Annexe A : Script Python

```python
## Importation des bibliothèques

import matplotlib.pyplot as plt
import numpy as np


## Paramétrage

e_0 = 8.85*10**(-12)
objets = []


## Construction de la distribution de charge et de la zone d'observation

def ajouter_objet(x,y,q):
    objets.append([x,y,q])

# Ajout d'une charge ponctuelle (enlever les commentaires pour ajouter la distribution)
"""
q_charge = - 1.6*10**(-19)
L = 10**(-10)
a = L / 100

ajouter_objet(0, 0, q_charge)
"""
# Ajout d'un dipôle (enlever les commentaires pour ajouter la distribution)
"""
q_dipole = 2.9*10**(-20)
d_dipole = 1.274*10**(-10)
L = 10**(-9)
a = L / 100

ajouter_objet(- d_dipole/2, 0, q_dipole)
ajouter_objet(+ d_dipole/2, 0, - q_dipole)
"""
# Ajout d'un condensateur (enlever les commentaires pour ajouter la distribution)
"""
q_cond = 10 ** (-14)
L_cond = 0.01 # Longueur des plaques du condensateur
e_cond = 0.005 # Distance entre les deux plaques
L = 0.015 # Longueur du domaine
a = L/100 # pas spatial

xi = np.linspace(0, L_cond, 100)
for i in xi:
    ajouter_objet(-L_cond/2 + i, e_cond/2, q_cond)

for i in xi:
    ajouter_objet(- L_cond/2 + i, - e_cond/2, - q_cond)
"""
```



```python
## Calcul du potentiel

def calculV(xM,yM,q,xO,yO):
    """
    calcul du potentiel en xM, yM créé par une charge ponctuelle q placée en xO,yO
    """

    # problème pour la position de coordonnées xO,yO
    if xM == xO and yM == yO :
        return 0

    else :
        return q/(4*np.pi*e_0)*1/((xM-xO)**2+(yM-yO)**2)**(1/2)

x = np.linspace(- L/2, L/2, int(L/a))
y = np.linspace(- L/2, L/2, int(L/a))
X, Y = np.meshgrid(x,y)

nM = len(X)
no = len(objets)
V = np.zeros((nM,nM))

for i in range(0,nM) :
    for j in range (0,nM):
        for k in range(0,no):
            V[i,j] = V[i,j] + calculV(x[i], y[j], objets[k][2], objets[k][0], objets[k][1])


## Calcul du champ électrique

Ex = np.zeros((nM - 1, nM - 1))
Ey = np.zeros((nM - 1, nM - 1))

for i in range (0, nM-2) :
    for j in range (0, nM-2) :
        Ex[i,j] = - (V[i,j+1] - V[i,j]) / a
        Ey[i,j] = - (V[i+1,j] - V[i,j]) / a


## Ajustage des matrices V, X et Y

Vajust = np.zeros((nM - 1, nM - 1))

for i in range (0, nM - 1) :
    for j in range (0, nM - 1) :
        Vajust[i,j] = V[i,j]

Xajust = np.zeros((nM - 1, nM - 1))
for i in range (0, nM - 1):
    for j in range (0, nM - 1):
        Xajust[i,j] = X[i,j]

Yajust = np.zeros((nM - 1, nM - 1))
for i in range (0, nM - 1):
    for j in range (0, nM - 1):
        Yajust[i,j] = Y[i,j]


## Affichage des courbes

schema = plt.contour(Xajust, Yajust, Vajust, 40)

cbar = plt.colorbar()
cbar.set_label("Potentiel en Volt")
plt.axis("equal")
plt.title("Equipotentielles et lignes de champ")

schema = plt.streamplot(Xajust, Yajust, Ex, Ey, linewidth=1, density=1)

plt.show()
```