Computer Project: Solving Least Squares Problems using QR Decomposition

Author: 樊泽義
Student ID: 2200010816
Date: 2025-10-15

1. Problem Statement

This project aims to implement and apply the QR decomposition method for solving both linear systems and linear least squares problems. The implemented algorithms are then utilized to complete three distinct computational tasks, serving as a practical test of their effectiveness and stability.

The first task is to solve three different linear systems of the form $A\mathbf{x} = \mathbf{b}$. The first system is a nearly tridiagonal matrix of size 84×84, where the matrix A_1 and vector $\mathbf{b}_1$ are defined as follows:

$$(A_1)_{ij} = \begin{cases} 6 & \text{if } i = j \\ 8 & \text{if } i = j+1 \\ 1 & \text{if } i = j-1 \\ 0 & \text{otherwise} \end{cases}, \quad (\mathbf{b}_1)_i = \begin{cases} 7 & \text{if } i = 1 \\ 14 & \text{if } i = 84 \\ 15 & \text{otherwise} \end{cases} \tag{1}$$

The second system involves a symmetric tridiagonal 100×100 matrix A_2 and vector $\mathbf{b}_2$:

$$(A_2)_{ij} = \begin{cases} 10 & \text{if } i = j \\ 1 & \text{if } |i-j| = 1 \\ 0 & \text{otherwise} \end{cases}, \quad (\mathbf{b}_2)_i = \begin{cases} 11 & \text{if } i = 1 \text{ or } i = 100 \\ 12 & \text{otherwise} \end{cases} \tag{2}$$

The third system is defined by a 40×40 Hilbert matrix A_3, which is known to be ill-conditioned. The components of the matrix and the vector $\mathbf{b}_3$ are given by the formulas:

$$(A_3)_{ij} = \frac{1}{i+j-1} \quad \text{for } i,j = 1,\ldots,40 \quad \text{and} \quad (\mathbf{b}_3)_i = \sum_{k=1}^{40} \frac{1}{i+k-1} \tag{3}$$

For this last system, the exact solution is known to be a vector of all ones.

The second task is a polynomial curve fitting problem. The objective is to find the coefficients of a quadratic polynomial, $y(t) = at^2 + bt + c$, that best fits a given set of data points in the least squares sense. This problem can be expressed as finding the vector $\mathbf{x}$ that minimizes $\|A\mathbf{x} - \mathbf{y}\|_2$, where the components are defined as:

$$A = \begin{pmatrix} t_1^2 & t_1 & 1 \\ t_2^2 & t_2 & 1 \\ \vdots & \vdots & \vdots \\ t_m^2 & t_m & 1 \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} a \\ b \\ c \end{pmatrix}, \quad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix} \tag{4}$$

The third task is a practical application in real estate price prediction. Based on 28 sets of housing data, the goal is to determine the parameters of a multiple linear regression model. The model is given by the equation $y = a_0 + a_1x_1 + a_2x_2 + \cdots + a_{11}x_{11}$. This is a classic least squares problem where we seek to find the parameter vector $\mathbf{a} = [a_0, a_1, \ldots, a_{11}]^T$ that minimizes the 2-norm of the residual between the predicted and actual house prices.

2. Algorithm Design and Implementation

The core of the solution method is the QR decomposition of a matrix A into the product of an orthogonal matrix Q and an upper triangular matrix R. This implementation employs Householder reflections, a numerically stable method for achieving this factorization. A Householder transformation is a reflection matrix of the form $P = I - 2\mathbf{v}\mathbf{v}^T$, where $\mathbf{v}$ is a unit vector. By choosing an appropriate $\mathbf{v}$ for each column, we can apply a sequence of these transformations to systematically zero out the elements below the main diagonal of A, yielding the matrix R. The product of these Householder matrices then forms the orthogonal matrix Q.

