

Semaine 1 : Les sources lumineuses

TP1

Maxime CATTEAU
Léane TEXIER

1ère partie

Note : La première partie reposant sur l'introduction à l'outil de développement, nous avons été informé qu'il ne fallait pas que les questions de cette partie apparaissent dans ce rapport, et ce même si tous les exercices ont bien été réalisés.

2ème partie

Question n°1 :

Exécuter le code ci-dessous dans Spyder et commentez le résultat obtenu

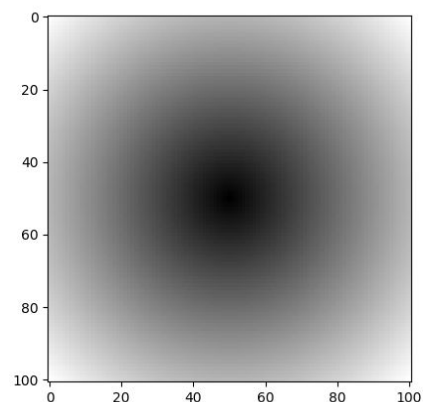
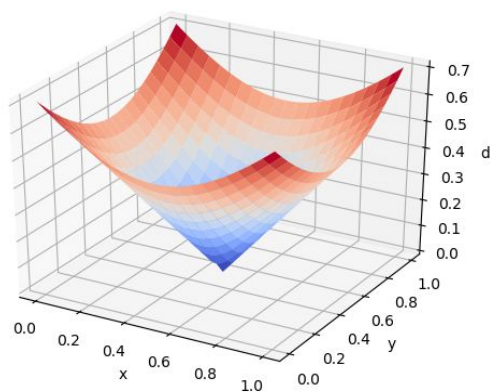


Figure 1 : La figure représente les distances entre la source lumineuse et chaque point. On remarque alors que plus on est proche de la source lumineuse, plus la distance est faible, donc plus la couleur tends vers le bleu. Au contraire, plus on est loin de la source lumineuse, plus la distance est élevée et plus la couleur tends vers le rouge.

Figure 2 : La figure représente également les distances entre la source lumineuse et chaque point. On remarque ainsi que plus un point est proche du centre, plus son niveau de gris est élevé (proche du noir), cela est dû au fait que la distance entre la source lumineuse et ce point est faible. Au contraire, plus on est éloigné, moins son niveau de gris est élevé (proche du blanc, distance élevée avec la source lumineuse).

Question n°2 :

*Expliquer la signification des lignes de ce code correspondant aux initialisations et affectations de **axe**, **x**, **y** et **d**. Pour bien comprendre les opérateurs et types utilisés, vous pouvez exécuter chaque instruction individuellement dans la console IPython et vous aider de l'explorateur de variables.*

La variable **axe** permet de définir les valeurs qui seront présentes sur chaque axe.

La variable **x** correspond aux valeurs des abscisses du graphique de tous les points. Il s'agit d'un tableau de 101 lignes identiques.

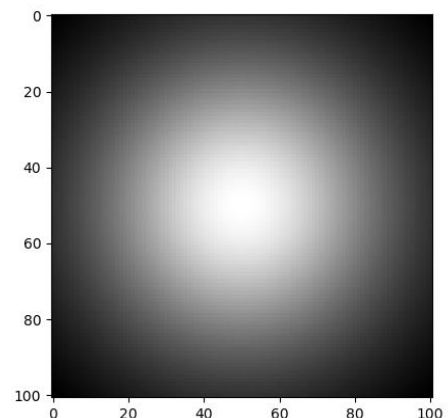
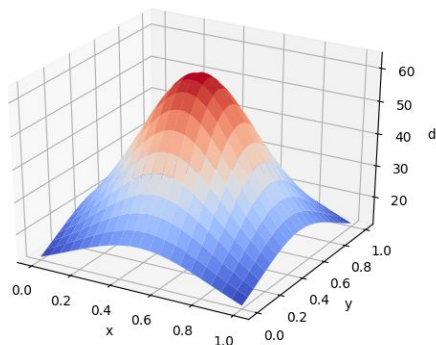
La variable **y** correspond aux valeurs des ordonnées du graphique. Il s'agit d'un tableau de 101 colonnes identiques.

