Ting Fung Lam 2924629375

MATH 501 PA5

Both methods give same solution

Output:

Reduced matrix A without pivot

[[ 0.2641 0.1735 0.8642 ]

[ 3.56342295 -0.60075388 -2.93321011]

[-3.27186672 -0.23864827 2.19864169]]

Solution without pivot, x = [ 0.81475769 -2.35698393 -0.64607822]

Reduced matrix A with pivot

[[ 0.28062905 -0.41297365 0.90798109]

[ 0.9411 0.0175 0.1463 ]

[-0.91818085 -0.40823184 0.20542986]]

Solution with pivot, x = [ 0.81475769 -2.35698393 -0.64607822]

Check Ax: [-0.7521 0.631 0.2501]

import numpy as np  
  
  
def gaussian(A):  
 *"""  
 Gaussian elimination without pivoting* ***:param*** *A: Input matrix A* ***:return****: Reduced matrix A with ratios overwritten to the zeros  
 """* n = np.size(A, 0)  
 for k in range(0, n - 1):  
 for i in range(k + 1, n):  
 z = A[i, k] / A[k, k]  
 A[i, k] = z  
 for j in range(k + 1, n):  
 A[i, j] = A[i, j] - z \* A[k, j]  
 return A  
  
  
def gaussianPivot(A):  
 *"""  
 Gaussian elimination with pivoting* ***:param*** *A: Input matrix A* ***:return****: Reduced matrix A with ratios overwritten to the zeros  
 """* n = np.size(A, 0)  
 p = np.zeros(n)  
 s = np.zeros(n)  
 for i in range(0, n):  
 p[i] = i  
 p = p.astype(int)  
 s[i] = np.max(abs(A[i, 0:n]))  
 for k in range(0, n - 1):  
 temp = np.array([abs(A[p[i], k]) / s[p[i]] for i in range(k, n)])  
 j = np.argmax(temp) + k  
 p[k], p[j] = p[j], p[k]  
 for i in range(k + 1, n):  
 z = A[p[i], k] / A[p[k], k]  
 A[p[i], k] = z  
 for j2 in range(k + 1, n):  
 A[p[i], j2] = A[p[i], j2] - z \* A[p[k], j2]  
 return A, p  
  
  
def solution(A, p, b):  
 *"""  
 Solution phase* ***:param*** *A: Reduced matrix A with ratios overwritten on zeros* ***:param*** *p: Permutation* ***:param*** *b: Vector b* ***:return****: Solution vector x  
 """* n = np.size(A, 0)  
 x = np.zeros(n)  
 for k in range(0, n - 1):  
 for i in range(k + 1, n):  
 b[p[i]] = b[p[i]] - A[p[i], k] \* b[p[k]]  
 for i in reversed(range(0, n)):  
 x[i] = (b[p[i]] - np.dot(A[p[i], i:n], x[i: n])) / A[p[i], i]  
 return x  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 A = np.array([[0.2641, 0.1735, 0.8642], [0.9411, 0.0175, 0.1463], [-0.8641, -0.4243, 0.0711]])  
 b = np.array([-0.7521, 0.6310, 0.2501])  
 p = np.array([0, 1, 2])  
  
 A = gaussian(A)  
 x = solution(A, p, b)  
 print("Reduced matrix A without pivot")  
 print(A)  
 print("Solution without pivot, x = " + str(x))  
  
  
 A = np.array([[0.2641, 0.1735, 0.8642], [0.9411, 0.0175, 0.1463], [-0.8641, -0.4243, 0.0711]])  
 b = np.array([-0.7521, 0.6310, 0.2501])  
 A, p = gaussianPivot(A)  
 x = solution(A, p, b)  
 print("Reduced matrix A with pivot")  
 print(A)  
 print("Solution with pivot, x = " + str(x))  
 A = np.array([[0.2641, 0.1735, 0.8642], [0.9411, 0.0175, 0.1463], [-0.8641, -0.4243, 0.0711]])  
 b = np.array([-0.7521, 0.6310, 0.2501])  
 print("Check Ax: " + str(np.dot(A, x)))