DS 03

```
# -*- coding: utf-8 -*-
"""
ÃĽditeur de Spyder

Ceci est un script temporaire.
"""

import numpy as np
import matplotlib.pyplot as plt

def triangle(A,b):
    n =len(b)
    x = [0 for i in range(n)]
    x[n-1] = b[n-1]/A[n-1][n-1]
    for i in range(n-2,-1,-1):
        s = 0
        for j in range(i+1,n-1):
            s = s + A[i][j]*x[j]
        x[i] = (b[i] - s)/ A[i][i]
    return x

def recherche_pivot(A, j0):
    n = len(A)
    imax = j0
    for i in range(j0+1, n-1):
        if abs(A[i][j0]) > abs(A[imax][j0]):
            imax = i
    return imax

def permutation(A,i,j):
    t = A[i]
    A[i] = A[j]
    A[j] = t

def transvection(A,i,j,x):
    # si vecteur
    if type(A[0]) != list:
        A[i] = A[i] + x*A[j]
    else:
        n = len(A[0])
        for k in range(n+1):
            A[i][k] = A[i][k] + x*A[j][k]

def resolution_systeme(A,b):
    n = len(A)
    for i in range(n-1):
        i0 = recherche_pivot(A,i)
```

```
        permutation(A,i,i0)
        permutation(b,i,i0)
        for k in range(i+1,n):
            x = A[k][i]/A[i][i]
            transvection(A,k,i,-x)
            transvection(b,k,i,-x)
    return triangle(A,b)

def methode_euler(F,y0,t):
    y = [0]*len(t)
    y[0] = y0
    for i in range(len(t)-1):
        y[i+1] = y[i]+(t[i+1]-t[i])*F(y[i],t[i])
    return y

A0=[[2.,3.,4.],[1.,5.,5.],[3.,2.,4.]]
t=np.linspace(0,10,1000)
b0=[3.,8.,7.]
x0,y0,z0=resolution_systeme(A0,b0)
print x0,y0,z0
def F(x,t):
    return (-y0*x-z0)/x0
plt.figure(1)
x=methode_euler(F,0,t)
plt.plot(t,x)

plt.figure(2)
for i in range(1,20):
    b0=[3.*i,8.*i,7.]
    x0,y0,z0=resolution_systeme(A0,b0)
    print x0,y0,z0
    def F(x,t):
        return (-y0*x-z0)/x0
    x=methode_euler(F,0,t)
    plt.plot(t,x)
```

```
# -*- coding: utf-8 -*-
"""
ÃĽditeur de Spyder

Ceci est un script temporaire.
"""



def ligneRef(A,i) :
    n=len(A)
    maxi = abs(A[i,i])
    jmax = i
    for j in range(i+1,n) :
        if abs(A[j,i]) > maxi :
            maxi = abs(A[j,i])
            jmax = j
    return jmax

import copy
def echangeLignes (A,i,j) :
    A1=copy.deepcopy (A[i])
    A2=copy.deepcopy (A[j] )
    A[j]=A1
    A[i]=A2

def transvection(A,i,j,alpha) :
    A[j]=A[j]-alpha*A[i]


def pivotGauss (A,B) :
    n=len(A)
    for i in range (n) :
        jmax = ligneRef (A,i )
        pivot=A[jmax,i]
        if pivot==0:
            return ( 'erreur' )
# echange l i g n e s i e t jmax dans A e t dans B
        echangeLignes(A,i,jmax)
        echangeLignes(B,i,jmax)
# n o rm a l i s a t i o n de l a l i g n e du p i v o t
        for j in range (n) :
            A[i,j] = A[i,j]/pivot
            B[i] = B[i]/pivot
# on e l im i n e l e s termes au d e s s u s e t en d e s s o u s du p i v o t
        for j in range (n) :
            if j != i :
```

```
                alpha=A[j,i]
                transvection (A,i,j,alpha)
                B[j]=B[j]-alpha*B[i]

                return B

A=np.array([[2.,3.],[1.,4.]])
B=np.array([1.,2.])

print pivotGauss(A,B)
```