pr2_math548_final_Lazizbek

May 2, 2024

1 Math 548, Final Exam. Problem 1. LS

(a)

For which values of b, the matrix A is positive definite? (answer is on paper)

```
     [ 1    -1/2  b]

A =  [-1/2  2 -1/2]

     [ b  -1/2  2 ]
```

(b) Find the LU decomposition of this matrix when b=0, using the Gaussian elimination method. Write down the matrices L and U. Explain and justify every step in the process.

1.

```
[ 1    -1/2  b][x1]    [ 0]

[-1/2  2 -1/2][x2] = [ 0]

[ b  -1/2  2 ][x3]    [ 0]
```

(c)

Now solve the system of equations given by

2.

```
[ 1    -1/2  b][x1]    [ 2]

[-1/2  2 -1/2][x2] = [ 3]

[ b  -1/2  2 ][x3]    [14]
```

2 (b). I am introducing b = [0, 0, 0] RHS vector to perform LU decompositon on the matrix A

[1]:
```python
import numpy as np

def lu_decomposition(A):
    # Function to perform LU decomposition of matrix A
    n = len(A)
    L = np.zeros((n, n))  # Initialize lower triangular matrix L
    U = np.zeros((n, n))  # Initialize upper triangular matrix U
    P = np.identity(n)    # Initialize permutation matrix P

    # Main loop for LU decomposition with partial pivoting
    for i in range(n):
        # Partial pivoting: find pivot row with maximum absolute value
        # pivot_row = max(range(i, n), key=lambda k: abs(A[k][i]))

        pivot_row_max_value = -1  # Initialize with a minimum value
        for k in range(i, n):  # Iterate through rows from i to n-1
            absolute_value = abs(A[k][i])  # Compute the absolute value of␣
 ↪element at column i and row k
            if absolute_value > pivot_row_max_value:  # Check if the absolute␣
 ↪value is greater than current maximum
                pivot_row_max_value = absolute_value  # Update maximum absolute␣
 ↪value
                pivot_row = k  # Update the row index with the maximum absolute␣
 ↪value

        # Swap current row with pivot row in A, P
        A[[i, pivot_row]] = A[[pivot_row, i]]
        P[[i, pivot_row]] = P[[pivot_row, i]]

        # Set diagonal of L to 1
        L[i][i] = 1
        # Calculate entries of U and L using formulae for LU decomposition
        for j in range(i, n):
            U[i][j] = A[i][j] - sum(L[i][k] * U[k][j] for k in range(i))
        for j in range(i + 1, n):
            L[j][i] = (A[j][i] - sum(L[j][k] * U[k][i] for k in range(i))) /␣
 ↪U[i][i]

    return P, L, U

def forward_substitution(L, b):
    # Function to perform forward substitution to solve Ly = b
    n = len(b)
```

```python
    y = np.zeros(n)  # Initialize solution vector y
    for i in range(n):
        # Compute y[i] using forward substitution formula
        y[i] = b[i] - np.dot(L[i, :i], y[:i])
    return y

def back_substitution(U, y):
    # Function to perform back substitution to solve Ux = y
    n = len(y)
    x = np.zeros(n)  # Initialize solution vector x
    for i in range(n - 1, -1, -1):
        # Compute x[i] using back substitution formula
        x[i] = (y[i] - np.dot(U[i, i+1:], x[i+1:])) / U[i, i]
    return x

# Given matrix A and vector b
A = np.array([[ 2, -1/2,  0], [-1/2,  2, -1/2], [ 0, -1/2,  2]])
b = np.array([ 0,  0,  0])

# Keep the original A and b
original_A = A
original_b = b

# Perform LU decomposition of A
P, L, U = lu_decomposition(A)

# Permute b according to pivot matrix P
b_permuted = np.dot(P, b)

# Solve Ly = b_permuted using forward substitution
y = forward_substitution(L, b_permuted)

# Solve Ux = y using back substitution
x = back_substitution(U, y)

# Verify if A = LU
is_A_LU = np.allclose(original_A, np.dot(L, U))

# Calculate the determinant of A
det_A = np.linalg.det(original_A)

# Construct the augmented matrix [A|b]
augmented_matrix = np.hstack((original_A, np.expand_dims(original_b, axis=1)))

# Print all intermediate and final results
print("\nAugmented matrix [A|b]:")
print(augmented_matrix)
```

```python
print("\nSolution vector x:")
print(x)
print("\nPermutation matrix P:")
print(P)
print("\nLower triangular matrix L:")
print(L)
print("\nUpper triangular matrix U:")
print(U)
print("\nIs A equal to LU:", is_A_LU)
print("\nDeterminant of A:", det_A)
```

```
Augmented matrix [A|b]:
[[ 2.  -0.5  0.   0. ]
 [-0.5  2.  -0.5  0. ]
 [ 0.  -0.5  2.   0. ]]

Solution vector x:
[0. 0. 0.]

Permutation matrix P:
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]

Lower triangular matrix L:
[[ 1.          0.          0.        ]
 [-0.25        1.          0.        ]
 [ 0.         -0.26666667  1.        ]]

Upper triangular matrix U:
[[ 2.         -0.5         0.        ]
 [ 0.          1.875      -0.5       ]
 [ 0.          0.          1.86666667]]

Is A equal to LU: True

Determinant of A: 6.999999999999998
```

