Université Lille 1

MASTER 1 INFORMATIQUE

Deuxième semestre

Traitement d'image

TP n°01

BARCHID Sami

CARTON Floriane

Année académique 2018-2019

Introduction

La matière vue dans ce TP traite de la relation entre l'image et la lumière, de la source lumineuse jusqu"à l'observateur en passant par les objets qui réfléchissent cette lumière.

Ce rapport de TP est divisé en trois parties :

- **Éclairement d'une source ponctuelle isotrope :** réponses aux questions posées dans la partie du TP concernant le calcul de l'éclairement d'une surface par une source ponctuelle isotrope.

- **Éclairement d'une source ponctuelle lambertienne** : réponses aux questions posées dans la partie du TP concernant le calcul de l'éclairement d'une surface par une source lambertienne.

- **Éclairement d'une grille de sources ponctuelles** : réponses aux questions posées dans la partie du TP concernant le calcul de l'éclairement d'une surface par plusieurs sources (quasi) ponctuelles lambertiennes.

Éclairement d’une source ponctuelle isotrope

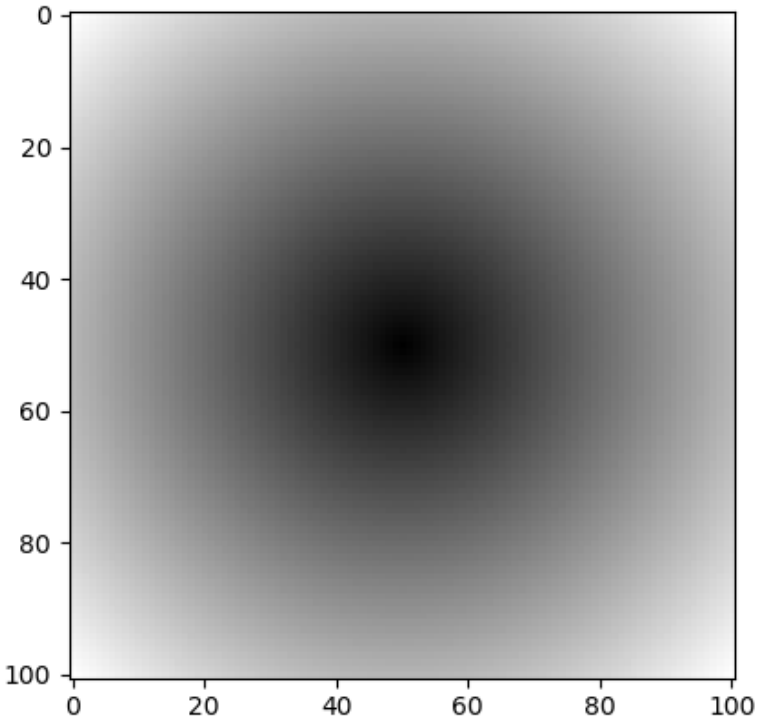
Commenter le résultat obtenu par le code ci-dessous

```
# Importer le module numpy pour manipuler des vecteurs / matrices
import numpy as np

# Importer le module matplotlib.pyplot pour l'affichage de figures
import matplotlib.pyplot as plt

# Permettre l'affichage de surfaces 3D
from mpl_toolkits.mplot3d import Axes3D

# Fermeture des figures précédemment ouvertes
plt.close('all')

# Définition des échantillons sur un axe
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))

# 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, d, 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(d, cmap='gray')

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

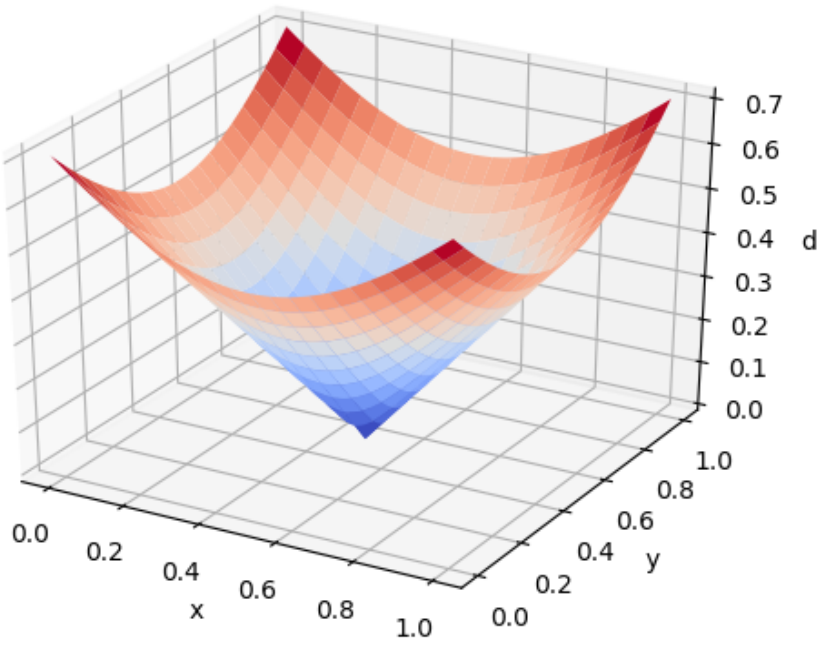
Ce code calcule la distance par rapport à la source lumineuse ponctuelle isotrope de chaque centimètre carré de la surface. Pour ce faire, il calcule x et y, des matrices 101x101 qui représentent les positions en x et en y de chaque centimètre carré. Ensuite, il définit xs et ys, la position de la source lumineuse. Puis il calcule d, la matrice 101x101 qui représente la distance de chaque point par rapport à la source. Le calcul de d est une application directe du théorème de Pythagore (comme vue en TD) où la distance d est l'hypoténuse.

Le code produit deux figures :

1. Une image en niveaux de gris qui représente la matrice d et donc la valeur de la distance par rapport à la source pour chaque centimètre carré de la surface. On distingue que plus on se rapproche du centre de l'image (et donc de la source), plus on vire vers le noir (et donc plus la valeur de « d » se rapproche de 0).



2. Un graphique en trois dimensions qui représente aussi la distance par rapport à la source lumineuse de chaque centimètre carré de la surface. On remarque également ici que plus on se rapproche de la source, plus la valeur tend vers 0 (puisque la distance par rapport à la source se réduit).



Expliquer les lignes de code

