Important to Remember:

- Complete the assignment individually
- Include a front page with name, email and assignment number
- Present detailed solutions that are easy to follow together with comments to your results and answers to the questions that are asked.
- Use MATLAB as a tool for your computations and give a link to the code necessary to reproduce your results
- Use visualizations whenever appropriate
- Give references to the theorems you use
Exercise 1:

This exercise concerns basic iterative methods for solving the square linear system

$$Ax = b$$

(1)

where the $n \times n$ matrix $A$ is assumed to be large and sparse.

1.1) Write two Matlab routines `jacobi.m` and `gs.m` which implements Jacobi and Gauss-Seidel iteration. The input arguments should be the matrix $A$, the vector $b$, and a number $tol$ specifying the desired accuracy. Each routine should return the computed solution approximation $x$ and the number of iterations $k$ performed. Thus the syntax for calling the routines should be

$$[x,k] = \text{jacobi}(A,b,tol)$$

and

$$[x,k] = \text{gs}(A,b,tol)$$

Use a zero starting guess and iterate until the relative residual $\|r^{(k)}\|/\|r^{(0)}\|$ is less than $tol$. Test your codes by solving the linear system (1) with

$$A = \begin{bmatrix}
3 & 12 & 0 & -1 & 0 & 0 \\
4 & 0 & 31 & 1 & 0 & 0 \\
2 & 1 & 0 & 0 & 17 & -3 \\
27 & 2 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & -1 & 1 & 11 \\
0 & 0 & 0 & 24 & -1 & 0
\end{bmatrix}, \quad b = \begin{bmatrix}
39 \\
117 \\
12 \\
98 \\
14 \\
55
\end{bmatrix}$$

*Hint:* You might want to do row permutations before applying the iteration schemes.

1.2) Compare the number of iterations required by the Jacobi and Gauss-Seidel methods to converge to a given accuracy from a zero starting guess. Solve (1) with $A$ and $b$ defined by

```matlab
e = ones(n,1);
A = spdiags([-e 2*e -e], -1:1, n, n);
b = rand(n,1);
```

Record the number of iterations needed to achieve the tolerance $0.1$, $0.01$, $0.001$, $0.0001$ and $0.00001$ for a few different values of $n$, say $10$, $50$, and $100$. Plot your results and draw some conclusions.

1.3) Show that Jacobi iteration may take the form

$$x^{(k+1)} = x^{(k)} + H r^{(k)}$$

where $H$ is a matrix to be defined (by you) and $r^{(k)} = b - Ax^{(k)}$ is the residual at stage $k$. Can you interpret this result from the point of view of one-dimensional projection methods for $Ax = b$. Is it possible to write the Gauss-Seidel iteration on the above form? Explain how (or why if it is not possible).
Exercise 2:
In class we showed how to solve the Poisson problem

\[-( \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} ) = 1 \quad x, y \in \Omega = [0, 1] \times [0, 1] \]

\[u = 0 \quad x, y \in \partial \Omega\]

using finite differences on a square \((N + 1) \times (N + 1)\) grid. As you might recall this leads to a large sparse linear system \(Ax = b\) where the vector \(x\) contains the nodal values \(u_{i,j}\) of the finite difference solution approximation.

A simple way of setting up the finite difference method is to use the Matlab commands `numgrid`, `delsq`, and `surf` to construct the grid, build the coefficient matrix \(A\), and visualize the solution approximation. The following code does all this except solve the linear system.

```
% Generate grid G, matrix A, and vector b.
N = 20; h = 1/(N+1)
G = numgrid('S', N+2);
A = delsq(G);
b = h^2*ones(N^2, 1);

% Solve Ax = b
x = ??? % solve with PCG

% Map the vector x onto the grid and plot it.

u = G;
u(G>0) = full(x(G(G>0)));
[x,y] = meshgrid(0:1/(N+1):1);
surf(x,y,u)
```

Your task is to modify this code so that the linear system resulting from the finite difference discretization process is solved with the Preconditioned Conjugate Gradient method (PCG). Use the built-in `pcg` routine and study the residual norm at each iteration by plotting the `resvec` vector which can be output from `pcg`. Try to accelerate the convergence of the method by using an incomplete Cholesky factorization of \(A\) as preconditioner, \(P\). Set up \(P\) with \(P = \text{cholinc}(A,\text{droptol})\) and then call PCG with \(x = \text{pcg}(A,b,\text{tol},\text{maxit},P',P)\). A reasonable value of \(\text{tol}\) is \(10^{-6}\). Let \(\text{maxits}\) be \(1,000,000\). Record both the time used by `cholinc` to compute the preconditioner and the time used by `pcg` to solve the linear system. Try to find a drop tolerance which makes these times equal. Vary the grid size \(N\). Also, use `spy` to look at the sparsity pattern and the amount of fill-in in \(P\) for various values of the drop tolerance. Comment on your observations.

**Hint:** For timing you might find the commands `clock` and `etime` useful.
Exercise 3:

Consider the $m$-th Krylov subspace $K_m(A; b) = \text{span} \{ b, Ab, A^2b, \ldots, A^{m-1}b \}$, and the corresponding Krylov matrix

$$K_m = \begin{bmatrix} b & Ab & A^2b & \ldots & A^{m-1}b \end{bmatrix}$$

Let $A=\text{diag}([1 \ 3 \ 6 \ 9])$ and $b=[1 \ 1 \ 1 \ 1]'$.

3.1) Compute the Krylov matrix $K_4$. Then express the vector $x = A^{-1}b = [1, \frac{1}{3}, \frac{1}{6}, \frac{1}{9}]^T$ as a linear combination of the columns of $K_4$.

3.2) Use the Matlab code from the lecture notes to compute the $4 \times 4$ matrices $Q$ and $H$ in the Arnoldi factorization of $A$, (i.e., such that $AQ = QH$). Use $q_1 = b/\|b\|$ as starting vector. (Note that since the Arnoldi algorithm stops at stage 3, the last column of $H$ is not actually computed. It comes from a final command $H(:,4) = Q'*A*Q(:,4)$.)

3.3) Assume that we have run Arnoldi’s algorithm for 2 steps so that we (at least) have access to the orthogonal basis $Q_2 = [q_1, q_2]$ that span the Krylov subspace $K_2(A; b)$. Show how the matrix $H_2$ can be used to get a Galerkin solution $x_2$, that is, such that the residual $r_2 = b - Ax_2$ is orthogonal to the span of the basis vectors $q_1$ and $q_2$. Compute $x_2$. What is the residual $r_2$? For the above setup, $r_4$ will of course be zero (verify that), but is that the case for any $4 \times 4$ matrix $A$?