisons in a small test framework showed Eigen performs the **fastest** for linear algebra operations.

Issue

One of the shortcomings of Eigen is that currently, it does not provide any separate template class for Symmetric Matrices which are heavily used in ATLAS software. So to store a symmetric matrix, one has to use a general **Eigen:: Matrix**. Due to this lot of space (think in terms of PB) is wasted for nothing since the upper and lower triangular parts of a matrix are same. Also, a lot of CPU computing power is used.

Solution

The current need is a Symmetric Matrix class just similar to the **Eigen:: Matrix.** A working implementation of symmetric matrices in Eigen, ready to be submitted as a patch for Eigen. This will help ATLAS find particles while using less computing power and less storage space.

Project Goals

Provide a working implementation of symmetric matrices and all the operations related to a matrix in **Eigen** which can be submitted as a patch for Eigen.

Step 1: Create a standalone template Symmetric class similar to **Eigen:: Matrix**

Step 2: Add different types of typedefs similar to typedefs of **Eigen:: Matrix**

Step 3: Implement constructors for the defined typedefs, member functions and operator overloadings which are there in **Eigen:: Matrix**

Step 4: Add Exception handling in all possible sections.

Step 5: Add Coefficient accessors, comma-initialization method.

Step 6: Add all the resizing methods like resize(), Assignment and resize methods similar to **Eigen:: Matrix**

Step 7: Testing of all the member functions, constructors, operators and there integration with **Eigen:: Matrix** class with all possible test cases.

Note that several parts of Step 1, Step 2, Step 3, Step 4 and Step 5 have already been implemented during the evaluation phase of the project. The code and compilation details are available at this GitHub link http://github.com/abhising10p14/Symmetric-Matrix .