```
# Définition des échantillons sur un axe
axe = np.linspace(0, 1, 101)
```

La fonction « **numpy.linspace** » prend trois paramètres (que l'on va nommer a, b et c) et renvoie un tableau d'une dimension de c éléments

allant d'une valeur de départ a jusqu'à une valeur d'arrivée b de manière croissante. Ici, « axe » vaut donc un tableau de 101 éléments répartis de 0 à 1. Les premiers éléments valent, par exemple, « 0, 0.01, 0.02, ... ».

```
x = np.ones((101,1)).dot(axe.reshape((1,101)))
```

- « **numpy.ones((101,1))** » : crée une matrice 101x1 remplie de 1
- « **matrice1.dot(matrice2)** » : fait le produit matriciel de matrice1 et matrice 2
- « **matrice.reshape((1,101))** » : redimensionne la matrice en une matrice 1x101. Il faut veiller à ce que le nombre d'éléments soit le même dans la dimension que l'on a entrée en paramètre. Si « matrice » ne contenait pas 101 éléments, on aurait eu une erreur.

Ainsi, dans la ligne de code :

- on redimensionne « axe » (qui est une matrice 101x1) en une matrice 1x101
- On crée une matrice 101x1 remplie de « 1 »
- On fait le produit matriciel de la matrice remplie de 1 avec la matrice axe redimensionnée. On obtient donc une matrice 101x101 où chaque ligne est composée de 101 éléments de 0 à 1 (chaque ligne est identique). Cela représente les coordonnées x de chaque centimètre carré de la surface.

