Pregunta 12

April 25, 2019

Autor:
Lazaro Camasca Edson
Analisis Numerico
Universidad Nacional de Ingenieria #### Programe un procedimiento que realice la factorización LU con pivoteo parcial de una matriz A. Aplíquelo a la matriz del problema 10

```
In [1]: from toolNick import *
        import numpy as np
```

```
toolNick se ha importado correctamente.
```

0.0.1 Factorizacion de LU con pivote

```
In [3]: def elim_LU(a, b, p = False, v = False):
            #v nos muestra el procedimiento detallado de la eliminacion
            #p se utiliza si requiere pivotacion total al inicio

            A_b = np.c_[a, b] # Matriz aumentada
            (fil, col) = A_b.shape #Guarda el #filas y #columnas
            line = "============================================="
            print("Matriz aumentada [A|b] al inicio:\n{}\n{}".format(A_b,line))

            if p:
                A_b = pivoteo_Total(A_b, v)

            A = A_b[:, :col - 1] #A sera todo menos la ultima columna
            b = A_b[:, col - 1] #b sera solo la ultima columna

            #Creamos una lista para almacenar las matrices L y P
            L_list = []
            P_list = []  #Matrices de permutacion

            for i in range(fil): # se condiera dim(a)
                #PIVOTEO
                P = pivoteo(A, i)

                if v : print("P_{}:\n{}".format(i+1,P))
```

```
            A = P @ A
            P_list.append(P)

            #Obtenemos el L_i
            L_i = get_L(A, i, v)
            L_list.append(L_i)

            A = L_i @ A

            #mostrar L(i) * (A)
            if v : print("L_{} * A:\n{}\n{}".format(i+1,A,line))


        U = A
        L,P = get_L_and_P_LU(L_list, P_list)
        print("L_:\n{}\nU_:\n{}\nP_:\n{}".format(L,U,P))

        Pb = P @ b
        #Tenemos L, U, P, b

        # Ax = b  &  PA = LU
        # PAx = Pb -> LUx = Pb
        # Ly = Pb  hallamos y  ->  Ux = y  hallamos x

        y = resolverMTriangularInf(L, Pb)
        x = resolverMTriangularSup(U, y)
        return x
```

Definimos A y b

```
In [4]: A = np.array([[2, 1, 1, 0],
                      [4, 3, 3, 1],
                      [8, 7, 9, 5],
                      [6, 7, 9, 8]])

        print("Matriz de coeficientes A:\n", A); (fil,col) = A.shape
        b = np.array([1, 8, 30, 41]); b.reshape(fil,1)
        print("Matriz b: \n", b)
```

```
Matriz de coeficientes A:
 [[2 1 1 0]
 [4 3 3 1]
 [8 7 9 5]
 [6 7 9 8]]
Matriz b:
 [ 1  8 30 41]
```

Resolvemos el sistema

```
In [5]: elim_LU(A,b,v=True)
```

```
Matriz aumentada [A|b] al inicio:
[[ 2  1  1  0  1]
 [ 4  3  3  1  8]
 [ 8  7  9  5 30]
 [ 6  7  9  8 41]]
==========================================
P_1:
[[0. 0. 1. 0.]
 [0. 1. 0. 0.]
 [1. 0. 0. 0.]
 [0. 0. 0. 1.]]
alpha_1:
[[0.  ]
 [0.5 ]
 [0.25]
 [0.75]]
alpha_1 * e_1:
[[0.   0.   0.   0.  ]
 [0.5  0.   0.   0.  ]
 [0.25 0.   0.   0.  ]
 [0.75 0.   0.   0.  ]]
L_1:
[[ 1.    0.    0.    0.  ]
 [-0.5   1.    0.    0.  ]
 [-0.25  0.    1.    0.  ]
 [-0.75  0.    0.    1.  ]]
L_1 * A:
[[ 8.    7.    9.    5.  ]
 [ 0.   -0.5  -1.5  -1.5 ]
 [ 0.   -0.75 -1.25 -1.25]
 [ 0.    1.75  2.25  4.25]]
==========================================
P_2:
[[1. 0. 0. 0.]
 [0. 0. 0. 1.]
 [0. 0. 1. 0.]
 [0. 1. 0. 0.]]
alpha_2:
[[ 0.        ]
 [ 0.        ]
 [-0.42857143]
 [-0.28571429]]
alpha_2 * e_2:
[[ 0.          0.          0.          0.        ]
 [ 0.          0.          0.          0.        ]
```

```
 [-0.         -0.42857143 -0.         -0.        ]
 [-0.         -0.28571429 -0.         -0.        ]]
L_2:
[[1.         0.         0.         0.        ]
 [0.         1.         0.         0.        ]
 [0.         0.42857143 1.         0.        ]
 [0.         0.28571429 0.         1.        ]]
L_2 * A:
[[ 8.          7.          9.          5.        ]
 [ 0.          1.75        2.25        4.25      ]
 [ 0.          0.         -0.28571429  0.57142857]
 [ 0.          0.         -0.85714286 -0.28571429]]
============================================
P_3:
[[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 0. 1.]
 [0. 0. 1. 0.]]
alpha_3:
[[0.        ]
 [0.        ]
 [0.        ]
 [0.33333333]]
alpha_3 * e_3:
[[0.         0.         0.         0.        ]
 [0.         0.         0.         0.        ]
 [0.         0.         0.         0.        ]
 [0.         0.         0.33333333 0.        ]]
L_3:
[[ 1.          0.          0.          0.        ]
 [ 0.          1.          0.          0.        ]
 [ 0.          0.          1.          0.        ]
 [ 0.          0.         -0.33333333  1.        ]]
L_3 * A:
[[ 8.          7.          9.          5.        ]
 [ 0.          1.75        2.25        4.25      ]
 [ 0.          0.         -0.85714286 -0.28571429]
 [ 0.          0.          0.          0.66666667]]
============================================
P_4:
[[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]
alpha_4:
[[0.]
 [0.]
 [0.]
```

```
 [0.]]
alpha_4 * e_4:
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
L_4:
[[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]
L_4 * A:
[[ 8.          7.          9.          5.        ]
 [ 0.          1.75        2.25        4.25      ]
 [ 0.          0.         -0.85714286 -0.28571429]
 [ 0.          0.          0.          0.66666667]]
============================================
L_:
[[ 1.          0.          0.          0.        ]
 [ 0.75        1.          0.          0.        ]
 [ 0.5        -0.28571429  1.          0.        ]
 [ 0.25       -0.42857143  0.33333333  1.        ]]
U_:
[[ 8.          7.          9.          5.        ]
 [ 0.          1.75        2.25        4.25      ]
 [ 0.          0.         -0.85714286 -0.28571429]
 [ 0.          0.          0.          0.66666667]]
P_:
[[0. 0. 1. 0.]
 [0. 0. 0. 1.]
 [0. 1. 0. 0.]
 [1. 0. 0. 0.]]
```

Out[5]: array([-1., 2., 1., 3.])