Outils Numériques pour Ingénieur·es en Physique

Bloc 2

Ce second bloc du module ONIP se déroule sur 4 séances de 2h, la 4ème séance étant dédiée à l'évaluation.

Objectifs et méthodes

Le principal objectif de ce bloc est de mettre en œuvre une méthode numérique appelée méthode des moindres carrées afin de déterminer la qualité spatiale d'un faisceau laser à partir d'images expérimentales.

Il n'est pas question ici de faire un cours sur les faisceaux laser mais d'introduire quelques notions sur les faisceaux gaussiens nécessaires pour aborder le projet numérique proposé. Ces notions seront approfondies lors du cours sur les lasers au S6.

Une problématique en physique des lasers est de pouvoir mesurer l'écart d'un faisceau réel à un faisceau « modèle » parfaitement gaussien, que l'on dit aussi « limité par la diffraction ». Une méthode expérimentale consiste à prendre sur une caméra des images en coupe transverse selon sa direction de propagation, notée (0z) par la suite.

Le schéma de l'expérience ainsi qu'une image de caméra sont représentés ci-dessous :

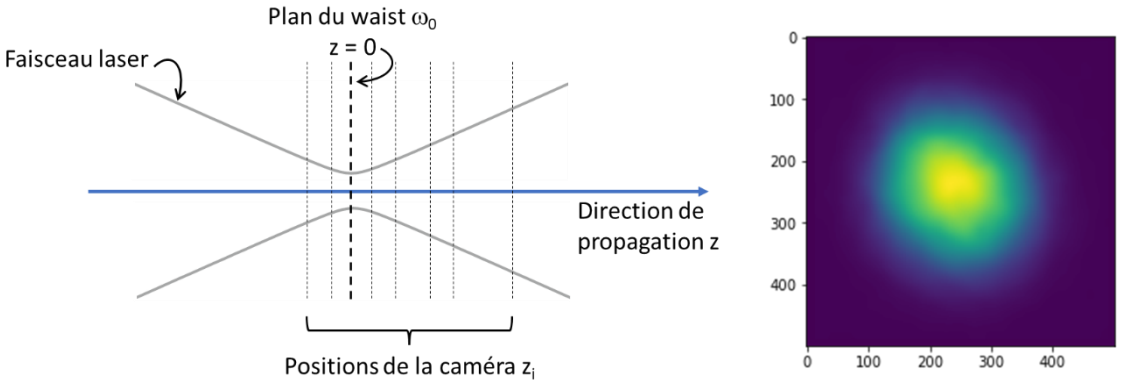

Figure 1. À gauche : schéma de principe de l'expérience. À droite : profil transverse du faisceau à une abscisse z.

Pour chaque image, située dans un plan perpendiculaire à (0z), on repère les coordonnées du barycentre du faisceau (x_{bary} et y_{bary}) puis on effectue deux coupes transverses dans l'image, l'une dans la direction $x = x_{bary}$ et la seconde dans la direction $y = y_{bary}$. Un exemple de coupes transverses dans le plan de l'image est donné ci-après :

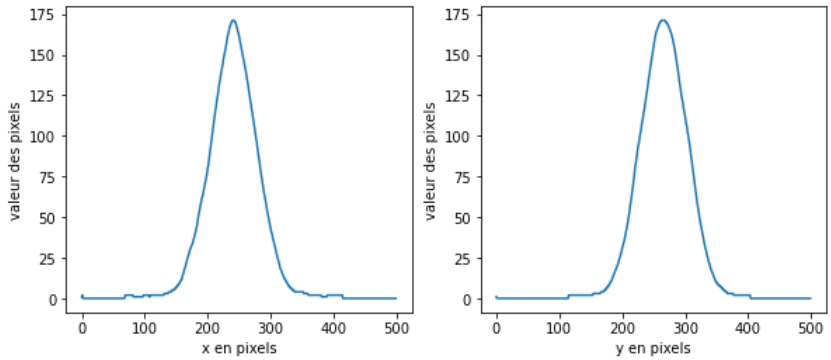

Figure 2. Coupe transverse dans les directions x et y.

On effectue ensuite un ajustement de ces deux coupes avec une fonction gaussienne à partir de la méthode des moindres carrés, présentée en annexe. Cet ajustement permet de déduire les rayons du faisceau selon x et y, que l'on notera respectivement ω_x et ω_y. Ces rayons peuvent en effet être différents si le faisceau est elliptique. En traçant les valeurs de ω_x et ω_y obtenues à chaque image en fonction de la position z de la caméra, on reconstruit l'évolution longitudinale expérimentale du faisceau. L'évolution longitudinale théorique $\omega(z)$ est connue analytiquement pour un faisceau parfaitement gaussien. La formule est donnée en annexe.

En réalisant un nouvel ajustement des valeurs expérimentales avec $\omega(z)$ par la méthode des moindres carrés, on peut déterminer un paramètre appelé le facteur M^2 du faisceau, qui caractérise l'écart du faisceau réel à un faisceau parfaitement gaussien limité par la diffraction dont le M^2 vaut 1.