```
y = axe.reshape((101,1)).dot(np.ones((1,101)))
```

On calcule, avec les mêmes fonctions qu'au-dessus, une matrice 101x101 où chaque colonne est composée de 101 éléments de 0 à 1. Chaque colonne est identique. La matrice représente les coordonnées en y de chaque centimètre carré de la surface.

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

On calcule, pour chaque centimètre carré de la surface, sa distance par rapport à la source ponctuelle. Pour ce faire, nous appliquons le théorème de Pythagore puisque la distance en x et en y d'un point avec la source forment un triangle rectangle dont l'hypoténuse est la distance que l'on cherche.

Il est à noter que Numpy gère directement les calculs avec une constante. Ainsi, lorsque je calcule « xs - x » (où xs est un nombre et x est une matrice), je calcule en réalité pour chaque élément de x automatiquement. On évite de se fatiguer à faire des boucles imbriquées, ce qui donne un code plus lisible.

Modifier le code pour obtenir les valeurs d'éclairement

Il faut d'abord calculer l'intensité énergétique qu'émet la source lumineuse. Comme celle-ci est isotrope, l'intensité est pareille dans toutes les directions.

Cette intensité énergétique est donnée par :

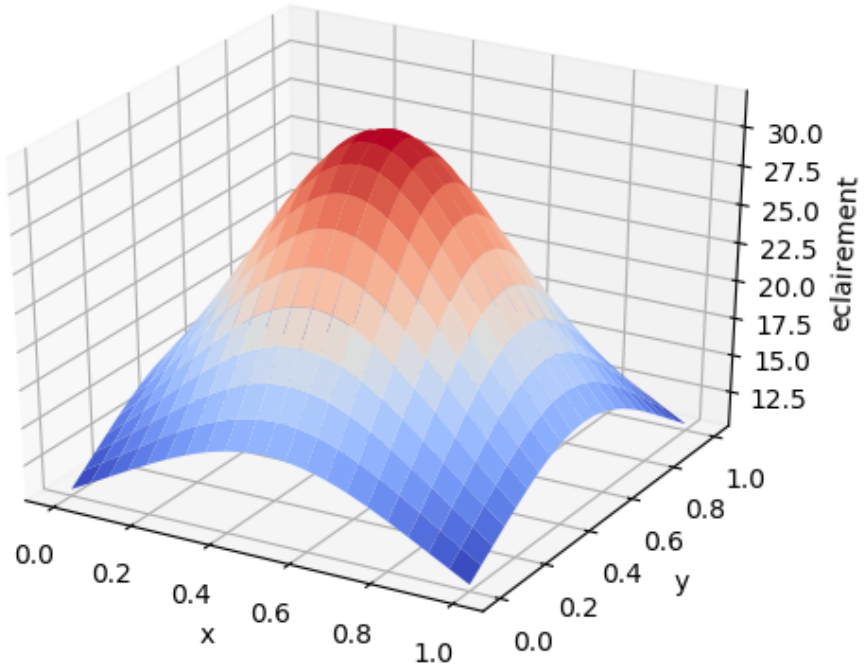
```
# calcul de l'intensité
intensite = 100/(2*np.pi)
```

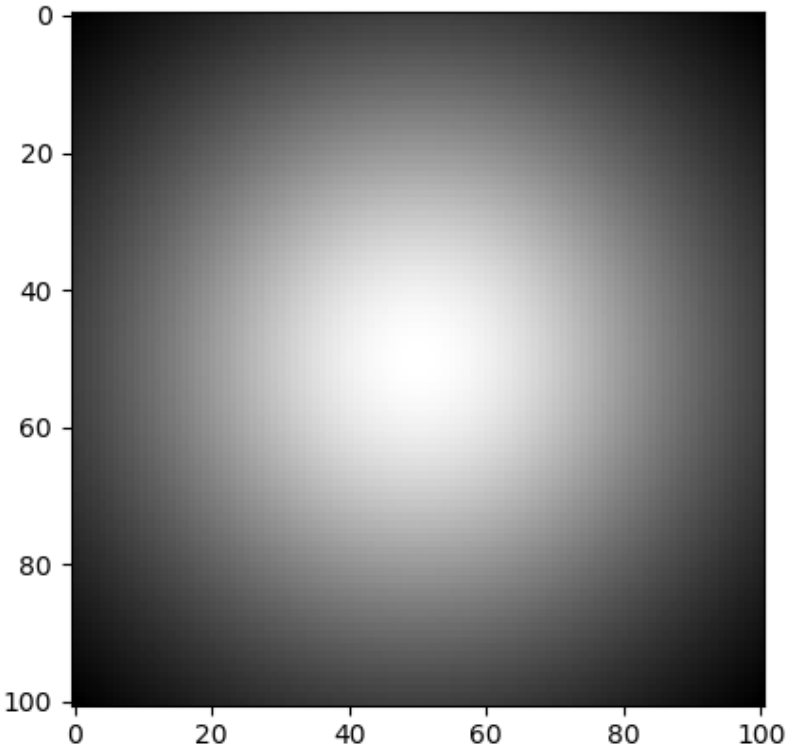
- 100 = les 100 Watts de les
- 2*PI car la source isotrope éclaire une demi-sphère et 2PI stéradian est l'angle solide pour une demi-sphère.

Ensuite, nous calculons l'éclairement en ayant modifié la formule de base énoncée en cours pour ne plus devoir prendre en compte l'angle. Cette transformation de la formule est atteinte grâce à la formule du cosinus d'un angle dans un triangle rectangle (https://fr.wikiversity.org/wiki/Trigonométrie/Cosinus_dans_un_triangle_rectangle).