=> Un point de la surface est défini par un couple (**x,y**).

La variable **d** correspond à la distance entre chaque point de la surface et la source lumineuse.

Question n°3 :

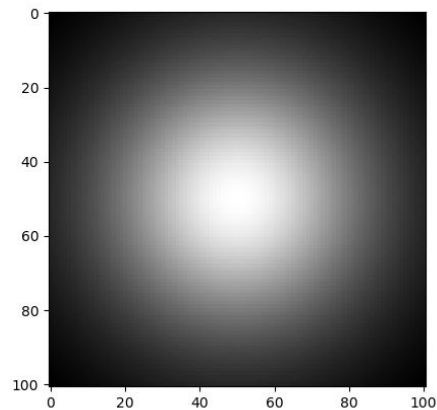
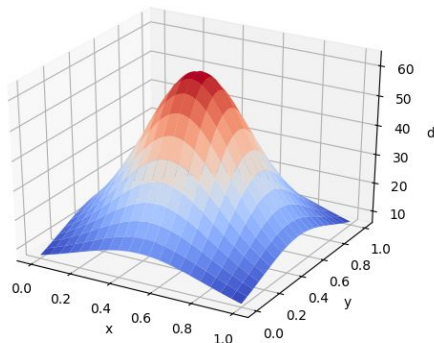
Modifier ce code afin de calculer, puis représenter sous forme d'image et de fonction 3D, les valeurs d'éclairement reçues par les éléments de la surface plane éclairée par la source ponctuelle isotrope.



3ème partie

Question n°1 :

Modifier le code de la fonction précédente afin de calculer, puis représenter sous forme d'image et de fonction 3D, les valeurs d'éclairement reçues par les éléments de la surface plane éclairée par une source ponctuelle **lambertienne**. Comme dans le TD, on supposera que l'intensité de la source selon l'axe vertical (maximum) est identique à celui de la source ponctuelle.



Question n°2 :

Comparer les valeurs d'éclairement obtenues dans les deux conditions, en calculant dans les deux cas (isotrope et lambertien) la **variation relative maximale** obtenue sur la surface = $(\max - \min) / \max$. Cette variation sera exprimée en pourcentage.

Cas Isotrope :

Dans le cas isotrope, la variation relative maximale est d'environ 80.75%. Cela a été calculé grâce à la formule donnée dans l'énoncé avec comme valeur maximale, la valeur d'éclairement en 50,50 qui correspond au point d'éclairement maximale et comme valeur minimale, la valeur d'éclairement en 0,0 qui est un des points les plus loin de la source lumineuse donc qui a la valeur d'éclairement minimale.

$$E[50][50] = E.\max() = 63.661977236758133$$

$$E[0][0] = E.\min() = 12.251753231595377$$

$$(E[50][50] - E[0][0]) / E[50][50] * 100 = 80.75499102701248$$

Cas Lambertien :

Dans le cas lambertien, la variation relative maximale est d'environ 88.89%. Cela a été calculé grâce à la formule donnée dans l'énoncé avec comme valeur maximale, la valeur d'éclairement en 50,50 qui correspond au point d'éclairement maximale et comme valeur minimale, la valeur d'éclairement en 0,0 qui est un des points les plus loin de la source lumineuse donc qui a la valeur d'éclairement minimale.

