Code Structure and design:

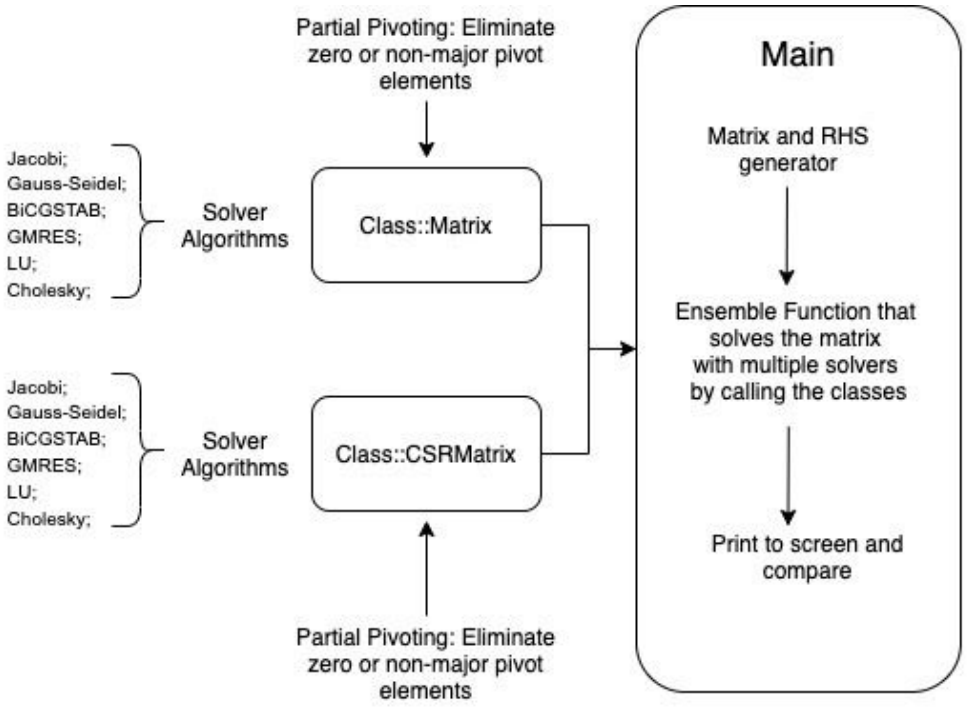

Figure 1. Code Structure and design

Brief Solver Algorithm introduction

Jacobi method is an iterative process to produce new x values using the x values obtained from the previous iteration until the error is smaller than our tolerance. **Gauss-Seidel** is very similar to the Jacobi method. The only difference is that Gauss-Seidel uses the x values of this iteration as soon as they are calculated combined with x values in the previous iteration.

GMRES is also an iterative method that first uses Arnoldi method to find a vector in a Krylov subspace then approximates the correct solution using this vector until the minimal Euclidean norm of the residual. **BiCGSTAB** is a combination of BiCG and GMRES. It uses previous x values and searches directions and residuals to get the new x values.

LU factorization $A = LU$ and **Cholesky** factorization $A = LL^T$ are implemented. Both of LU and Cholesky are factorizing a square nonsingular matrix A into a product of triangular matrices. The lower and upper triangular matrices produced by Cholesky factorization are transposes of each other.

Jacobi method and **Gauss-Seidel** are implemented when the diagonal elements of the coefficient matrix A can't be zeros. One sufficient convergence condition for both is that the matrix A is diagonally dominant. **Gauss-Seidel** has a smoother convergence than Jacobi. And a more sufficient convergence condition for this method is that A is symmetric positive-definite.

GMRES and **BiCGSTAB** don't have the limitations that the diagonal element must be non-zero. And they can also compute relatively correct solutions to non-diagonally dominant matrices. And BiCGSTAB is faster than GMRES but GMRES can be applied to an n*m matrix.

LU and **Cholesky** factorization can only be implemented on non-singular matrices, while Cholesky requires the matrix to be symmetric and positive-definite in the context of real matrices.

Partial Pivoting:

Pivoting elements are important in some algorithms, for instance, Jacobi, Gauss-Seidel, LU, and Cholesky. These algorithms require pivoting elements as the denominators that assign values to non-pivoting elements. Thus, a zero or close-to-zero pivoting element is the main source of instability for these algorithms. Partial pivoting that assigns value, which is non-zero and relatively dominant, to the diagonal position can improve the stability for both methods. This is implemented in our mat_sort function.