```
# calcul de l'éclairement de chaque point
eclairement = (intensite * 0.5) / ((d**2 + 0.5**2)**(3/2))
```

Nous obtenons donc les figures suivantes :


30.0
27.5
25.0
22.5
20.0
17.5
15.0
12.5
eclairement
0.0
0.2
0.4
0.6
0.8
1.0
x
0.0
0.2
0.4
0.6
0.8
1.0
y


0
20
40
60
80
100
0
20
40
60
80
100

Nous distinguons ici que l’éclairement est plus grand lorsque nous nous rapprochons de la source lumineuse. L’éclairement étant la « quantité de lumière reçue par unité de surface », il est logique que nous recevons plus de lumière lorsque nous sommes plus proche de la source lumineuse.

Éclairement d’une source ponctuelle lambertienne

Modifier le code de la fonction précédente

Comme la source est lambertienne, nous devons calculer la luminance énergétique. On supposera que l'intensité de la source selon l'axe vertical (maximum) est identique à celui de la source ponctuelle.

Ce qui change par rapport au code précédent est le calcul de l’éclairement :

```
# calcul de l'éclairement de chaque point
eclairement = (intensite * (0.5**2)) / ((d**2 + 0.5**2)**(2))
```

Que l’on a trouvé grâce au calcul suivant :

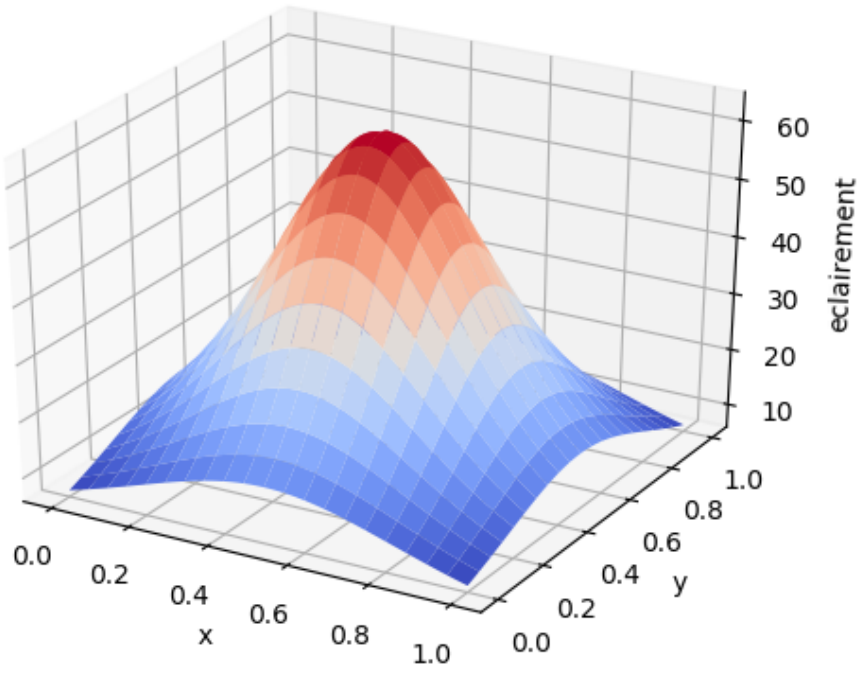
$E(P)=\frac{I_0.\cos^2(\alpha)}{d^2+h^2}=\frac{I_0.h^2}{(d^2+h^2)^2}$ où :

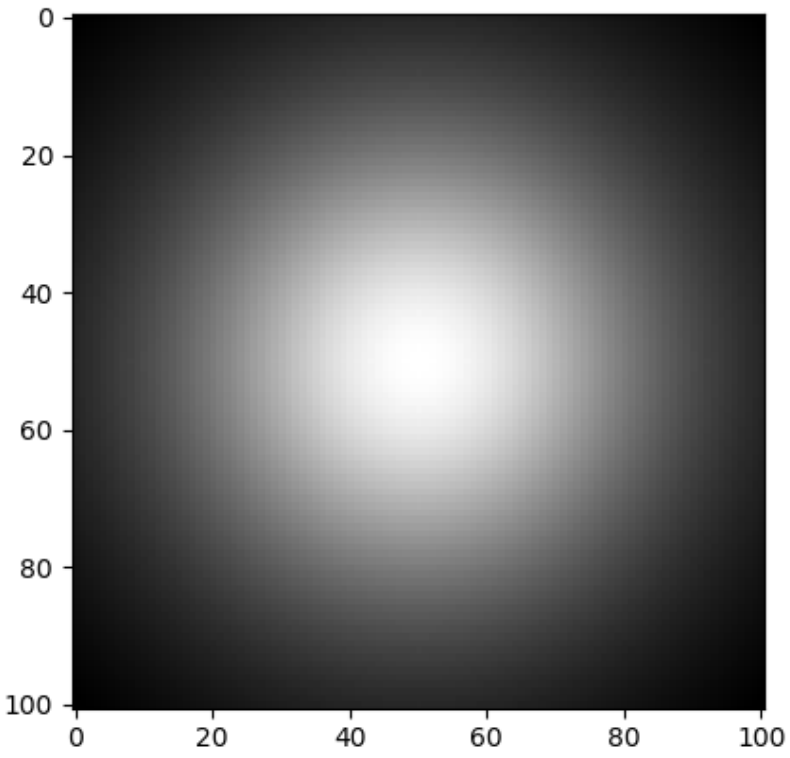
- E(P) est l’éclairement pour le point P
- I_0 est la luminance énergétique maximale (intensité énergétique dans une direction orthogonale à la surface)
