POLYTECHNIQUE MONTRÉAL

UNIVERSITÉ D'INGÉNIERIE

PHS3910 – Techniques expérimentales et instrumentation

Groupe : 02

Mandat 2 : Spectromètre

Travail préparatoire

Présenté à
Jean Porvost

Par l'équipe M1 :
Germain Desloges (2139895)
Tom Dessauvages (2133573)
Aurélie Gonthier-Théberge (1998404)
Salma Moussif (2085715)

23 octobre 2024
Département de Génie physique
Polytechnique Montréal

1 Introduction

La conception d'un spectromètre se fait d'abord par la caractérisation du système optique utilisé. Ce système comprend entre autres un réseau de diffraction blazé entre deux lentilles dans une configuration 4F. Dans ce premier rapport, l'objectif est d'identifier tous les paramètres pertinents qui permettent d'observer un gamme de lumière visible entre 400 et 700 nm et de quantifier leur impact sur la résolution optique. Les paramètres discutés sont : l'ordre de diffraction, les longueurs focales des lentilles, la taille et le nombre de fentes, le pas du réseau de diffraction, la dimension des pixels, le chromatisme des lentilles et plus. L'optique de Fourier sera exploitée au travers du rapport pour atteindre l'objectif posé. Dans ce qui suit, les modèles mathématiques ainsi que leur applications en language Python seront présentés. Les signaux spectraux seront ensuite présentés dans les résultats. Puis, une analyse de ses résultats permettront de discuter et d'évaluer une paramétrisation optimale des variables du spectromètre conçu.

2 Méthodes

2.1 Optimisation des paramètres du système

Pour la suite de cette section, l'approximation paraxiale est appliquée tel que suit.

$$\sin\theta \sim \theta,\ \tan\theta \sim \theta \text{ et } \cos\theta \sim 1$$

Cette approximation est valable car comme vu sur la table optique de la figure 1 les lentilles sont très proches au réseau de diffraction par rapport à leur distances focales, ce qui veux dire que les angles impliqués sont assez petits. Cette approximation a tout de même un rôle dans les limitations de l'expérience discutées plus tard. M_1 et M_2 sont les miroirs du système, L_1 et L_2 sont les lentilles convergentes de longueurs focales f_1 et f_2, a est la dimension de la fente, θ_B est l'angle de sortie du réseau de diffraction blasé et Λ est le pas périodique des fentes de ce même réseau de diffraction. [4]

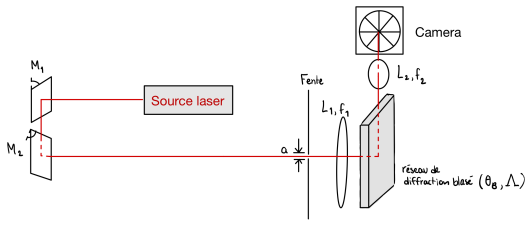

FIGURE 1 – Schéma simplifié de la table optique

En analysant les composantes du système 4F de la figure 2, il est possible de trouver l'équation d'intensité au détecteur de la caméra. Il est à noter que contrairement à ce que propose la figure 2, les lentilles n'ont pas la même distance focale.

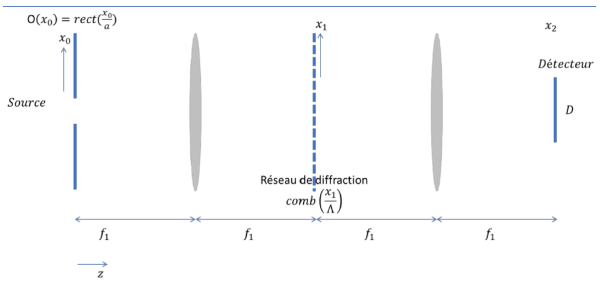

FIGURE 2 – Schéma du corrélateur 4F provenant du procédurier [5]

Après la fente a et avant la première lentille, on peut écrire :

$$U_0(x) = rect\left(\frac{x}{a}\right)$$

Entre la lentille L_1 et le réseau de diffraction :

$$U_1(x) = \mathcal{F}\left\{rect\left(\frac{x}{a}\right)\right\}\left(\frac{x}{\lambda f_1}\right) = sinc\left(\frac{xa}{\lambda f_1}\right)$$

