system resolution

December 1, 2024

[1]:
```
import numpy as np

# Définir la matrice des coefficients (A) et le vecteur des constantes (b)
A = np.array([[2, 1], [1, -1]])
b = np.array([3, -1])

# Résoudre le système A x = b
x = np.linalg.solve(A, b)

print("Solution : ", x)
```

```
Solution :  [0.66666667 1.66666667]
```

[3]:
```
def gauss_jordan_elimination(A, b):
    n = len(b)
    # Combiner A et b pour former une matrice augmentée
    augmented_matrix = np.hstack([A, b.reshape(-1, 1)])

    for i in range(n):
        # Normaliser la ligne actuelle
        augmented_matrix[i] = augmented_matrix[i] / augmented_matrix[i, i]

        # Éliminer les autres coefficients dans la colonne actuelle
        for j in range(n):
            if i != j:
                factor = augmented_matrix[j, i]
                augmented_matrix[j] = augmented_matrix[j] - factor *␣
 ↪augmented_matrix[i]

    # Extraire le vecteur solution
    return augmented_matrix[:, -1]

# Exemple
A = np.array([[2, 1], [1, -1]], dtype=float)
b = np.array([3, -1], dtype=float)

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

```
Solution :  [0.66666667 1.66666667]
```

[5]:
```python
from scipy.linalg import lu_factor, lu_solve

# Définir la matrice et le vecteur
A = np.array([[2, 1], [1, -1]], dtype=float)
b = np.array([3, -1], dtype=float)

# Factorisation LU
lu, piv = lu_factor(A)

# Résoudre le système
x = lu_solve((lu, piv), b)

print("Solution : ", x)
```

```
Solution :  [0.66666667 1.66666667]
```