To solve a square linear system $A\mathbf{x} = \mathbf{b}$, we first perform the QR decomposition to get $A = QR$. The system transforms into $QR\mathbf{x} = \mathbf{b}$. Since Q is orthogonal, its transpose is its inverse, so we can multiply by Q^T to obtain the equivalent upper triangular system $R\mathbf{x} = Q^T\mathbf{b}$. This system can then be solved efficiently for $\mathbf{x}$ using the method of back substitution.

For solving the linear least squares problem of minimizing $\|A\mathbf{x} - \mathbf{y}\|_2$, the QR decomposition provides a robust approach. The 2-norm is invariant under orthogonal transformations, so the problem is equivalent to minimizing $\|Q^T(QR\mathbf{x} - \mathbf{y})\|_2$, which simplifies to minimizing $\|R\mathbf{x} - Q^T\mathbf{y}\|_2$. We can partition the matrix R and the vector $Q^T\mathbf{y}$ as:

$$R = \begin{pmatrix} \hat{R} \\ 0 \end{pmatrix}, \quad Q^T\mathbf{y} = \begin{pmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{pmatrix} \tag{5}$$

where $\hat{R}$ is an $n \times n$ upper triangular matrix. The squared norm of the residual thus becomes $\|\hat{R}\mathbf{x} - \mathbf{y}_1\|_2^2 + \|\mathbf{y}_2\|_2^2$. This expression is minimized when the first term is zero, which is achieved by solving the square upper triangular system $\hat{R}\mathbf{x} = \mathbf{y}_1$ for $\mathbf{x}$ via back substitution.

3. Results and Analysis

The implemented Python scripts were executed to solve the three tasks, and the results were analyzed by comparing them with standard NumPy library functions.

For the first task of solving linear systems, the results were varied. The solver for Equation 1, the nearly tridiagonal system, produced a solution that differed from NumPy's result by a significant norm of 20.64. This large difference suggests numerical instability, possibly due to error accumulation in the custom QR implementation compared to the highly optimized LAPACK routines used by NumPy. In contrast, for the well-conditioned symmetric system in Equation 2, the QR solver performed excellently. Its solution was virtually identical to that of the Cholesky method, with a difference norm of approximately 1.88×10^{-15}, attributable to normal floating-point arithmetic. As anticipated, the solver struggled with the ill-conditioned Hilbert matrix of Equation 3. The resulting solution was highly erratic, and the difference norm compared to NumPy's solver was a massive 2499.37. This highlights the extreme sensitivity of ill-conditioned systems to small numerical errors, a challenge for both custom and standard solvers.

In the second task of polynomial curve fitting, the `qr_least_squares` solver proved to be accurate and effective. It determined the coefficients of the quadratic polynomial to be $a = 1.0$, $b = 1.0$, and $c = 1.0$, resulting in the function $y = t^2 + t + 1$. This result was identical to the one obtained using `numpy.linalg.lstsq`, indicating the correctness of our implementation for this well-conditioned problem.

The third task, predicting real estate prices, further demonstrated the robustness of the QR method for linear regression. The 12 model parameters calculated by our `qr_least_squares` function were identical to those from NumPy's benchmark function, with a negligible difference norm of 4.28×10^{-14}. This confirms that the QR approach provides a reliable and precise way to solve real-world data analysis problems.

4. Conclusion

This project successfully demonstrated the implementation and application of QR decomposition for solving square linear systems and linear least squares problems. The Householder QR method proved to be a stable and effective algorithm for well-conditioned problems, as evidenced by the high accuracy achieved in the polynomial fitting and real estate prediction tasks, where results matched those of standard optimized libraries. However, the

experiments also highlighted the inherent difficulties posed by ill-conditioned systems, where the limitations of standard floating-point arithmetic lead to significant inaccuracies in both our implementation and standard methods. Overall, QR decomposition stands out as a powerful and essential tool in numerical linear algebra, offering a reliable method for solving least squares problems, which are fundamental to data fitting and statistical modeling.