A Programming Platform for Numerical Analysis and Data Visualization

Alex Clevenger and Michael Speckhart

Department of Computer Science and Engineering

Abstract

We live in a fast-paced world. This is a world where the pace is driven not only by the competitive market, but also by our rapidly advancing technologies. As a society, we are at the gateway of innovation with the rise of artificial intelligence, quantum computing, medical imaging, and robotic automation. The performance of these technologies all rely on one thing: fast computation. Most modern services deal with an astronomical amount of data, and organizing these data in specific matrix formats is becoming an industry norm. We present a comprehensive, user-friendly platform for performing numerical analysis and matrix operations. Advanced matrix operations and numerical systems are at the forefront of our most sophisticated advancements today, and there is much room for improvement in optimizing these computations. Our software supports matrix operations in both dense and sparse formats, and we offer multiple approaches for solving linear systems. We aspire to develop a powerful library that is capable of enhancing the technologies of today.

Motivation

- Explore and promote the field of numerical analysis
- Provide a streamlined and performant solution for matrix operations and linear system solving

Solution

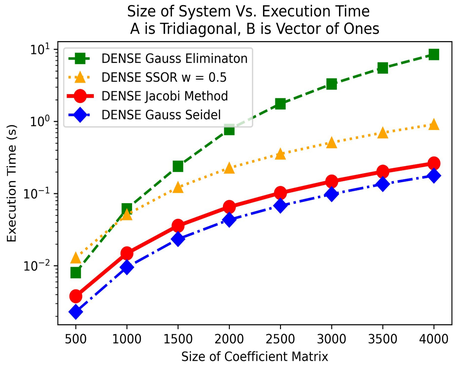
- Compressed sparse vector format only stores nonzero values to improve performance and memory allocation

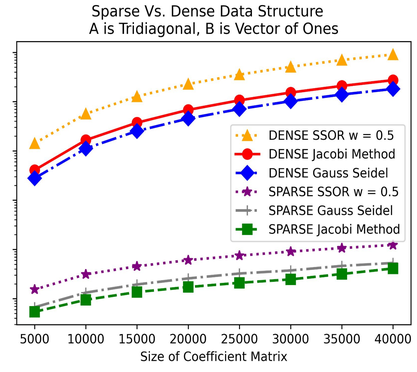
$$A = \begin{pmatrix} 7.5 & 2.9 & 2.8 & 2.7 & 0 & 0 \\ 6.8 & 5.7 & 3.8 & 0 & 0 & 0 \\ 2.4 & 6.2 & 3.2 & 0 & 0 & 0 \\ 9.7 & 0 & 0 & 2.3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5.8 & 5.0 \\ 0 & 0 & 0 & 0 & 6.6 & 8.1 \end{pmatrix}$$

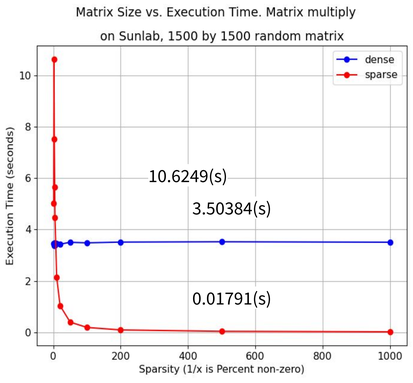
```
rowptr:          ( 0  4  7  10 12 14 16 )
colind: ( 0  1  2  3  0  1  2  0  1  2  0  3  4  5  4  5 )
val:    ( 7.5 2.9 2.8 2.7 6.8 5.7 3.8 2.4 6.2 3.2 9.7 2.3 5.8 5.0 6.6 8.1 )
```

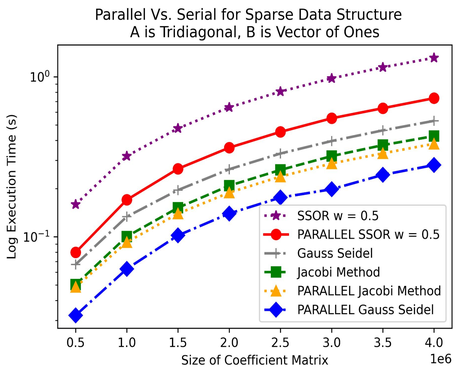
- Iterative methods and Krylov methods to solve linear systems of sparse matrices
- Parallelizing code with TBB to improve performance
- SIMD intrinsics to take advantage of hardware
- User-friendly API that allows users without programming knowledge to use the functions smoothly

Results







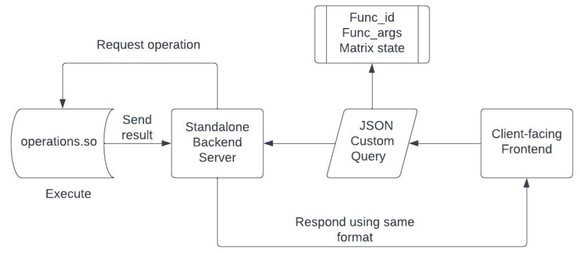


- The three graphs show our progression of developing solvers for Ax=b
  - Right graph: Indirect methods are much faster than direct methods
  - Bottom left graph: Sparse data structures are more efficient in processing power and memory than dense data structures
  - Bottom right graph: Iterative methods are embarrassingly parallel
- Parallel Sparse Gauss Seidel was 300,000x faster than Serial Dense Gauss Elimination

This work is done under the CSE/CSB capstone project 2023. We would like to thank Professor Arielle Carr, Professor Elroy Sturdivant, and Professor Corey Montella for your supervision and guidance.

Server Architecture

- Our interface is made up of two components:
  - A backend server written in C++ using the Boost Asio framework that directly calls the functions iterative solvers
  - A frontend web application written in Javascript using React and Axios that sends JSON http requests to the backend
- The service is hosted on Prof. Carr's Raspberry Pi
- We use an asynchronous, multi-threaded method of accepting data and allowing multiple connections



Conclusion/Future Work

- Sparsity patterns significantly impact the convergence of iterative solvers.
- Parallelizing components of iterative methods, particularly with the Jacobi Method, Gauss-Seidel, and SSOR, leads to enhanced performance.
- Separating the client interface from iterative solvers ensures performance maintenance without significant overhead.
- Our project establishes a robust baseline for seamlessly integrating more advanced iterative solvers into a user-friendly application, poised for future performance optimization.
- The future team can explore additional Krylov methods for solving linear systems and/or delve into preconditioning techniques.