This project consists of three parts. The first part is worth 55 points, the second part is worth 25 points, and the third part is worth 20 points. You can earn up to 15 extra points for the quality of discussion.

You may work in teams of **up to three** persons. Code for your project should be submitted in either Java, Python, or MATLAB. (Choose only one of these languages to use for the whole project—do not mix and match.) Your final deliverable should be submitted in a single `.zip` archive file. The archive file should be uploaded to the dropbox on T-Square of *one* of your team members by **11:59 p.m.** on **November 24**. The file should contain:

- A `team_members.txt` file listing the name of each person in the project team.
- All source files for each part of the project you submit.
- A `.doc`, `.docx`, or `.pdf` file for each part of the project you submit, containing the written component of the project.
- A `readme` file for each part of the project you submit, explaining how to execute that part of the project.
- For Java or Python users, do not submit a project that has to be run on an IDE. The graders should be able to at least initialize your project on a command line interface, either Linux or Windows.
- The input matrices required in several parts of the project will be `.dat` files in ASCII or text format, that contain double floating point numbers as the matrix entries in the same row separated by spaces or commas; and different rows in the matrix are different rows in the file.

1 The Symmetric Pascal Matrix

1. (10 points) LU-decomposition. For the `lu_fact`, your code will be run against a number of test cases in `.dat` files. The test cases will consist of $n \times n$ matrices A of floating-point numbers with linearly independent columns, these numbers contains postive, zero, and negative real numbers. The output for each test matrix should be two matrices: an $n \times n$ lower triangular matrix L and an upper triangular matrix U such that LU equals the input matrix; and the error $||LU - H||_\infty$. Your code will be inspected manually: no points will be awarded if the function does not implement the LU-decomposition algorithm.

2. QR-factorization. For each of the following functions, your code will be run against a number of test cases in `.dat` files. The test cases will consist of $n \times n$ matrices A of floating-point numbers with linearly independent columns, these numbers contains postive, zero, and negative real numbers. The output for each test matrix should be two matrices: an $n \times n$ orthogonal matrix Q and an $n \times n$ upper-triangular matrix R

such that QR equals the input matrix; and the error $||QR - A||_\infty$. Your code will be inspected manually: no points will be awarded if the function does not implement the specified QR-factorization algorithm.

- (10 points) `qr_fact_househ`
- (10 points) `qr_fact_givens`

3. The implementation to solve the linear system $A\overline{\mathbf{x}} = \overline{\mathbf{b}}$ based on LU or QR factorizations will be tested on $n \times (n+1)$ augmented matrices $[A|\overline{\mathbf{b}}]$ in `.dat` files, where A is $n \times n$ and $\overline{\mathbf{b}}$ is $n \times 1$. The output should be the solution vector $\overline{\mathbf{x}}_{sol}$, and the error $||A\overline{\mathbf{x}}_{sol} - \overline{\mathbf{b}}||_\infty$. Your code will be inspected manually: no points will be awarded if the function makes use of an inverse matrix algorithm, or if the solution vector is incorrect.

   - (5 points) `solve_lu_b`
   - (5 points) `solve_qr_b`

4. (10 points) Solution and errors for the symmertic Pascal matrix of dimension $n \times n$, $n = 2, 3, \ldots, 12$ and $\overline{\mathbf{b}}$ as described in the project. This should be a routine (or set of routines) that calls the appropriate programs and returns a readable outputs as described in the projects for the input n.

5. (5 points) Plots and discussion. It should include plots of the errors obtained for the ~~Hilbert~~ Pascal matrix as a function of n. You should have 3 plots for the errors of $\overline{\mathbf{x}}_{sol}$, each obtained with LU and both QR decompositions; one for $||LU - P||_\infty$, and two more for $||QR - P||_\infty$ one for each case of QR-factorization. Your plots will probably look better in a Log scale, and can be grouped in a way that makes comparisons possible. The discussion should address the questions posted in the description of the project.

2 Convergence of the iterative methods

1. Iterative methods for $A\overline{\mathbf{x}} = \overline{\mathbf{b}}$. For each of the following functions, your code will be tested against a number of test cases. The input will be an initial guess vector $\overline{\mathbf{x}}_0$ of floating-point numbers, an error tolerance number tol, and a maximum iteration number M. The output of your code should be a approximate solution x_N and the the number of iterations N required to reach the tolerance $\|\overline{\mathbf{x}}_N - \overline{\mathbf{x}}_{N-1}\|_\infty$; alternatively, if the algorithm does not attain the desired accuracy within N iterations, some form of value indicating failure should be returned. (Please indicate in your documentation what the expected failure value is.) Your code will be inspected manually: no points will be awarded if the function does not implement the iterative method.