$$E[50][50] = E.max() = 63.661977236758133$$

$$E[0][0] = E.min() = 7.0735530263064561$$

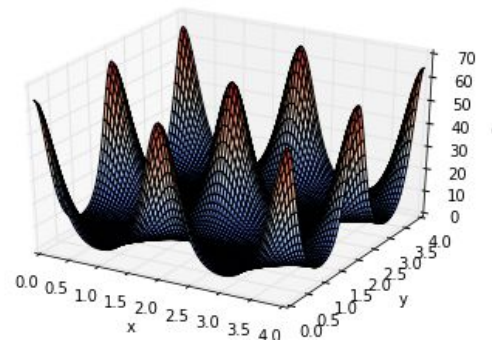
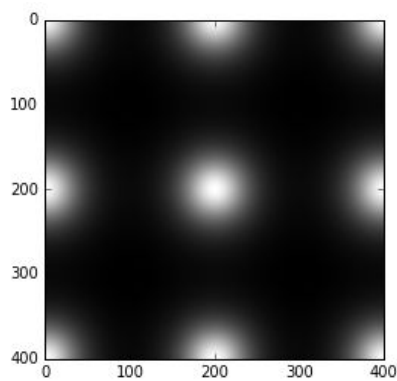
$$(E[50][50]-E[0][0])/E[50][50]*100 = 88.8888888888889$$

4ème partie

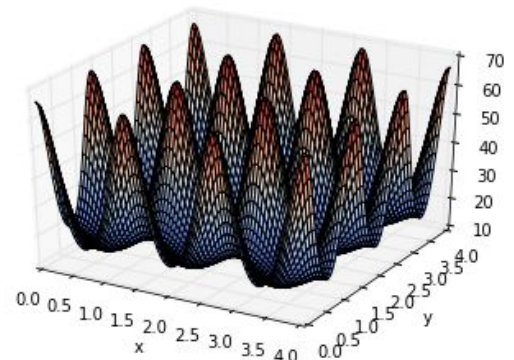
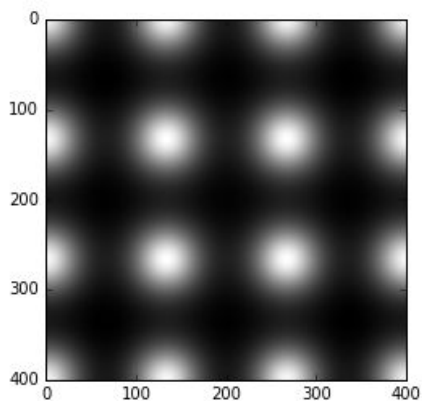
Question n°1 :

Compléter votre code afin de calculer l'éclairement reçu par la surface carrée **en fonction de N**. On rappelle que l'éclairement total reçu par un élément de surface est simplement la somme des éclairements reçus de chaque source.

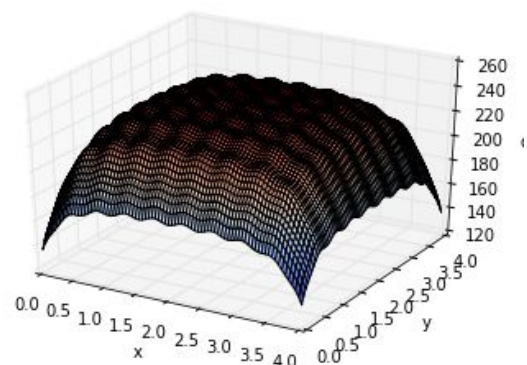
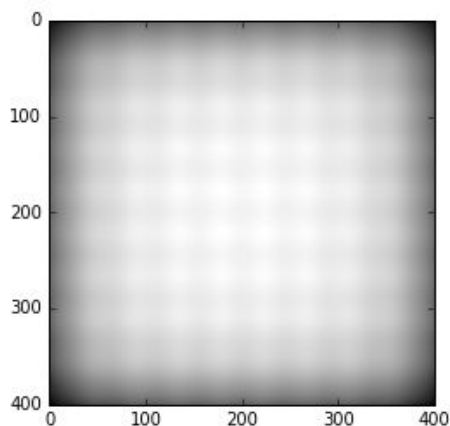
n = 3 :



n = 4 :



n = 10 :



Question n°2 :

