Faster matrix algebra for ATLAS

Parismita Das

March 18, 2018

1. Project info

Project title: Faster matrix algebra for ATLAS

Corresponding Project: ATLAS

Participating Organizations: RAL

URL of project idea page:
http://hepsoftwarefoundation.org/gsoc/2018/proposal_ATLASEigen.html

2. Contact Information

Student name: Parismita Das

Student postal address: Ft – 803, Sawan Highness, Sector-6, Kharghar, Maharashtra, India, pin: 410210

Telephone(s): 9435660716

Email(s): das.parismita@gmail.com

Google+ : https://plus.google.com/u/0/111652042944176453050

Skype : parismita77

3. Student affiliation

Institution: Indian Institute of Technology, Guwahati

Program: B-Tech – Engineering Physics, Pr-Final Year

Stage of completion: Graduating in 2019

Contact to verify: registrar@iitg.ernet.in

* See Appendix

4. Bio of Student

Academic background

I am a Pr-Final year undergraduate student, in Engineering Physics at the Indian Institute of Technology, Guwahati. I have a good academic background in linear algebra, statistics and numerical computations. Some of the relevant courses that I have taken, include, computational physics; linear algebra; statistics; data structures and algorithms; monte carlo techniques; ODE and PDE. I have been using programming languages such as C++, Python and R since 3 years in coursework assignments & academic projects, etc. From the experience gained upon working with these, I have developed the mathematical background to take on this project.

Projects

I possess strong programming skills in C, C++, R and have a good knowledge of machine learning. As such I have not worked with any big project related to Matrix computations but I have done several Matrix computation algorithms in the computational physics course such as cholesky, LU, jacobi, power method etc. The link to the codes is Here.

And here are the list of projects I have worked on and contributed to and gained experience on software development.

- Pion mass reconstruction using dbscan
- Object detection, image to text conversion and face and emotion detection using Cnn, Command Interpretation using nlp with nltk. (link)
- HDLSS(High Dimension Low Sampling) analysis of Gene to Detect Cancer using glmnet.
- I have contributed to Cloud CV, Evalai in 2017 (link)
- Autonomous object collecting bot (link)
- Home Automation (link)

Here's the link to my cv :
https://drive.google.com/file/d/1m9aPSxECp-pttY_3jRW9RJ49C4AcrLmO/view?usp=sharing

Here is my Github profile :

https://github.com/parismita

5. Schedule Conflicts

I have no schedule conflicts that will interfere with treating Gsoc as a full time job except that my college reopens on **24th July.** Hence I will work from 12:30 GMT to 17:30 GMT (5 hours per day) and I plan to manage the extra hours on Sunday's so that I can meet the minimum (40 hours) criteria during the period: **24th July – 14th August.** And, as indicated in my proposed timeline, I plan on doing the necessary preparatory work before this

* See Appendix

period.

I am not applying to any other job or internship. Besides the aforementioned, I do not have any other planned time commitments either. Also I do understand that GSoC is a full time paid internship and I plan to commit about 40 hours per week for it.

6. Mentors

- Stewart Martin-Haugh <stewart.martin-haugh@stfc.ac.uk >
- Dmitry Emeliyanov <dmitry.emeliyanov@stfc.ac.uk >

I have been in touch with the mentors via email.

7. Coding Plan and Methods

Following is a basic plan of the components that the package will have.

- Symmetric Matrix class analogous to Eigen::Matrix class
  - Typedef to cover the usual cases like int, float, double and matrix size 2, 3, 4.
  - Handle Dynamic size that is unknown at compile time.
  - Constructor taking Eigen::Matrix , accessors and mutators.
  - Comma separated initializer
  - Resize function, to resize the matrix conserving its matrix size (rows*cols), return Eigen::Matrix.

- Symmatrix matrix arithmetic for 2 symmetric matrices or symmetric matrix with Eigen::Matrix and Eigen::Vector.
  - Binary, unary, compound operators
  - Addition, Subtraction, Multiplication, Division, Dot product, Cross product, Transposition, conjunction.
  - Reduction operations to reduce a given matrix or vector to a single value like sum, product, trace.

- The block operations like .block() of Eigen.

- Reductions, visitors and broadcasting
  - norm computation like lpNorm() and squareNorm() in Eigen.
  - Boolean reduction using any(), all(), count().
  - Visitors, to obtain the location of a coefficient, like greatest or smallest coefficient.
  - broadcasting , where a vector (column or row) is interpreted as a matrix by replicating it in one direction like colwise() and rowwise() of Eigen.

- Geometry
  - Transform
  - Translation
  - Scaling
  - Rotation 2D and 3D (Quaternion, AngleAxis)

* See Appendix

- LU decomposition with solver
    - Fully Pivoting LU
    - Partial Pivoting LU

- Cholesky factorization with solver
    - LLT
    - LDLT

- SVD decompositions with least-squares solver
    - JacobiSVD
    - BDCSVD

- Eigenvalue, eigenvector decompositions
    - EigenSolver
    - SelfAdjointEigenSolver
    - ComplexEigenSolver

- Unit tests for each component will be used to guide the development.
    - Unit tests will be implemented using testthat package
    - Continuous integration using Travis
- Documentation
    - Inline package documentation using roxygen2
    - Vignette
    - Include examples on how to use the included data.

If time permits , the following components will be implemented.