- d est la distance à la source lumineuse (calculée avant)
- h est la hauteur de la source lumineuse par rapport à la surface éclairée (dans l’énoncé, elle vaut 0.5m)

- α est l'angle d'incidence du centimètre carré de la surface par rapport à la source lumineuse.

Le calcul varie par rapport à une source isotrope car nous prenons en compte que la luminance énergétique varie selon l'angle d'incidence du point mesuré.

Les résultats obtenus sont :





Nous remarquons ici que, comme pour la source isotrope, l'éclairement diminue lorsque nous nous éloignons de la position de la source lumineuse. Cependant, on distingue aussi que l'éclairement décroît plus fort dans le cas de la source lumineuse lambertienne.

Calculer les variations relatives maximales

- Source isotrope : 66.66666666666667 %
- Source lambertienne : 88.8888888888889 %

On observe que la variation relative maximale pour la source lambertienne est plus élevée que celle de la source isotrope. Ceci se traduit par le fait que, comme observé plus haut, l'éclairement décroît plus rapidement quand nous nous éloignons de la source lumineuse dans le cas de la source lumineuse lambertienne.

Noter que, dans le cas théorique où toute la surface du carré serait éclairée de façon maximale, la variation relatixe maximale serait égale à 0, puisque l’éclairement maximum et minimum seraient égaux.

Éclairement d’une grille de sources ponctuelles

Compléter le code