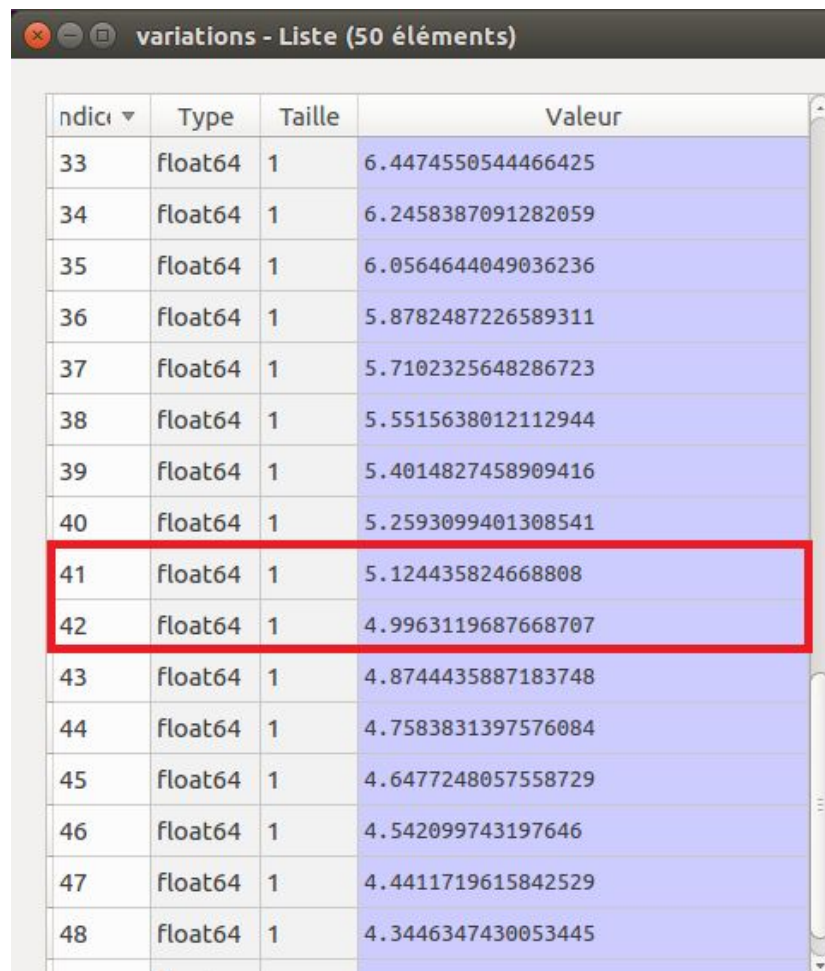
Pour des valeurs croissantes de N, calculer la variation relative de l'éclairement reçu par la surface. Combien de sources faut-il utiliser pour avoir une variation maximale sur la surface inférieure à 5% ?

Nous avons recherché la formule de la variation relative :

$$Vr = (V1 - V0) / V0$$

Pour en obtenir le pourcentage, il suffit de multiplier par 100.

Nous avons donc construit un tableau contenant les variations à chaque incrémentation de n. Ce tableau de variations est relié avec le tableau des moyennes, à chaque itération de l'éclairement. Ainsi, grâce à cela, nous observons par l'intermédiaire de l'explorateur de variables le résultat suivant :



indice ▾	Type	Taille	Valeur
33	float64	1	6.4474550544466425
34	float64	1	6.2458387091282059
35	float64	1	6.0564644049036236
36	float64	1	5.8782487226589311
37	float64	1	5.7102325648286723
38	float64	1	5.5515638012112944
39	float64	1	5.4014827458909416
40	float64	1	5.2593099401308541
41	float64	1	5.124435824668808
42	float64	1	4.9963119687668707
43	float64	1	4.8744435887183748
44	float64	1	4.7583831397576084
45	float64	1	4.6477248057558729
46	float64	1	4.542099743197646
47	float64	1	4.4411719615842529
48	float64	1	4.3446347430053445

On constate que lorsque n vaut 42, nous passons sous la barre des 5%. Cela signifie qu'il faut 42 sources pour avoir une variation maximale sur la surface inférieure à 5%.