Le but de ce projet est de déterminer le M^2 d'un faisceau laser à partir d'images expérimentales.

Les images fournies sont des images carrées de 500 pixels de large, codées en niveau de gris sur 8 bits. Chaque pixel possède donc une valeur comprise entre 0 et 255. Le nom des images est mis sous la forme **ProfilXX.tif**, où XX est le numéro de l'image, allant de 1 à 15.

Un fichier **data.csv** est également fourni. Ce fichier contient sur la première colonne la position z de l'image (donnée en mm) par rapport au plan du waist. La seconde colonne contient le nom de l'image correspondant. Ainsi, l'image **Profil1.tif** correspond à une distance z = -51 mm, etc …

Pour traiter ce problème, nous allons tout d'abord nous intéresser à une image seule, Profil1.tif, avant de passer au traitement des 15 images.

Travail à réaliser pour une image seule : Profil1.tif.

Voici les étapes à réaliser, dans l'ordre. Chaque étape doit obligatoirement être validée avant de passer à la suivante !

1- On vous fournit les lignes de code ci-dessous :

```
z = []
nom = []

with open('data.csv', 'r') as f:
    reader = csv.reader(f , delimiter=',')
    for row in reader:
        z = np.append(z,float(row[0]))
```

```
nom = np.append(nom,row[1])
```

Écrire ces lignes dans votre programme. Que permettent-elles de faire ?

2- Lire l'image **Profil1.tif** et à l'afficher à l'écran.
3- Calculer les coordonnées du barycentre de l'image (en pixels) x_{bary} et y_{bary}.
4- Que vaut l'intensité maximale de l'image ? L'intensité minimale ?
5- Tracer sur l'image la droite d'équation $x = x_{bary}$ et la droite d'équation $y = y_{bary}$. Vérifier que ces deux droites se coupent bien au barycentre du faisceau.
6- Tracer sur deux graphiques différents les deux profils transverses du faisceau selon ces deux droites.
7- Écrire une fonction **`gaussienne`** définie par la formule suivante :

$$A + Bexp\left(-2\frac{(x-x_0)^2}{\omega^2}\right)$$

Pour créer une fonction, on utilisera systématiquement la déclaration `def` vue en cours.
Cette fonction **`gaussienne`** possède 4 paramètres ajustables A, B, x_0 et ω et une variable de position x qui seront tous les 5 passés en arguments. Que représentent les 4 paramètres ? Tester cette fonction en la traçant pour des valeurs particulières des 4 paramètres.
8- Mettre en place la méthode des moindres carrés. Pour cela, utiliser la fonction **`curve_fit`**.
Le choix des paramètres de départ de l'ajustement est important lorsque les unités sont très différentes, les paramètres de départ par défaut étant pris égaux à 1. Comment initialiser correctement l'ajustement ?
La taille d'un pixel est de 4,65 µm. Quelles sont les valeurs des tailles de faisceau en x et en y données par cette fonction ?

Travail à réaliser sur l'ensemble des images

Vous avez le droit ici d'utiliser une boucle **`for`** et une seule !

1- Répéter les opérations précédentes pour l'ensemble des 15 images.
2- Récupérer dans un tableau les valeurs de ω_x et ω_y.
3- Créer une fonction **`rayon`** définie par :

$$\omega_0\sqrt{1+\left(\frac{zM^2\lambda}{\pi\omega_0^2}\right)^2}$$

Cette fonction `rayon` possède deux paramètres ajustables, ω_0 et M^2 et une variable de position z. On rappelle que cette position a été récupérée dans le fichier **data.csv**.
Tester cette fonction `rayon` en la traçant.
4- Mettre en place la méthode des moindres carrés avec la fonction **`curve_fit`** pour trouver ω_0 et M^2 dans les deux directions x et y. La longueur d'onde du faisceau laser est de 1,3 µm.

Analyse critique des résultats

1- Que pourrait-il se passer lorsque le fond des images n'est globalement pas égal à zéro ?

2- En regardant attentivement les images, on peut constater que le faisceau est légèrement elliptique et que les axes de l'ellipse ne sont pas exactement le long de x et de y. Quelle peut être l'influence sur les résultats ? Comment pourrait-on améliorer la méthode ?

Annexes

Faisceau gaussien

Un faisceau gaussien est défini par une répartition d'intensité possédant la forme suivante (donnée ici en une dimension) :

$$I(x) = I_0(z)exp\left(-2\frac{x^2}{\omega(z)^2}\right)$$

$\omega(z)$ est le rayon du faisceau à 1/e² de l'intensité maximale $I_0(z)$.

L'évolution longitudinale du rayon d'un faisceau gaussien est donnée par :

$$\omega(z) = \omega_0\sqrt{1 + \left(\frac{z\lambda}{\pi\omega_0^2}\right)^2}$$

ω_0 est le waist du faisceau, c'est son rayon minimum. λ est la longueur d'onde du faisceau laser.

Lorsque le faisceau n'est pas limité par la diffraction, l'évolution longitudinale peut être décrite par :

$$\omega(z) = \omega_0\sqrt{1 + \left(\frac{zM^2\lambda}{\pi\omega_0^2}\right)^2}$$

Le facteur M^2 permet de mesurer l'écart à la limite de diffraction.

