```
# ============================== #
# PSI*                           #
# DS d'informatique du 12/2/2018 #
# Corrigé                        #
# Version du 17/03/2018          #
# ============================== #

## Importations (telles que définies dans le sujet)
import numpy as np
import matplotlib.pyplot as plt
import math

# ------------------------ #
# --> METHODE DE JACOBI <-- #
# ------------------------ #

## Q6.
def nouveau_potentiel(V,rhos,frontiere,i,j):
    if frontiere[i,j]:
        return(V[i,j])
    else:
        return((V[i+1,j]+V[i-1,j]+V[i,j+1]+V[i,j-1]+rhos[i,j])*0.25)

## Q8.
def itere_J(V,rhos,frontiere):
    N = V.shape[0] # RQ : notre N correspond ici à N+1 dans l'énoncé
    new_V = np.zeros((N,N))
    e = 0
    for i in range(N):
        for j in range(N):
            new_V[i,j] = nouveau_potentiel(V,rhos,frontiere,i,j)
            e += (new_V[i,j] - V[i,j])**2
    e = e**0.5/(N-1) # ATTENTION au dénominateur (cf. la remarque RQ ci-
dessus)
    V = new_V.copy()
    return(e)

## Q9.
def poisson(f_iter,V,rhos,frontiere,eps):
    e = eps # ou tout autre valeur supérieure ou égale à eps
    while e >= eps:
        e = f_iter(V,rhos,frontiere)

# ----------------------------- #
# --> METHODE DE GAUSS-SEIDEL <-- #
# ----------------------------- #

## Q11.
def itere_GS(V,rhos,frontiere):
    N = V.shape[0]
    e = 0
    for i in range(N):
        for j in range(N):
            # Le nouveau potentiel est d'abord placé dans la variable newP
            # pour le calcul d'erreur (l'ancien et le nouveau potentiel
étant
            # requis.
            newP = nouveau_potentiel(V,rhos,frontiere,i,j)
            e += (newP - V[i,j])**2
            V[i,j] = newP
```

```
    e = (e**0.5)/(N-1)
    return(e)

# ---------------------------------------- #
# --> METHODE DE GAUSS-SEIDEL ADAPTATIVE <-- #
# ---------------------------------------- #

## Q12.
def nouveau_potentiel_SOR(V,rhos,frontiere,i,j,omega):
    if frontiere[i,j]:
        return(V[i,j])
    else:
        # On utilise la fonction nouveau_potentiel...
        return((1-
omega)*V[i,j]+omega*nouveau_potentiel(V,rhos,frontiere,i,j))

## Q13.
def itere_SOR(V,rhos,frontiere):
    N = V.shape[0]
    # Détermination de la valeur optimale de omega
    omega = 2 / (1 + math.pi / (N-1))
    e = 0
    for i in range(N):
        for j in range(N):
            # Le nouveau potentiel est d'abord placé dans la variable newP
            # pour le calcul d'erreur (l'ancien et le nouveau potentiel
étant
            # requis.
            newP = nouveau_potentiel_SOR(V,rhos,frontiere,i,j,omega)
            e += (newP - V[i,j])**2
            V[i,j] = newP
    e = (e**0.5)/(N-1)
    return(e)
```