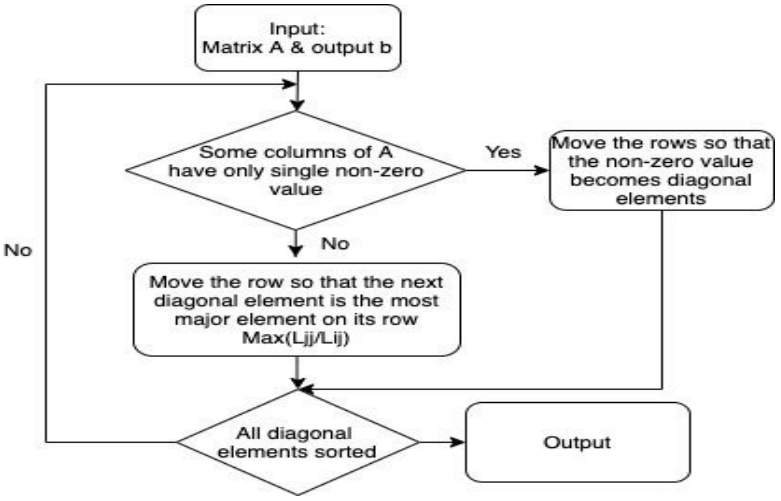

Figure 2. Partial Pivoting by *sort_mat*

```
printing matrix

140 1 2 2 5 5 7 9 7 9
1 0 6 1 19 7 9 9 16 3
7 1 150 3 2 3 2 8 8 2
2 3 6 190 2 3 1 3 3 3
1 100 3 7 0 4 2 3 1 1
2 2 4 1 9 170 7 3 9 8
5 16 1 7 3 9 120 7 9 3
6 0 6 5 4 4 5 160 6 8
7 1 5 8 2 6 3 9 170 7
2 2 1 7 5 1 4 3 5 140


After pivoting
printing matrix

140 1 2 2 5 5 7 9 7 9
1 100 3 7 0 4 2 3 1 1
7 1 150 3 2 3 2 8 8 2
2 3 6 190 2 3 1 3 3 3
1 0 6 1 19 7 9 9 16 3
2 2 4 1 9 170 7 3 9 8
5 16 1 7 3 9 120 7 9 3
6 0 6 5 4 4 5 160 6 8
7 1 5 8 2 6 3 9 170 7
2 2 1 7 5 1 4 3 5 140
```

```
non-zero values
14 2 10 16 3 2 3 11
row_position
0 2 3 5 8
col_index:
1 3 0 2 3 1 2 3

After pivoting
non-zero values
10 14 2 16 3 2 3 11
row_position
0 1 3 5 8
col_index:
0 1 3 2 3 1 2 3
```

Figure 3. Example of sort_mat (Dense and CSR format respectively)

Comparison and Discussion:

For these implemented functions, we tested their running time by passing in matrices with different properties (symmetric and diagonal-dominant) and size, one of which is shown below (complete version please check Readme.md):

Table 1: Run time (in ms) of Symmetric Diagonal-dominant Matrix

	Jacobi	Gauss-Seidel	BiCGSTAB	GMRES	LU	Cholesky
10*10	0.21	0.08	0.03	16.091	0.006	0.005
100*100	19.29 (max iteration reached)	0.66	0.62	2.65	1.48	0.67
500*500	464.12 (max iteration reached)	16.65	9.30	6.11	158.56	66.65
1000*1000	1882.87 (max iteration reached)	63.91	38.33	19.37	1305.10	524.08

The comparison between different solvers implicate the following points:

- Though Gauss-Seidel and Jacobi are mostly similar, the convergence rate is much faster and the stability is better in Gauss-Seidel;
- Symmetricity does not influence the performance of Jacobi, Gauss-Seidel, and BiCGSTAB significantly, while all these methods do not converge and are possible to blow up when non-diagonal-dominant matrices are passed in;
- LU decomposition with pivoting performs stably for all test conditions;
- In contrast with all the other 5 algorithms we implemented, the GMRES algorithm is not sensitive to the size of matrices;

Table 2: Comparison between Sparse and Dense Matrix

	Jacobi	Gauss-Seidel	BiCGSTAB	GMRES	LU	Cholesky
100*100 dense	19.29 (max iteration reached)	0.66	0.62	2.65	1.48	0.67
100*100 sparse	0.373	0.325	0.399	29.954	10.24	6.601

Compared with dense matrix, time consumed in solving a sparse matrix with CSR format is less in the first three methods. In contrast, the latter three methods spend more time. This is caused by our complicate indexing method in CSR format, which caused over-complicity in the GMRES, LU, and Cholesky. This is consistent with the presumption that, though CSR format is efficient in data storage, it may not be efficient in the context of data manipulation and treatment. It's also presumed that when it comes to the larger size of matrices, the data manipulation will become one of the advantages of CSR format.