TP 3 - éléments de correction

```python
import numpy as np
import math

## Partie 1 :
## Exercice 2 : définir des matrices particulières
M1 = np.arange(25).reshape(5,5)
M2 = np.ones((4,4))
M2[2,3] = 2
M2[3,1] = 6

v_2_6 = np.arange(2,7)
M3_carre = np.diag(v_2_6,-1)
print(M3_carre)
M3 = M3_carre[:,0:5]
print(M3)

## Partie 2 :
## Exercice 3 : résolution avec des fonctions python
A = [[1,2,3],[2,3,1],[3,1,2]]
b = [1,2,3]
x_solve = np.linalg.solve(A,b)
x_inv = np.dot(np.linalg.inv(A),b)
print(x_solve)
print(x_inv)
print(A)

## Exercice 4
print("\nExercice 4\n")
import time

A = np.diag(3 * np.ones(300), 0) + np.diag(np.ones(300-1),-1) + np.diag(np.ones(300-1),1)
b = np.ones(300)
n = 300

tps_debut_solve = time.time()
for m in range(n):
    np.linalg.solve(A,b)
tps_solve = time.time() - tps_debut_solve

tps_debut_inv = time.time()
for m in range(n):
    np.dot(np.linalg.inv(A),b)
tps_inv = time.time() - tps_debut_inv

print(f"Temps de résoudre {n} fois\navec solve : {tps_solve:.3f} s\navec inv : {tps_inv:.3f} s")

## Partie 3 : pivot de Gauss et remontée
def pivotgauss_part1(A,b):
    A_c = np.copy(A)
    b_c = np.copy(b)
    n = np.shape(A_c)[0]
    for i in range(n-1):
        for k in range(i+1,n):
            l_ki = A_c[k,i]/A_c[i,i]
            A_c[k,i] = 0
            for j in range(i+1,n):
                A_c[k,j] = A_c[k,j] - l_ki * A_c[i,j]
            b_c[k] = b_c[k] - l_ki * b_c[i]
    return A_c,b_c

def pivotgauss_part2(A_mod,b_mod):
    n = np.shape(A_mod)[0]
    x = np.zeros(n, dtype=float)
    for i in range(n-1,-1,-1):
        s = 0
        for k in range(i+1,n):
            s = s + A_mod[i,k] * x[k]
        x[i] = 1/A_mod[i,i] * (b_mod[i] - s)
    return x

```

```python
def pivotgauss(A,b):
    A_mod , b_mod = pivotgauss_part1(A,b)
    x = pivotgauss_part2(A_mod,b_mod)
    return x

A = np.array([[1,2,3],[4,5,6],[6,7,9]],dtype=float)
b = np.array([1,2,3],dtype=float)
x = pivotgauss(A,b)
print(f"A={A}")
print(f"b={b}")
print(f"x={x}")
print(f"Vérification : Ax = {A@x}")

## Partie 4 : conditionnement
A = np.array([[10,7,8,7],[7,5,6,5],[8,6,10,9],[7,5,9,10]],dtype = float)
b = np.array([32,23,33,31])
x = np.linalg.solve(A,b)
print(f"x = {x}")

delta_b = np.array([0.1,-0.1,0.1,-0.1])
x_perturbe = np.linalg.solve(A,b+delta_b)
print(f"x_perturbe = {x_perturbe}")

## Partie 5 : LU
from scipy import linalg

## Exercice 6
A = np.array([[1,2,3],[4,5,6],[6,7,9]])
matrice_lu_compacte, v = linalg.lu_factor(A)
print(f"matrice_lu_compacte =\n {matrice_lu_compacte}\n\nvecteur_info_permutation =\n {v}\n")

P, L, U = linalg.lu(A)
print(f"P =\n{P}\nL =\n{L}\nU =\n{U}\n")

P_v = np.array([[0,0,1],[1,0,0],[0,1,0]])
print(f"P_v x A =\n {np.dot(P_v,A)}")
print(f"L x U =\n {np.dot(L,U)}")
print(f"A =\n{A}")
print(f"PxLxU =\n {P@L@U}")

## Exercice 7
A = np.diag([-2,5,-1,4,-2],0) + np.diag([-4,-3,-2,2],-1) + np.diag([1,2,-1,1],1)
b = np.array([1,-2,1,3,6])
info_plu = linalg.lu_factor(A)
x = linalg.lu_solve(info_plu,b)
```