mult_matrices_eliminacion_gaussiana

March 1, 2019

0.1 Ejercicio 1: utilizar productos matriciales o vectoriales para las versiones que revisamos la clase pasada (ijk, kij, ikj) de la multiplicación de matrices de modo que los 2 loops mas internos utilicen estos productos

Inventamos como ejemplo dos matrices A y B. Vemos el resultado de la multiplicación:

```
In [336]: import numpy as np
          A=np.matrix([[4,-1,2],[21,-2,7],[15,32,6],[9,8,7]])
          B=np.matrix([[3,5,8],[6,9,1],[15,2,3]])
          A*B
```

```
Out[336]: matrix([[ 36,  15,  37],
                  [156, 101, 187],
                  [327, 375, 170],
                  [180, 131, 101]])
```

algoritmo ijk:

```
In [337]: m=A.shape[0]
          n=B.shape[1]
          C=np.zeros((m,n))
          for i in range(0,m):
                  C[i,:]=A[i,:]*B

          C
```

```
Out[337]: array([[ 36.,  15.,  37.],
                 [156., 101., 187.],
                 [327., 375., 170.],
                 [180., 131., 101.]])
```

algoritmo ikj queda igual que el anterior:

```
In [338]: m=A.shape[0]
          n=B.shape[1]
          C=np.zeros((m,n))
          for i in range(0,m):
                 C[i,:]=A[i,:]*B

          C
```

```
Out[338]: array([[ 36.,  15.,  37.],
                 [156., 101., 187.],
                 [327., 375., 170.],
                 [180., 131., 101.]])
```

algoritmo kij:

```
In [339]: m=A.shape[0]
          n=B.shape[1]
          r=B.shape[0]
          C=np.zeros((m,n))
          for k in range(0,r):
              C= C + A[:,k]*B[k,:]

          C
```

```
Out[339]: matrix([[ 36.,  15.,  37.],
                  [156., 101., 187.],
                  [327., 375., 170.],
                  [180., 131., 101.]])
```

0.2 Ejercicio 2: Eliminacion gaussiana hacia atrás

```
In [340]: A=np.array([[4,-1,2,3],[0,-2,7,-4],[0,0,6,5],[0,0,0,3]])
          n=A.shape[0]
          b=np.array([[20],[-7],[4],[6]])
          print(A)
          print(b)
```

```
[[ 4 -1  2  3]
 [ 0 -2  7 -4]
 [ 0  0  6  5]
 [ 0  0  0  3]]
[[20]
 [-7]
 [ 4]
 [ 6]]
```

```
In [341]: x=np.zeros((n,1))
          x[n-1]=b[n-1,0]/A[n-1,n-1]
          print(x)
```

```
[[0.]
 [0.]
 [0.]
 [2.]]
```

```
In [342]: for i in np.linspace(n-2,0,n-1,dtype=int):
              print(i)

              x[i]=(b[i]-np.dot(A[i,i+1:n],x[i+1:n]))/A[i,i]

          x
```

```
2
1
0
```

```
Out[342]: array([[ 3.],
                 [-4.],
                 [-1.],
                 [ 2.]])
```

0.3 Ejercicio 3: Eliminacion gaussiana hacia delante

```
In [343]: A=np.array([[3,0,0,0],[-1,1,0,0],[3,-2,-1,0],[1,-2,6,2]])
          b=np.array([[5],[6],[4],[2]])
          print(A)
          print(b)
```

```
[[ 3  0  0  0]
 [-1  1  0  0]
 [ 3 -2 -1  0]
 [ 1 -2  6  2]]
[[5]
 [6]
 [4]
 [2]]
```

```
In [344]: n=A.shape[0]
```

```
In [345]: x=np.zeros((4,1))
          x[0]=b[0]/A[0,0]
          x
```

```
Out[345]: array([[1.66666667],
                 [0.        ],
                 [0.        ],
                 [0.        ]])
```

```
In [346]: for i in range(1,n):
              x[i]=(b[i]-np.dot(A[i,0:i],x[0:i,0]))/A[i,i]
          x

Out[346]: array([[  1.66666667],
                 [  7.66666667],
                 [-14.33333333],
                 [ 50.83333333]])
```