Assignment 2

LU Decomposition & Jacobi Iteration

Sahil Raj
2301CS41

20 August 2025

Contents

Problem 1: Solving System of Linear Equation with LU Decomposition

Problem Statement

The objective of this problem is to solve a system of linear equations

$$AX = B$$

using the LU decomposition method, where A is a square matrix and B is a known vector.

NOTE: The code can be accessed using this link: MATLAB, Julia.

Methodology

The LU decomposition technique factors a square matrix A into the product of a lower triangular matrix L and an upper triangular matrix U, such that:

$$A = LU$$

This decomposition enables solving the system $AX = B$ in two steps:

1. Forward substitution: Solve $LY = B$ for Y.
2. Backward substitution: Solve $UX = Y$ for X.

Pseudo-code

1. Initialize L and U as zero matrices of the same size as A.
2. For each column c and row r:
   - If $c \geq r$, compute entries of U.

     $$U[r, c] = A[r, c] - L[r, 1 : r - 1] \cdot U[1 : r - 1, c] \tag{1.1}$$

   - If $c < r$, compute entries of L.

     $$L[r, c] = A[r, c] - \frac{A[r, c] - L[r, 1 : c - 1] \cdot U[1 : c - 1, c]}{U[c, c]} \tag{1.2}$$

   - Set diagonal entries of L to 1.

3. Solve the system:

$$\begin{aligned} LY &= B \qquad\qquad \text{(forward substitution)} \\ Y[i] &= \frac{B[i] - L[i, 1:i-1] \cdot Y[1:i-1]}{L[i,i]} \end{aligned} \tag{1.3}$$

$$\begin{aligned} UX &= Y \qquad\qquad \text{(backward substitution)} \\ X[i] &= \frac{Y[i] - U[i, i+1:N] \cdot X[i+1:N]}{U[i,i]} \end{aligned} \tag{1.4}$$

Results

For the given system

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 4 & 3 & -1 \\ 3 & 5 & 3 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 6 \\ 4 \end{bmatrix},$$

the LU decomposition produces:

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 4 & 1 & 0 \\ 3 & 2 & 1 \end{bmatrix}, \quad U = \begin{bmatrix} 1 & 1 & 1 \\ 0 & -1 & -5 \\ 0 & 0 & -6 \end{bmatrix}.$$

Forward and backward substitution yield the solution vector:

$$X = \begin{bmatrix} 2.0000 \\ 3.0000 \\ -4.0000 \end{bmatrix}.$$

Conclusion

The LU decomposition method successfully solves the system of linear equations by breaking the problem into simpler triangular systems. This method is computationally efficient for multiple right-hand sides B, since the factorization of A is computed only once and can be reused.

Problem 2: Solving Problem of Classical Harmonic Oscillator with LU Decomposition

Problem Statement

The objective of this problem is to solve the classical harmonic oscillator equation numerically using LU decomposition. The oscillator follows the equation of motion:

$$m\frac{d^2x}{dt^2} + kx = 0,$$

where m is the mass and k is the spring constant. The method discretizes the second-order ODE and formulates a system of linear equations that is solved using LU decomposition.

NOTE: The code can be accessed using this link: MATLAB, Julia.

Methodology

The equation of motion is discretized using the finite-difference approximation:

$$x_{n+1} = \frac{k\,\Delta t^2}{m} x_n - 2x_n + x_{n-1},$$

where Δt is the time step. This recurrence relation is transformed into a tridiagonal linear system:

$$AX = B,$$

where A is constructed from the discretized coefficients, and B encodes initial conditions.

The solution proceeds as follows:

1. Construct the coefficient matrix A from the finite-difference scheme.
2. Form the right-hand side vector B using initial conditions x_0 and x_1.
3. Apply LU decomposition to factorize $A = LU$. See (1.1) and (1.2)
4. Solve the system by:
   (a) $LY = B$ (forward substitution) (1.3)
   (b) $UX = Y$ (backward substitution) (1.4)
5. Reconstruct the full displacement vector $Z(t)$ including the initial values.
6. Plot the solution $x(t)$ against t.