```
# Définition du nombre de LEDs de la grille NxN
N = 9

# Définition des coordonnées de la grille NxN de leds
grille = np.linspace(-1.5, 2.5, N);

# calcul de l'intensité des LEDs
intensite = 100/(2*np.pi)

eclairementTotal = np.zeros((101,101))

for i in np.nditer(grille):
    for j in np.nditer(grille):
        xs,ys = i,j
        # Calcul de la distance
        d = np.sqrt((x - xs) * (x - xs) + (y - ys) * (y - ys)) # matrice 101x1

        # calcul de l'éclairement de chaque point
        eclairement = (intensite * (0.5)**2) / ((d**2 + 0.5**2)**(2))
        eclairementTotal += eclairement
```

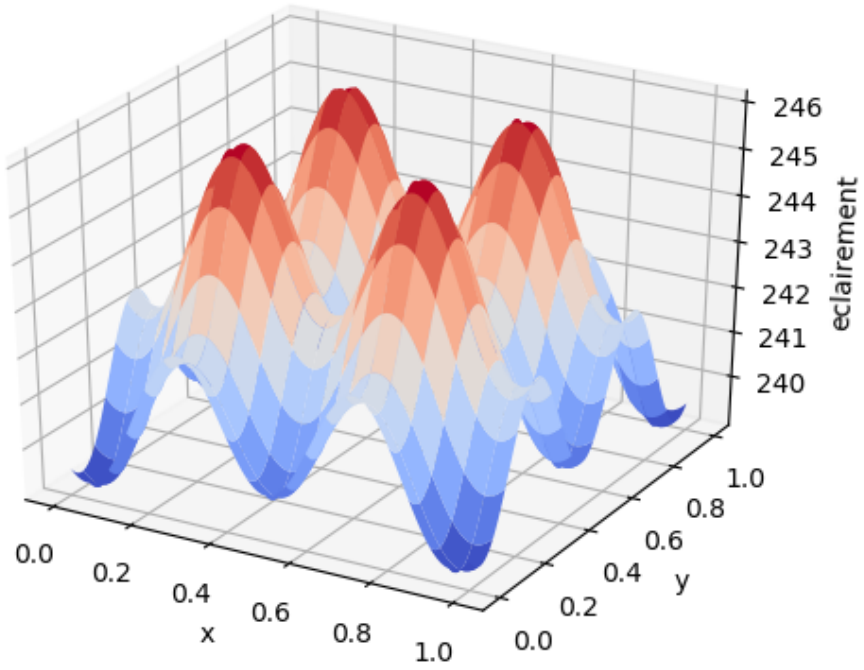
Nous définissons la grille NxN de 4 mètres de côté. La grille commence en -1.5 et termine en 2.5 (en abscisse comme en ordonnée) de sorte que la surface éclairée soit au centre de la grille de LEDs. Nous faisons ensuite la somme des éclairements de chaque LED.