Annexe

2ème partie

Question n°3 :

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

plt.close('all')
axe = np.linspace(0, 1, 101)

# Définition des éléments de surface
x = np.ones((101,1)).dot(axe.reshape((1,101)))
y = axe.reshape((101,1)).dot(np.ones((1,101)))

# Position de la source
xs = 0.5
ys = 0.5

# Calcul de la distance
d = np.sqrt((x - xs) * (x - xs) + (y - ys) * (y - ys))

# Hauteur
h = 0.5

# Intensité
I = 50/np.pi

# Calcul des valeurs d'éclairement reçus
E = (I * h) / (d ** 2 + h ** 2)** (3./2.)

# Création d'une figure matplotlib
fig1 = plt.figure()

# Création des axes sur la figure fig1
axes = fig1.gca(projection=Axes3D.name)
axes.set_xlabel('x')
axes.set_ylabel('y')
axes.set_zlabel('d')

# Affichage de la fonction distance sur fig1
axes.plot_surface(x, y, E, cmap='coolwarm', rstride=5, cstride=5, antialiased=True)

# Visualisation de la fonction distance sous forme d'image en niveaux de gris
fig2 = plt.figure()
plt.imshow(E, cmap='gray')

# Affichage des figures matplotlib à l'écran
plt.show()
```

3ème partie

Question n°1 :

Modification du calcul de E par la formule ci-dessous (le reste du code reste inchangé par rapport au code précédent) :

$$E = (I * h^{**2}) / (d^{** 2} + h^{** 2})^{** 2}$$

4ème partie

Question n°1 :

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

plt.close('all')

# Nombre de LEDs par ligne
n = 10

# Definition des echantillons sur un axe
axe = np.linspace(0, 4, 401)

# Definition de la grille de points de 4m * 4m
grille = np.linspace(0,4,101)

# Definition de l'emplacement des spots en fonction de n
grille_init = np.linspace(0,4, n)

# Definition des elements de surface
x = np.ones((401,1)).dot(axe.reshape((1,401)))
y = axe.reshape((401,1)).dot(np.ones((1,401)))

# Position de la source
xs = 0.5
ys = 0.5

# Calcul de la distance
# d = np.sqrt((x - xs) * (x - xs) + (y - ys) * (y - ys))

# Hauteur
h = 0.5

# Intensite
I0 = 50/np.pi
```

```

# Calcul des valeurs d'eclairement recus
#E = (I0 * h**2) / (d ** 2 + h ** 2)** 2
E = np.zeros((401,401))
E = np.array(E)
# Parcours des points de sources lumineuses
for i in grille_init:
    for j in grille_init:
        # 1 : Calcul de la distance a chaque fois
        d = np.sqrt((x - j) * (x - j) + (y - i) * (y - i))
        # 2 : construire le tableau des valeurs d'eclairement pour i
        tmp = (I0 * h**2) / (d ** 2 + h ** 2)** 2
        tmp = np.array(tmp)
        # 3 : additionner au tableau de E
        E = E+tmp
# Creation d'une figure matplotlib
fig1 = plt.figure()

# Creation des axes sur la figure fig1
axes = fig1.gca(projection=Axes3D.name)
axes.set_xlabel('x')
axes.set_ylabel('y')
axes.set_zlabel('d')

# Affichage de la fonction eclairement sur fig1
axes.plot_surface(x, y, E, cmap='coolwarm', rstride=5, cstride=5, antialiased=True)

# Visualisation de la fonction eclairement sous forme d'image en niveaux de gris
fig2 = plt.figure()
plt.imshow(E, cmap='gray')

# Affichage des figures matplotlib a l'ecran
plt.show()

```