Barycentre d'une image

Pour calculer les coordonnées du barycentre d'une image de n lignes par m colonnes, on réalise une moyenne pondérée par l'intensité des pixels par ligne et par colonne $I(i,j)$.

Ainsi :

$$x_{bary} = \frac{\sum_{j=1}^{m}\sum_{i=1}^{n} i \times I(i,j)}{\sum_{j=1}^{m}\sum_{i=1}^{n} I(i,j)}$$

$$y_{bary} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} j \times I(i,j)}{\sum_{j=1}^{m}\sum_{i=1}^{n} I(i,j)}$$

Méthode des moindres carrés

Un problème très courant en physique est d'ajuster un modèle physique sur un nuage de points expérimentaux. Le modèle physique peut posséder un ou plusieurs paramètres inconnus p_i. « Ajuster » consiste à déterminer le jeu de paramètres qui permet de reproduire au mieux les données

expérimentales. Si ces paramètres p_i ont un sens physique, on a alors une estimation indirecte de leur valeur.

En d'autres termes, l'expérience produit n points de mesures, par exemple sous forme de couples (x_1, y_1) … (x_n, y_n). On souhaite vérifier un modèle décrit par une formule mathématique de la forme :

$$y = f(x, p_i)$$

Le but est donc de trouver les p_i tels que la fonction f représente au mieux les points expérimentaux.

On introduit une fonction χ^2 de ces paramètres p_i telle que :

$$\chi^2(p_i) = \sum_{k=1}^{n} [f(x_k, p_i) - y_k]^2$$

χ^2 est la somme quadratique des écarts des mesures au modèle, d'où le nom de cette méthode. On peut montrer que cette fonction χ^2 est minimale lorsque les paramètres p_i sont les plus proches du modèle physique décrit par la fonction f. L'ajustement des paramètres se fait donc à l'aide d'une recherche de minimum de la fonction χ^2, en général dans un espace à plusieurs dimensions. Il s'agit d'un problème non trivial, que nous ne démontrons pas ici. Heureusement, il existe une fonction sous Python qui effectue le travail pour nous ! Il s'agit de la fonction **`curve_fit`**, décrite dans la suite de l'énoncé.

Fonctions à utiliser dans le projet

- Dans la bibliothèque **`matplotlib`** :

`plt.imread(nom)` : lit une image depuis le fichier `nom` et renvoie un tableau sur cette image.

`plt.imshow(image)` : affiche les données contenues dans le tableau image sous forme d'une image.

`plt.subplot(nb lignes, nb colonnes, n° de la figure)` : affiche plusieurs graphiques sur une même figure.

`plt.axvline(x =` …`)` ou **`plt.axhline(y =` …`)`** : affiche une ligne verticale ou horizontale.

`plt.xlabel` ou **`plt.ylabel`** : affiche le titre d'un axe sur un graphique.

`plt.scatter` : affiche des points.

- Dans la bibliothèque **`scipy.optimize`** :

`scipy.optimize.curve_fit(f,xdata,ydata,p0=` …`,`…`)` : cette fonction possède un grand nombre d'arguments, nous ne nous servirons que des 4 premiers. **`f`** est la fonction qui traduit le phénomène physique, **`xdata`** est un vecteur qui contient les abscisses des données expérimentales, **`ydata`** est un vecteur qui contient les ordonnées des données correspondantes. **`p0`**est un vecteur qui contient les valeurs de départ des paramètres de l'ajustement. Par défaut, tous les paramètres de départ sont mis à 1, ce qui peut poser problème lorsque les unités sont très différentes.

Pour utiliser **`curve_fit`**, la loi physique traduite par la fonction **`f`** doit être déclarée avec comme premier argument le paramètre variable (x) et comme arguments suivants les différents paramètres que l'on souhaite optimiser.

- Dans la bibliothèque **`numpy :`**

`np.meshgrid`() : lorsque l'on manipule des images, on a souvent besoin de définir des fonctions de deux variables x et y, par exemple lorsqu'il faut accéder à l'intensité lumineuse en différents points de l'image. Il est naturel de penser le problème sous la forme d'une valeur (l'intensité) associée à une abscisse x et une ordonnée y. Calculer la fonction numériquement est un peu plus compliqué : il faut que pour chaque abscisse x, on calcule le vecteur image de y. Un moyen simple et pratique de faire cette opération est de créer un tableau de vecteurs lignes représentant les x autant de fois qu'il y a de points selon y et un tableau de vecteurs colonnes représentant les y autant de fois qu'il y a des points sur x. Les deux tableaux représentent ainsi toutes les coordonnées des points de discrétisation. Ces tableaux sont créés grâce à la fonction **`meshgrid`**.

Évaluation du projet en séance 4

Ce projet comporte 14 questions. Lors de la séance d'évaluation, vous nous présenterez les réponses aux questions, numériques ou graphiques, soit sous forme de Notebook Jupyter soit dans un fichier texte (word ou autre) à part.

Vous serez également évalué·e sur votre code selon les critères habituels (cf grille d'auto-évaluation).