Results

For the chosen parameters:

$$m = 0.01, \quad k = 5.0, \quad \Delta t = 0.01, \quad t_{\text{max}} = 2, \quad x(0) = 0, \quad v(0) = 0.5,$$

the computed trajectory of the oscillator is shown in Figure 2.1.

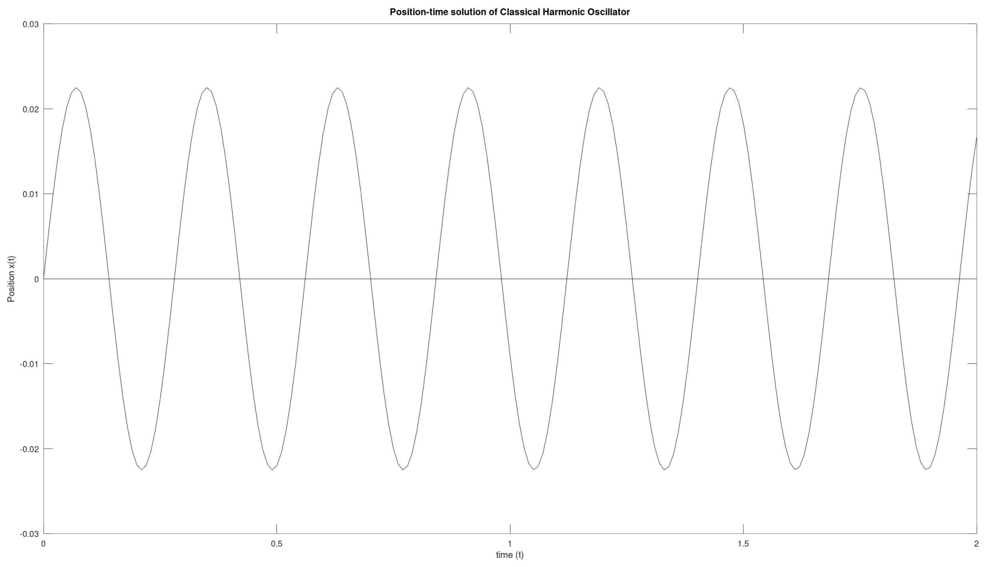

Figure 2.1: Numerical solution of the classical harmonic oscillator using LU decomposition.

The plot shows a sinusoidal oscillation with the expected period $T = 2\pi\sqrt{m/k}$, consistent with the analytical solution of the harmonic oscillator.

Conclusion

The LU decomposition method was successfully applied to solve the classical harmonic oscillator problem. By converting the discretized second-order ODE into a linear system, the problem reduces to triangular substitutions after decomposition. The numerical solution matches the expected sinusoidal oscillation, demonstrating the correctness of the method. This approach highlights the usefulness of LU decomposition in solving time-stepping problems arising in physics and engineering.

Problem 3: Solving Scrodinger Equation for a Infinite Well using the Cholesky Method

Problem Statement

The objective of this problem is to compute the eigenvalues and eigenfunctions of a quantum system by discretizing the Hamiltonian operator and solving the resulting eigenvalue problem. Specifically, the goal is to estimate the ground state energy of a particle in a one-dimensional infinite potential well using the **shifted inverse iteration method** with Cholesky decomposition.

NOTE: The code can be accessed using this link: MATLAB, Julia.

Methodology

The one-dimensional Hamiltonian for a free particle confined in a box of length a is given by

$$H = -\frac{\hbar^2}{2m}\frac{d^2}{dx^2},$$

subject to boundary conditions $x \in \left[-\frac{a}{2}, \frac{a}{2}\right]$ and $\psi(\pm a/2) = 0$.

