determinant

September 9, 2020

[1]:
```
# Python code: determinants

import numpy as np


# Elementary row operations

def Rswitch(matrix,i,j):
    r = matrix[i-1].copy()
    matrix[i-1] = matrix[j-1]
    matrix[j-1] = r
    return(matrix)
def Rmult(matrix,i,c):
    matrix[i-1]=matrix[i-1]*c
    return(matrix)

def Radd(matrix,i,j,c):
    matrix[j-1]=matrix[j-1] + c*matrix[i-1]
    return(matrix)


# Determinant using elementary row operations

def determinant(matrix):
    # do some checks
    if matrix.dtype == np.int32:
        print("Must be a floating point matrix")
        print("Use: determinant(X.astype(np.float64))")
        return(-1)
    if matrix.ndim != 2:
        print("Wrong dimensions")
        return(-1)
    if matrix.shape[0] != matrix.shape[1]:
        print("Must be a square matrix")
        return(-1)
    # check if the matrix is 1x1
    if matrix.shape[0] == 1:
```

```
        return(matrix[0,0])
    # get the leftmost column, and nonzero positions
    lcol = matrix[:,0]
    nz = np.arange(lcol.size)[lcol != 0]
    # when zero column, the determinant is zero
    if nz.size == 0:
        return(0)
    # find first non-zero entry in column, switch if
    # this is not the first position
    p=nz[0]
    if p!=0:
        Rswitch(matrix,1,p+1)
    # use elementary row operations to get zeros
    # under the pivot
    for r in range(1,lcol.size):
        Radd(matrix,1,r+1,-matrix[r,0]/matrix[0,0])
    # use cofactor expansion along the first column
    return(matrix[0,0]*determinant(matrix[1:,1:]))
```

[2]:
```
# Some tests that you can run

A = np.array([[1,1,1],[1,2,4],[1,3,9]])
B = np.random.randn(8,8)
C = np.array([[4,0,0,-1,-1],[0,2,0,1,-1],[0,0,6,-2,0],[1,-1,2,0,0],[1,1,0,0,0]])

np.set_printoptions(precision=4,suppress=True)
```

[3]:
```
print(A)
determinant(A.astype(np.float64))
```

```
[[1 1 1]
 [1 2 4]
 [1 3 9]]
```

[3]:
```
2.0
```

[4]:
```
print(B)
determinant(B)
```

```
[[ 0.6343 -0.9404 -2.0947  0.2662 -0.6739 -0.6186  1.1472 -0.1044]
 [ 1.2877 -2.0245  0.2309 -0.5788  0.3408  0.7287 -0.8601 -0.3883]
 [-0.7334  0.2416  0.6431  1.6768 -2.022   1.3332  0.6204  1.3405]
 [-0.7807  1.6244  0.1482  0.5256 -0.198  -1.6843 -0.3431  0.3314]
 [ 1.4285  0.8762  0.1391  0.6681  1.1516  1.0366 -0.3513 -0.3359]
 [-1.5389  0.4512 -0.5369  0.4095 -0.6106  0.8517  1.8711  0.1592]
 [ 0.3323  1.1115  2.3079 -0.5566  1.5445  0.1347  0.0486  0.4623]
 [-0.5507  0.3374  0.3395 -0.9697 -0.6754  0.8695  0.1655 -1.7343]]
```

[4]: 87.21729256083391

[5]:
```
print(C)
determinant(C.astype(np.float64))
```

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

[5]: 48.0

[ ]: