Problem 1: Write codes for LU-decomposition (Gaussian elimination) which can handle row exchanges. We will apply it to solve $A\mathbf{x} = \mathbf{b}$. A is an $n \times n$ matrix with each entry a random number in $[-1, 1]$, with $n = 100$. Take the exact solution $\mathbf{x}$ as a random vector and calculate the corresponding $\mathbf{b}$. Then apply your codes to solve $\mathbf{x}$ from $A\mathbf{x} = \mathbf{b}$.
Consider the following pivoting strategies: (1) no pivoting; (2) partial pivoting; (3) scaled partial pivoting. For each strategy, run the above numerical test 5 times and print the error $\|\mathbf{x}_{numerical} - \mathbf{x}_{exact}\|$ (with any matrix norm).
Repeat the above with a modified matrix A: each row of the original A is multiplied by a random number between 1 and 1000.

Problem 2: Write codes for Jacobi iteration, Gauss-Seidel iteration and conjugate gradient method. Make sure your codes can take advantage of sparse matrices. Consider an $n \times n$ tri-diagonal matrix A with $a_{ii} = a_0$, $a_{i,i+1} = a_+$, $a_{i,i-1} = a_-$ with $n = 1000$. Generate $\mathbf{x}, \mathbf{b}$ as in Problem 1.
Test Jacobi, Gauss-Seidel with the following examples: $(a_0, a_+, a_-) = (2, 1, -1)$, $(a_0, a_+, a_-) = (1, 0.3, 0.5)$. Test all three methods with the following examples: $(a_0, a_+, a_-) = (2, -1, -1)$. You should print the number of iterations when the desired tolerance is achieved. You should also print the error $\|\mathbf{x}_{numerical} - \mathbf{x}_{exact}\|$ by using the known exact solution, and observe how it is related with the tolerance you used.
For the last example, also apply the SOR method with a few choices of the parameter ω, and observe which choice gives the fewest number of iterations.

Problem 3: Write codes for preconditioned conjugate gradient method. Take A as the following $n \times n$ tri-diagonal matrix with $n = 1000$: generate the numbers $c_1, \ldots, c_{n-1}$ as random numbers between 1 and 100; $a_{i,i+1} = a_{i+1,i} = -c_i$ for $i = 1, \ldots, n-1$; $a_{ii} = c_i + c_{i-1}$ for $i = 1, \ldots, n$, with c_0 and c_n regarded as 0. Generate $\mathbf{x}, \mathbf{b}$ as in Problem 1. Compare the number of iterations for the original conjugate gradient method and the preconditioned conjugate gradient method with the Jacobi preconditioner.

Problem 4: Consider the following $m^2 \times m^2$ matrix A: the diagonal entries are all 4; For the other (i, j) entries, write $i = (i_1 - 1)m + i_2$ with $i_1, i_2 \in \{1, \ldots, m\}$, and the (i, j) entry is -1 whenever $|i_1 - j_1| + |i_2 - j_2| = 1$, otherwise the (i, j) entry is 0. (This matrix arises when solving 2D Poisson equation. It is irreducibly diagonally-dominant, symmetric positive definite, sparse, but does not have a tri-diagonal structure.)
For $m = 10, 20, 40, 80$, apply Gaussian elimination and conjugate gradient method to solve $A\mathbf{x} = \mathbf{b}$, where $\mathbf{x}, \mathbf{b}$ are generated as in Problem 1 (with reasonable tolerance for conjugate gradient method). Compare their run times.

Problem 5: Write codes for the power method for matrix eigenvalues. Run it for a random complex $n \times n$ matrix with $n = 10$ and obtain its largest eigenvalue (in absolute value) and its eigenvector. Check your result by checking the equation $A\mathbf{v} = \lambda\mathbf{v}$. Do this test for 10 different matrices.
Do the same for random real matrices. Do you observe any different phenomenon? Try to explain.