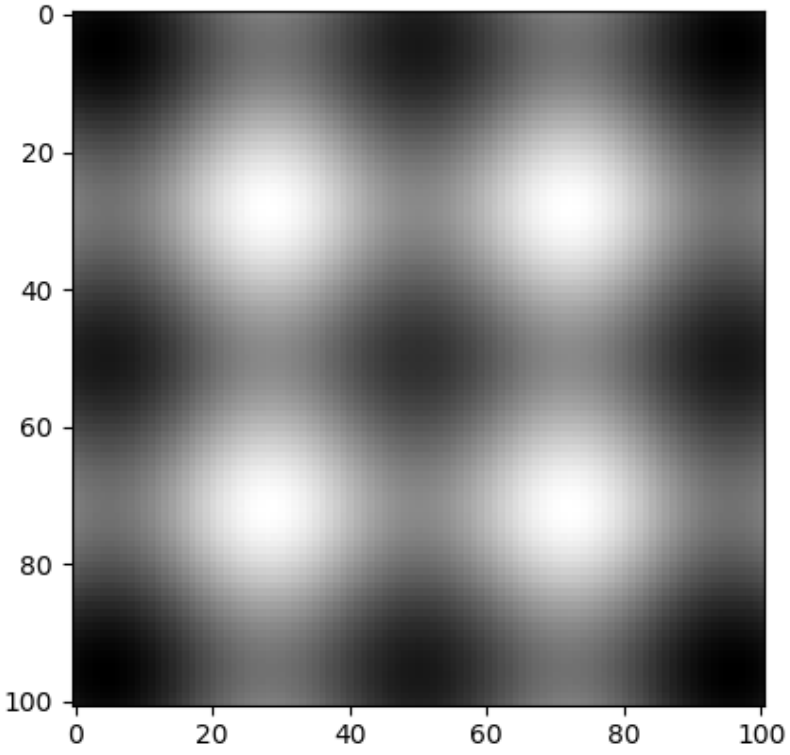
Combien de sources pour une variation maximale relative inférieure à 5 ?

Avec N = 9, la variation maximale relative atteint 5.171635983041575 %, donc légèrement plus que la limite demandée.

Avec N = 10, la variation maximale relative atteint 2.8660662671730934 %.

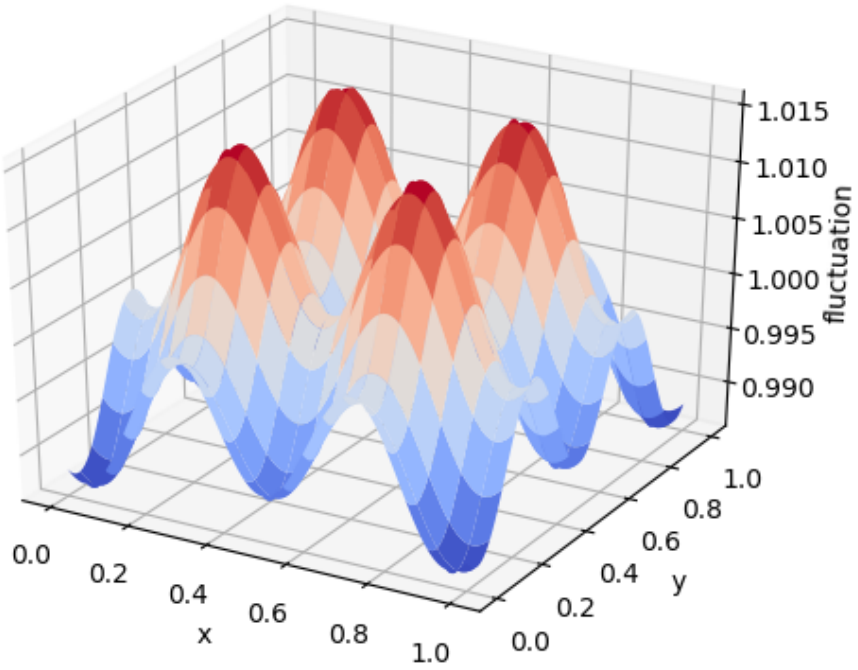
Nous obtenons, pour N = 10, les figures suivantes :





Représenter la fluctuation de l'éclairement de la surface par rapport à sa valeur moyenne

Le calcul de la fluctuation donne la figure suivante :



On remarque que l'écart entre les valeurs basses et haute de la fluctiation n'est pas élevé (ce qui est logique puisque la variation maximale relative n'est que de 2%). On remarque que l'éclairement est plus grand que la moyenne lorsque nous nous rapprochons de points situés juste en-dessous d'une source lumineuse.