3 (c) Now solve the system of equations given by

[2]:
```python
import numpy as np

def lu_decomposition(A):
    # Function to perform LU decomposition of matrix A
    n = len(A)
```

```
    L = np.zeros((n, n))  # Initialize lower triangular matrix L
    U = np.zeros((n, n))  # Initialize upper triangular matrix U
    P = np.identity(n)    # Initialize permutation matrix P

    # Main loop for LU decomposition with partial pivoting
    for i in range(n):
        # Partial pivoting: find pivot row with maximum absolute value
        # pivot_row = max(range(i, n), key=lambda k: abs(A[k][i]))

        pivot_row_max_value = -1  # Initialize with a minimum value
        for k in range(i, n):  # Iterate through rows from i to n-1
            absolute_value = abs(A[k][i])  # Compute the absolute value of
 ↪element at column i and row k
            if absolute_value > pivot_row_max_value:  # Check if the absolute
 ↪value is greater than current maximum
                pivot_row_max_value = absolute_value  # Update maximum absolute
 ↪value
                pivot_row = k  # Update the row index with the maximum absolute
 ↪value

        # Swap current row with pivot row in A, P
        A[[i, pivot_row]] = A[[pivot_row, i]]
        P[[i, pivot_row]] = P[[pivot_row, i]]

        # Set diagonal of L to 1
        L[i][i] = 1
        # Calculate entries of U and L using formulae for LU decomposition
        for j in range(i, n):
            U[i][j] = A[i][j] - sum(L[i][k] * U[k][j] for k in range(i))
        for j in range(i + 1, n):
            L[j][i] = (A[j][i] - sum(L[j][k] * U[k][i] for k in range(i))) /
 ↪U[i][i]

    return P, L, U

def forward_substitution(L, b):
    # Function to perform forward substitution to solve Ly = b
    n = len(b)
    y = np.zeros(n)  # Initialize solution vector y
    for i in range(n):
        # Compute y[i] using forward substitution formula
        y[i] = b[i] - np.dot(L[i, :i], y[:i])
    return y

def back_substitution(U, y):
    # Function to perform back substitution to solve Ux = y
    n = len(y)
```

```
    x = np.zeros(n)  # Initialize solution vector x
    for i in range(n - 1, -1, -1):
        # Compute x[i] using back substitution formula
        x[i] = (y[i] - np.dot(U[i, i+1:], x[i+1:])) / U[i, i]
    return x

# Given matrix A and vector b
A = np.array([[ 2, -1/2,  0], [-1/2,  2, -1/2], [ 0, -1/2,  2]])
b = np.array([ 2,  3,  14])

# Keep the original A and b
original_A = A
original_b = b

# Perform LU decomposition of A
P, L, U = lu_decomposition(A)

# Permute b according to pivot matrix P
b_permuted = np.dot(P, b)

# Solve Ly = b_permuted using forward substitution
y = forward_substitution(L, b_permuted)

# Solve Ux = y using back substitution
x = back_substitution(U, y)

# Verify if A = LU
is_A_LU = np.allclose(original_A, np.dot(L, U))

# Calculate the determinant of A
det_A = np.linalg.det(original_A)

# Construct the augmented matrix [A|b]
augmented_matrix = np.hstack((original_A, np.expand_dims(original_b, axis=1)))

# Print all intermediate and final results
print("\nAugmented matrix [A|b]:")
print(augmented_matrix)
print("\nSolution vector x:")
print(x)
print("\nPermutation matrix P:")
print(P)
print("\nLower triangular matrix L:")
print(L)
print("\nUpper triangular matrix U:")
print(U)
print("\nIs A equal to LU:", is_A_LU)
```

```python
print("\nDeterminant of A:", det_A)
```

```
Augmented matrix [A|b]:
[[ 2.   -0.5  0.    2. ]
 [-0.5  2.   -0.5  3. ]
 [ 0.   -0.5  2.   14. ]]

Solution vector x:
[2. 4. 8.]

Permutation matrix P:
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]

Lower triangular matrix L:
[[ 1.          0.          0.        ]
 [-0.25        1.          0.        ]
 [ 0.         -0.26666667  1.        ]]

Upper triangular matrix U:
[[ 2.         -0.5         0.        ]
 [ 0.          1.875      -0.5       ]
 [ 0.          0.          1.86666667]]

Is A equal to LU: True

Determinant of A: 6.999999999999998
```

[4]:
```python
# !sudo apt-get install texlive-xetex texlive-fonts-recommended
↪texlive-plain-generic
```

[5]:
```python
# !jupyter nbconvert --to pdf /content/Math548_hw7_Lazizbek.ipynb
```