NAME:- KETAN DILIP ATTARDE

REG NO:- 24-27-06

MTECH DATA SCIENCE

Gauss-Jordan Elimination Algorithm

Input

- Coefficient matrix A of size $n \times n$
- Constant vector b of size n

Output

- Solution vector x of size n

Steps

1. **Combine matrices:**
    - Create an augmented matrix by combining the coefficient matrix A and the constant vector b.

$$\{\text{augmented\_matrix}\} = [A \,|\, b]$$

2. **Partial Pivoting:**
    - For each column k from 0 to $n-1$:
        - Find the pivot (maximum element in the current column) in the submatrix below and including row k.
        - Swap rows to move the pivot element to the current row.

3. **Gauss-Jordan Elimination:**
    - For each row k from 0 to $n-1$:
        - Make the pivot element $A[k, k]$ equal to 1 by dividing the entire row by $A[k, k]$.
        - Eliminate entries above and below the pivot by performing row operations.
4. **Extract Solution:**
    - The solution vector x is the last column of the augmented matrix.

$$x = \{\text{augmented\_matrix}\}[:, -1]$$

```
import numpy as np


def gauss_jordan_elimination(A, b):
    # Combine the matrix A and vector b into one augmented matrix
    augmented_matrix = np.column_stack((A, b))

    n = len(b)

    for k in range(n):
        # Partial pivoting: find the pivot (maximum element in the current column)
        pivot_index = np.argmax(np.abs(augmented_matrix[k:, k])) + k

        # Swap rows to move the pivot element to the current row
        augmented_matrix[[k, pivot_index]] = augmented_matrix[[pivot_index, k]]

        # Make the pivot element 1
        augmented_matrix[k, :] /= augmented_matrix[k, k]

        # Eliminate entries above and below the pivot
        for i in range(n):
            if i != k:
                factor = augmented_matrix[i, k]
```

```
                factor = augmented_matrix[i, k]
                augmented_matrix[i, :] -= factor * augmented_matrix[k, :]

    # Extract the solution
    solution = augmented_matrix[:, -1]

    return solution


# Example usage
A = np.array([[2, 1, -1],
              [-3, -1, 2],
              [-2, 1, 2]]).astype(np.float64)

b = np.array([8, -11, -3]).astype(np.float64)

print("The augmented matrix is:")
print(np.column_stack((A, b)))
print("-" * 50)

solution = gauss_jordan_elimination(A, b)
print("Solution:", solution)
```

```
The augmented matrix is:
[[  2.   1.  -1.   8.]
 [ -3.  -1.   2. -11.]
 [ -2.   1.   2.  -3.]]
--------------------------------------------------
Solution: [ 2.  3. -1.]
```