Discretization

The spatial domain is discretized into N points with spacing $\Delta x = \frac{a}{N+1}$. The Hamiltonian matrix A is approximated by the finite-difference Laplacian:

$$A_{ij} = \begin{cases} \frac{\hbar^2}{m\Delta x^2}, & i = j, \\ -\frac{\hbar^2}{2m\Delta x^2}, & j = i \pm 1, \\ 0, & \text{otherwise.} \end{cases}$$

Shifted Inverse Iteration

To compute eigenvalues near a target σ, the matrix $A - \sigma I$ is factorized using Cholesky decomposition:

$$A - \sigma I = LL^\top,$$

where L is lower-triangular. Each iteration solves

$$(A - \sigma I)w = \psi^{(k)},$$

via forward and backward substitution. The vector is normalized:

$$\psi^{(k+1)} = \frac{w}{\|w\|}.$$

After K iterations, the approximate eigenvalue is obtained by the Rayleigh quotient:

$$\lambda \approx \psi^\top A \psi.$$

Results

For parameters

$$a = 1.0, \quad N = 100, \quad m = 1.0, \quad \hbar = 1.0, \quad \sigma = 40,$$

and $K = 1000$ iterations, the computed ground state energy is:

$$E \approx \lambda = \texttt{4.934404}.$$

Note: The value of m and $\hbar$ is taken as 1.0 here, to avoid the numerical roundoff, since they are very small value, they can be multiplied to the final solution during dimensional analysis to get the accurate result.
From the analytical expression of E_n for the solution,

$$E_n = \frac{n^2\pi^2\hbar^2}{2ma^2}$$

, if we substitute the value of m and $\hbar$ as 1.0, we get

$$E_1 = \frac{\pi^2}{2} \approx \texttt{4.934}$$

The normalized eigenfunction $\psi(x)$ corresponding to the lowest energy eigenvalue is plotted in Figure 3.1.

Conclusion

The shifted inverse iteration method successfully computes the ground state energy and eigenfunction of the discretized Hamiltonian. The method converges rapidly when the shift σ is chosen close to the target eigenvalue. This approach demonstrates the utility of iterative solvers combined with Cholesky decomposition in quantum eigenvalue problems, particularly for large sparse systems.

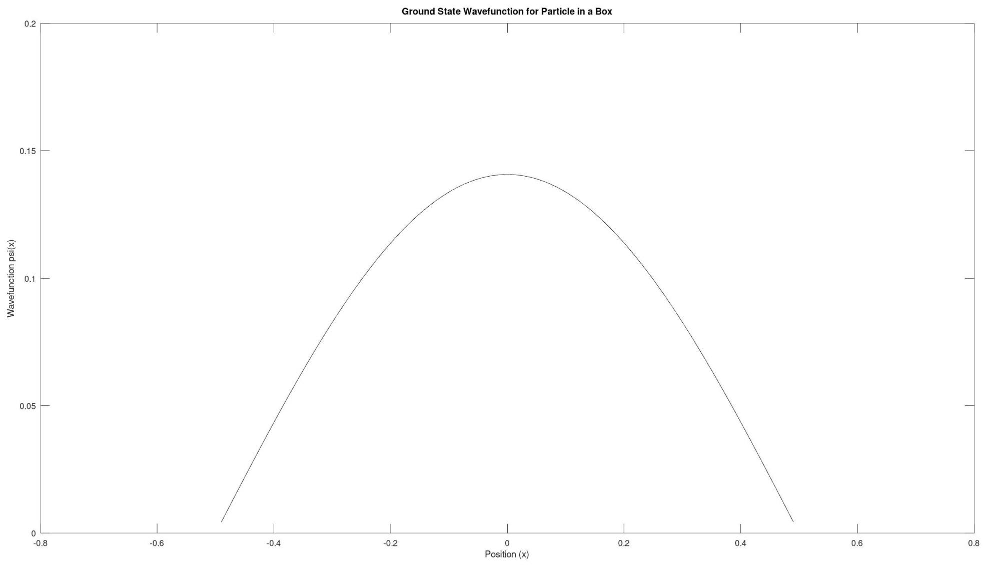

Figure 3.1: Normalized eigenfunction $\psi(x)$ obtained using shifted inverse iteration.

