This project consists of three parts. The first part (worth 60 points) is required, and you may do either the second or third part (each of which is worth 40 points). If you submit all three parts, designate either the second or third part as extra credit. You can earn up to 20 points on the extra-credit part (you will receive half the score you would have received if it were not extra credit).

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.

1 The Gauss-Newton Method

(a) QR-factorization: For each of the following functions, your code will be run against 10 test cases. The test cases will consist of $m \times n$ matrices of floating-point numbers with linearly independent columns. The output should be two matrices: an $m \times n$ orthogonal matrix Q and an $n \times n$ upper-triangular matrix R such that QR equals the input matrix. 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 specified QR-factorization algorithm.*

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

(b) Gauss-Newton: For each of the following functions, your code will be run against 7 test cases. The test cases will consist of n pairs of floating-point numbers (the data points), a triple of floating-point numbers (the initial parameters), and an integer N (the number of iterations). The output should be three floating-point numbers. *Your code will be inspected manually: half-credit will be deducted if the function does not implement the modified version of Gauss-Newton using a QR-factorization subroutine, and no credit will be awarded if your function does not implement Gauss-Newton at all.*

- (7 points) `gn_qua`
- (7 points) `qn_exp`

- (7 points) `qn_log`
- (7 points) `qn_rat`

(c) Discussion: You should give complete, well-written answers in a text, `.doc`, `.docx`, or `.pdf` file. The accuracy, completeness, and clarity of your writing will be considered in assigning credit.

- (6 points) Why is it justified to modify the algorithm to set $\boldsymbol{\beta}$ to $\boldsymbol{\beta} - R^{-1}Q^{\top}\mathbf{r}$? (Your answer should explain why this is mathematically equivalent to the original version of the Gauss-Newton algorithm. A mathematical proof is expected.)
- (6 points) What is the benefit of modifying the algorithm in this way? (Your answer should consider the benefit in terms of conditioning error.)

---

2 Convergence of the Power Method

(a) (15 points) `power_method`: Your code will be run against 15 test cases. The test cases will consist of $n \times n$ matrices of floating-point numbers, a vector $\mathbf{v}$ 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 and an associated vector of n floating-point integers 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.*

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

(c) Scatterplots:

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

(d) (10 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 $\operatorname{tr}(A^{-1}) = \operatorname{tr}(A)/\det(A)$, 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?]

3 Animation in Two Dimensions

(a) (5 points) Set-up of polygons reflecting three letters for each frame

(b) (15 points) Coding of matrix transformations to achieve the animation: first letter should rotate 3 times with respect to the z-axis, the second letter should rotate 2 times with respect to the y-axis, and the third letter should rotate 5 times with respect to the x-axis.

(c) (10 points) The animated movie, which shows 120 frames at a rate of 24 frames per second

(d) (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. Describe the linear and nonlinear transformations you needed in order to create the animation.