- Householder transformations

- QR decomposition with solver
    - HouseholderQR
    - ColPivHouseholderQR
    - FullPivHouseholderQR

8. Timeline

- **Preparation and pre-gsoc period (till April 23)**:
    - Understand the algorithm and the maths more clearly.
    - Read the Eigen implementations for LU,Cholesky, QR, SVD, Householder

- **Community Bonding (April 23 - May 14)**
    - Discuss and finalize the package components and methods.

* See Appendix

- ➢ Setup Travis in the github repository.
- ➢ Plan on the pseudo codes for all the modules.

- ❖ **Week 1 (May 14 - May 21)**
  - ➢ implement Class of Symmetric matrices with its basic arithmetics as mentioned in the components.
  - ➢ Write unit-tests for checking the correctness of the code
  - ➢ write Use-Case examples

- ❖ **Week 2,3,4 (May 21 - June 11)**
  - ➢ Implement reduction and block operations
  - ➢ Implement visitors and broadcasting operations.
  - ➢ Write unit-tests for checking the correctness of the code.
  - ➢ write Use-Case examples
  - ➢ Documentation for the completed modules.

- ❖ **First Evaluations (June 11 - 15, 2018)**

- ❖ **Week 5 (June 11 - June 15)**
  - ➢ Buffer time: fix bugs, clean up, finish any part of the coding or documentation that still remains.
  - ➢ Documentation.

- ❖ **Week 5,6 (June 15 - June 30)**
  - ➢ Implement the geometrical operations such as rotation, translation etc.
  - ➢ Unit-Tests and Use cases.
  - ➢ Documentation.
- ❖ **Week 7 (June 30 - July 9)**
  - ➢ Write module for LU decomposition and Cholesky factorization.
  - ➢ Unit-Test, Use-Cases and Documentation.

- ❖ **Second Evaluations (July 9 - 13, 2018)**

- ❖ **Week 8 (July 9 - July 13)**
  - ➢ Buffer time: fix bugs, clean up, finish any part of the coding or documentation that still remains, and move forward.

* See Appendix

- **Week 9 (July 14 - July 22)**
  - Eigenvalue solver
  - Unit-Test, Use-Cases and Documentation.

- **Week 10, 11 (July 22 - August 6)**
  - Write module for SVD decomposition
  - Testing and Performance analysis of each module.
  - Documentation.

- **Week 12 (August 6 - August 14)**
  - Buffer time
  - code clean up.
  - Testing and bug fixes.
  - Ensure that the package meets the required guidelines.

- **August 6-14, 2017 - Students Submit Code and Evaluations**

9. Success Criteria

While the exact scope of the project may change during the course of the summer, I have defined the following goals for this summer:

Mid-term: Complete with defining Symmetric Matrix class with all basic functionalities.

End-term: A complete package with all the components and documentations as described above.

What is your contingency plan for things not going to schedule?

While I will try my best to keep the project on time, I know from past experience that it is possible to get stuck at times, which might throw off the timeline. I have tried to include buffer time during the coding period for such instances. In the scenario of the project being finished early, there are many possible things to do such as :

- Householder transformations

- QR decomposition with solver
  - HouseholderQR
  - ColPivHouseholderQR
  - FullPivHouseholderQR

* See Appendix

10. Expected impact

In an environment like the Large Hadron Collider, we have millions of collisions per second, each producing thousands of particles passing through our detectors. Particle physics experiments like ATLAS rely on thousands of computers using clever algorithms to identify different types of particles and find interesting collisions, such as the production of Higgs bosons or even new particles. The optimisation of these algorithms can dramatically improve the physics we are sensitive to. ATLAS pattern recognition makes heavy use of matrix algebra, implemented with the Eigen library.

Currently there is no module for symmetric matrix in Eigen library, So the expected impact of this project will be a working implementation of symmetric matrices in Eigen, ready to be submitted as a patch for Eigen.

11. Management of Coding Project

How do you propose to ensure code is submitted / tested? How often do you plan to commit? What changes in commit behavior would indicate a problem?

How do you propose to ensure code is submitted / tested?

I will host the project on github, and commit frequently. Unit tests have been planned. I will try to use the unit-tests-first approach to development wherever possible.The Travis account will be setup for this project for continuous integration.

How often do you plan to commit?

I usually maintain a separate development branch with micro commits for each small addition / change. I would expect regular commits everyday to this branch.

What changes in commit behavior would indicate a problem?

No progress for more than a couple of days, with no prior information about vacation or some issues being faced would be an indicator of some problem.

How can we keep track of your progress?

I plan on getting regular code reviews from my mentor. This will also be very important to ensure code quality.

12. Test

Implementation of a Class SymMat, that takes Eigen::Matrix as input in constructor and constructs a upper triangular matrix, such that the computation time reduces for all basic arithmetics.

Also, code for assessors that can get value of S(i,j)=S(j,i). And mutations to change the

* See Appendix

value of S(i,j).

Basic arithmetic functions are built such as addition, subtraction and multiplication of SymMat-SymMat objects or SymMat-Matrix objects as arguments.

The subtraction and multiplication is done both for SymMat-Matrix objects and Matrix-SymMat objects.

When the matrices are not of same size, or incompatible, there is exception handling done properly.

There is also a module for testing the accuracy of the functions.

And an example file with example on how to use the functions and prove its correctness.
To find the solutions to the tests, here's the link :
https://github.com/parismita/gsoc-atlas-test-solutions

References

[1] http://eigen.tuxfamily.org

* See Appendix