   - (5 points) `jacobi`

- (5 points) `gauss_seidel`

2. (5 points) Your program should generate 100 random input vectors $\overline{\mathbf{x}}_0$. Use both `jacobi` and `gauss_seidel` to these vectors $\overline{\mathbf{x}}_0$, $\epsilon = 0.00005$, and $M = 100$. Your program should record these 100 inputs $\overline{\mathbf{x}}_0$, output approximations $\overline{\mathbf{x}}_N$, and output steps N. Your code for part (b) will be manually inspected, but will not be run against any test cases.

3. Data analysis. For (c) and (d), average these 100 $\overline{\mathbf{x}}_N$ to obtain an approximation solution $\overline{\mathbf{x}}_{approx}$. Compute the error of this approximate solution to the exact solution $\overline{\mathbf{x}}_{exact}$ for both iterative methods. Compute these 100 the ratio of number of iteration steps $N_{Jacobi}/N_{Gauss-Seidel}$, then take the average. Plot 100 points for these 100 cases on a colored scatterplot, the x-axis is $\|\overline{\mathbf{x}}_0 - \overline{\mathbf{x}}_{exact}\|_\infty$ and y-axis is N. Use black scatters for Jacobi results, and blue for Gauss-Seidel results. All display specifications from the project description should be met (2 scat-terplots, correct axes, colored data points, etc.)

   - (2 points) average of $\|\overline{\mathbf{x}}_{approx} - \overline{\mathbf{x}}_{exact}\|$ and the scatterplot for `jacobi`
   - (2 points) average of $\|\overline{\mathbf{x}}_{approx} - \overline{\mathbf{x}}_{exact}\|$ and the scatterplot for `gauss_seidel`
   - (1 points) average ration of $N_{Jacobi}/N_{Gauss-Seidel}$

4. (5 points) Discussion: You should give a complete, well-written answer in a text, `.doc`, `.docx`, or`.pdf` file. The accuracy, completeness, and clarity of your writing will be considered in assigning credit. Be sure to discuss:

   1. the effect of initial vector position, 2. the steps needs in both methods. [Hint: the ratio of $N_{Jacobi}/N_{Gauss-Seidel}$ seems to be a fixed number, it might be related to eigenvalues of the iteration martix] 3. the graphs in (d).

3 Convergence of the Power Method

1. (5 points) `power_method`: Your code will be run against 5 test cases. The test cases will consist of $n \times n$ matrices of floating-point numbers, a vector $\overline{\mathbf{v}}$ of n floating-point numbers that serves as the initial guess for an eigenvector, a positive floating-point number ϵ, and a positive integer N. The output should be a floating-point number that approximates the largest eigenvalue λ of the input matrix in absoluate value and an associated vector of n floating-point numbers that approximates an eigenvector corresponding to λ; alternatively, if the algorithm does not attain the desired accuracy within N iterations, some form of value indicating failure should be returned. (Please indicate in your documentation what the expected failure value is.) Correctness of each test case will be worth 1 point. Your code will be inspected manually: no points will be awarded if the function does not implement the power method.

2. (5 points) Your code for part (b) will be manually inspected, but will not be run against any test cases.

3. Scatterplots:

   - (3 points) All display specifications from the project description are met (2 scatterplots, correct axes, colored data points, etc.)
   - (2 points) The data shown is generated in the manner set out in part (b)

4. (5 points) Discussion: You should give a complete, well-written answer in a text, `.doc`, `.docx`, or `.pdf` file. The accuracy, completeness, and clarity of your writing will be considered in assigning credit. Be sure to discuss:

   - The general shape of the scatterplots. It would be good to talk the mathematical reasons behind how the two plots are related. [Hint: Show that $\mathrm{tr}(A^{-1}) = \mathrm{tr}(A) / \det(A)$, where tr means *trace*. This fact may be useful for talking about the relationship between the two plots.]
   - The relationship between the position of a matrix on the plot and the number of iterations needed in the power method. [Hint: Let λ_1 be the larger eigenvalue of A in magnitude and λ_2 the smaller in magnitude. Consider the ratio $\|\lambda_1/\lambda_2\|$: how is it related to the power method?]