Problem 4: Solving the System of Linear Equation with Jacobi Iteration

Problem Statement

The objective of this problem is to solve a system of linear equations

$$AX = b$$

using the Jacobi Iteration method, which is an iterative approach to approximate the solution of a system of equations without direct matrix inversion.

NOTE: The code can be accessed using this link: MATLAB, Julia.

Methodology

The Jacobi method decomposes the coefficient matrix A into a diagonal part D and the remainder Q:

$$A = D - Q, \quad D = \operatorname{diag}(A).$$

The iteration scheme is then given by:

$$X^{(k+1)} = D^{-1}\big(b + QX^{(k)}\big).$$

The algorithm proceeds as:

1. Initialize with a random guess $X^{(0)}$.
2. For $k = 1, 2, \dots, K$:
$$X^{(k+1)} = D^{-1}(b + QX^{(k)})$$
3. Repeat until convergence or until the maximum number of iterations K is reached.

To assess convergence, the error norm is computed with respect to the exact solution $X_{\text{true}} = A^{-1}b$ after a varying number of iterations.

Results

For the given system:

$$A = \begin{bmatrix} 10 & 2 & 1 \\ 2 & 20 & -2 \\ -2 & 3 & 10 \end{bmatrix}, \quad b = \begin{bmatrix} 9 \\ -44 \\ 22 \end{bmatrix},$$

the Jacobi iteration converges to the true solution:

$$X = \begin{bmatrix} 1.0000 \\ -2.0000 \\ 3.0000 \end{bmatrix}.$$

The convergence behavior is shown in Figure 4.1. The error decreases as the number of iterations increases, demonstrating the iterative refinement of the solution.

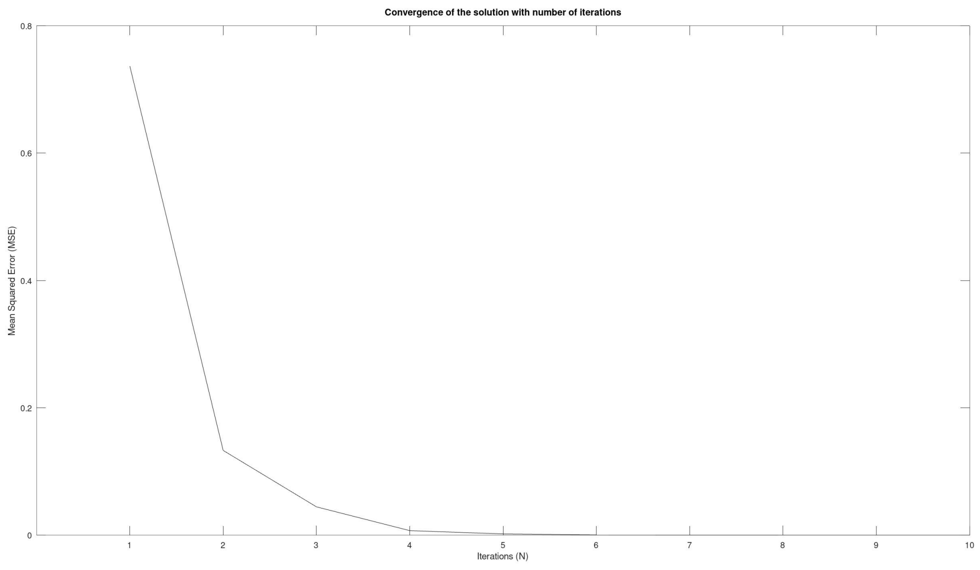

Figure 4.1: Convergence of the Jacobi iteration method. Error norm plotted against the number of iterations.

Conclusion

The Jacobi Iteration method successfully approximates the solution of the given linear system. The error decreases monotonically with iterations, illustrating convergence of the method. However, compared to direct methods like LU decomposition, Jacobi iteration requires more steps and may converge slowly depending on the properties of A. Its simplicity and parallelizability make it useful for large sparse systems.