Maymunah Hicks

Dr. Shannon Starr

Scientific Programming

4 November 2024

Homework 5

8. Use the Gauss elimination method to solve the following equations:

$$\begin{aligned} 3x_1 - x_2 + 4x_3 &= 2, \\ 17x_1 + 2x_2 + x_3 &= 14, \\ x_1 + 12x_2 - 7z &= 54. \end{aligned}$$

9. Use the Gauss–Jordan elimination method to solve the equations in Problem 8.

10. Obtain the lower triangular matrix L and upper triangular matrix U from the equations in Problem 8.

Final Answers

8. the final answer for the gauss elimination method is: [0.059016393442622515, 5.573770491803279, 1.8491803278688528]

9. the final answer for the gauss jordan method is: [0.059016393442622994, 5.573770491803279, 1.8491803278688526]

10. $L = \begin{bmatrix} 1 & 0 & 0 \\ 5.66667 & 1 & 0 \\ 0.33333 & 1.608695 & 1 \end{bmatrix}$ $\quad U = \begin{bmatrix} 3 & -1 & 4 & 2 \\ 0 & 7.66667 & -21.66667 & 2.6666 \\ 0 & 0 & 26.52173 & 49.043478 \end{bmatrix}$

Code for the Problems

Problem 8

```python
print("Section 14.7 Exercises")

print("Problem 8")
def gaussEliminationMethod(A, b):
    n = len(b)

    # forward elimination
    for i in range(n):
        for j in range(i+1, n):
            if A[i][i] == 0:
                raise ValueError("Error: Can not divide by zero.")
            factor = A[j][i] / A[i][i]
            for k in range(i, n):
                A[j][k] = A[j][k] - factor * A[i][k]
            b[j] = b[j] - factor * b[i]

    # back substitution
    x = [0 for _ in range(n)]
    for i in range(n-1, -1, -1):
        sum_ax = 0
        for j in range(i+1, n):
            sum_ax += A[i][j] * x[j]
        x[i] = (b[i] - sum_ax) / A[i][i]

    return x

A = [[3, -1, 4],
     [17, 2, 1],
     [1, 12, -7]]

b = [2, 14, 54]

answer = gaussEliminationMethod(A, b)
print("the final answer for the gauss elimination method is:", answer)
```

Problem 9

```python
print("Problem 9")
def gaussJordanMethod(A, b):
    n = len(b)

    # new augmented matrix A with the vector b
    newMatrix = [A[i] + [b[i]] for i in range(n)]

    # gauss-Jordan elimination
    for i in range(n):
        # make the diagonal element 1 by dividing the whole row
        diag = newMatrix[i][i]
        for k in range(len(newMatrix[i])):
            newMatrix[i][k] /= diag

        # make all elements above and below the pivot 0
        for j in range(n):
            if i != j:
                factor = newMatrix[j][i]
                for k in range(len(newMatrix[j])):
                    newMatrix[j][k] -= factor * newMatrix[i][k]

    # get the solution from the new augmented matrix
    solution = [newMatrix[i][-1] for i in range(n)]
    return solution

A = [[3, -1, 4],
     [17, 2, 1],
     [1, 12, -7]]

b = [2, 14, 54]

answer = gaussJordanMethod(A, b)
print("the final answer for the gauss jordan method is:", answer)
```

Problem 10

```
print("Problem 10")
def triangleMatrices(A):
    n = A.shape[0]  # number of rows (or columns since A is square)
    L = np.zeros_like(A, dtype=float)  # lower triangular matrix
    U = np.zeros_like(A, dtype=float)  # upper triangular matrix

    for i in range(n):
        # calculating the Upper triangular matrix U
        for k in range(i, n):
            sum_u = sum(L[i][j] * U[j][k] for j in range(i))  # sum for U
            U[i][k] = A[i][k] - sum_u  # find U element

        # calculating the lower triangular matrix L
        for k in range(i, n):
            if i == k:
                L[i][i] = 1  # diagonal elements of L are 1
            else:
                sum_l = sum(L[k][j] * U[j][i] for j in range(i))  # sum for L
                L[k][i] = (A[k][i] - sum_l) / U[i][i]  # find L element

    return L, U

A = np.array([[3, -1, 4],
              [17, 2, 1],
              [1, 12, -7]], dtype=float)

s
L, U = triangleMatrices(A)


print("The lower triangular matrix L is:")
print(L)

print("\nThe upper triangular matrix U is:")
print(U)
```