# The Jacobian Matrix: Reloaded

**Team Name:**
Boom, Oliver
Kumar, Hitesh

February 22, 2019

**Abstract**

A program was written in the C++ programming language to enable a user to solve a linear set of equations $\mathbf{Ax} = \mathbf{b}$ where $\mathbf{A}$ is a positive definite matrix, $\mathbf{x}$ is the model vector and $\mathbf{B}$ is the data vector. The user is presented with a selection of algorithms with which to solve this system.

## 1 Summary

Solving linear systems of equations is a common occurrence within the fields of science and engineering. There are many algorithms that can be used to solve these systems, with varying complexity and performance. A selection of direct and iterative solvers have been compiled together, allowing the user to solve a fully determined system by a method of their choice. The system of equations are assumed to be densely populated, and so sparse matrix techniques were not considered. Additionally an algorithm to generate random symmetric positive definite matrices of a defined input size was created. This was a proprietary algorithm developed by the authors.

## 2 Code structure

The software was structured with 6 C++ source files; *main, user_interface, matrix*[1]*, solvers, generate_spd* and *tests*, with each file (with the exception of *main*) having a corresponding header file.

Two classes were created in their respective namesake files; *Matrix* and *Solver*. *Matrix* handles the initialization of matrices, can load pre-defined matrices for testing, performs basic matrix operations and has pretty printing functionality. While *Solver* performs all operations required in the different solution algorithms. The *Solver* class also performs compatibility checks between $\mathbf{A}$ and $\mathbf{b}$ before any attempt to solve the system is made.

## 3 Solver algorithms

| Algorithm | Approach |
|---|---|
| Jacobi [2] | Iterative |
| Gauss-Seidel [2] | Iterative |
| Gauss-Seidel with SORs [3] | Iterative |
| Lower Upper Decomposition [4] | Direct |
| Cholesky Decomposition [5] | Direct |

**Table 1:** Summary of the available algorithms

There is extensive online documentation of these algorithms, and the resources describing these techniques have been included in the references. It is expected that the user is familiar with these algorithms and will therefore know which approach will be most appropriate for their system of equations. However in this instance, direct methods will always be faster than the iterative methods.

The only algorithm which had a notable deviation from the classical descriptions of these algorithms was in the lower upper decomposition approach. The approached used was the Doolitle method which has the added advantaged of avoiding pivot operations.

Both the Jacobi and Gauss-Seidel methods have the ability to choose a weighting that determines the ratio of old to new information (conventionally $w = 2/3$ for weighted Jacobi and $1 < w < 1.9$ for successive over-relaxation.

For the direct solvers, if the $\mathbf{A}$ matrix is a conventional general positive definite matrices the LU decomposition can be used. If the $\mathbf{A}$ matrix also has the property of being symmetric, then the Cholesky-decomposition algorithm can be used for increased performance.

## 4 Execution

A user-friendly command-line interface was developed. When the program is executed, the user is greeted with a selection of options. They can run a selection of predefined test to validate the algorithms are functioning correctly. Then there is either the option to creates a random positive definite matrix of a chosen size, or the user can input a custom matrix of their choice. Then the user can select which algorithm is used to solve the system and they are presented with the results.

## 5 Reflection

The main area for improvement would be a bug in the Jacobi solver for matrices over size 6, which causes divergence. For these matrices it is suggested that the Gauss-Seidel or the other algorithms are used. Other software improvements which were not implemented but would be desired features to include; operational overloads on the matrix operations, inclusion of a multi-grid solvers and algorithm performance metrics to accompany the results. Additionally, there are memory leaks in the LU and Cholesky decomposition functions, which do not effect performance and could be fixed with smart pointers.

## References


[1] Imperial acse advanced coding lecture series.

[2] University of notre-dame url = https://www3.nd.edu/~zxu2/acms40390F12/Lec-7.3.pdf.

[3] Wolfram alpha. url = http://mathworld.wolfram.com/SuccessiveOverrelaxationMethod.html.

[4] Mathonline - mathwiki. url = http://mathonline.wikidot.com/doolittle-s-method-for-lu-decompositions.

[5] G. fasshauer - lecture series . url = http://www.math.iit.edu/~fass/477577_Chapter_7.pdf.