Entre le réseau de diffraction blazé et la seconde lentille :

$$U_d(x) = sinc\left(\frac{xa}{\lambda f_1}\right) * \left[comb\left(\frac{x}{\Lambda}\right) \cdot rect\left(\frac{x}{\Lambda}\right)\right]$$

Après L_2, donc reçu sur l'écran :

$$\begin{aligned} U_2(x) &= \mathcal{F}\{U_d(x)\}\left(\frac{x}{\lambda f_2}\right) \\ &= rect\left(\frac{x f_1}{a f_2}\right) * \left(comb\left(\frac{x\Lambda}{\lambda f_2}\right) sinc\left(\frac{x\Lambda}{\lambda f_2}\right)\right) \end{aligned}$$

En se limitant à l'ordre 1 du spectre de diffraction, l'équation devient :

$$U_2(x) = rect\left(\frac{(x - \frac{\lambda f_2}{\Lambda})f_1}{af_2}\right) sinc\left(\frac{x\Lambda}{\lambda f_2}\right)$$

La fonction $rect(x)$ définit la position et la largeur du réseau diffracté dans le plan de la caméra en fonction des facteurs : $\frac{\lambda f_2}{\Lambda}$ et $\frac{af_2}{f_1}$. Pour répondre aux contraintes du spectromètre présenté, elle doit donc satisfaire deux conditions principales. La première, lié à l'étendue du spectre observable, peut être modélisé en considérant la distance maximale entre deux pics comme étant :

$$\Delta x = f_2\left(\frac{\Delta\lambda}{\Lambda} + \frac{a}{f_1}\right) \tag{1}$$

L'étendue du spectre observable par la caméra est donc calculable en prenant d = Δx, où d en est la largeur selon la direction x. Dans le cas de ce projet, d est une valeur connue et l'observable d'intérêt est la lumière visible, soit : $\Delta\lambda = 300nm$. Ainsi, les seules variables de cette équation sont les longueurs focales des deux lentilles et la largeur de la fente dans la direction x. La seconde, est-elle liée au calcul de la résolution du système et se sépare elle-même en deux sous conditions. La première est que deux rectangles consécutifs ne doivent pas superposer afin de ne pas créer de fausse information, où les intensités de deux longueurs d'ondes seraient superposées. Ce qui peut se traduire mathématiquement comme :

$$d_{pixel} \leq \frac{af_2}{f_1} \tag{2}$$

La seconde, directement liée est que la largeur d'un pic d'intensité ne doit pas être plus petit qu'un pixel, d'abord car cela ne servirait à rien en termes de résolution, mais aussi car pour deux pics collés, on aurait alors une superposition de leur intensité lors de la lecture du pixel. Géométriquement cela veut dire que la distance entre deux pics consécutifs doit toujours être plus grande ou égale à la somme des demi-largeurs des deux pics en question. Pour des largeurs de pics constantes :

$$\frac{\Delta\lambda f_2}{\Lambda} \geq \frac{af_2}{f_1}$$

Dans un cas idéal, cette équation est une égalité. On cherche donc à avoir :

$$r = \Delta\lambda = \Lambda \cdot \frac{a}{f_1}$$

Ainsi, minimiser la résolution du système permet de réécrire l'équation 1 comme :

$$\Delta x \approx f_2 \cdot \frac{\Delta\lambda}{\Lambda} \tag{3}$$

Ce modèle assume néanmoins une surface des composantes optiques (lentilles et réseau de diffraction) qui est infinie. Pour prendre la taille finie des composantes, il faudrait multiplier le masque de chaque composante par une fonction marche de la même largeur que la composante. Pour une solution plus simple, on pourrait se contenter de dire que l'intensité lumineuse ainsi perdue est négligeable si un faisceau gaussien passant par l'ouverture au premier plan focal ne déborde pas hors de la première lentille.

3 Résultats

Les valeurs caractéristiques du système sont : d = 4.968 mm, Λ = 1/600 mm et $d_{pixels} = 5.2\ \mu m$. En reprenant l'équation 3 et en considérant donc une résolution faible, la valeur de f_2 optimale est donnée par :

$$f_2 = \frac{4.958}{600 * 300 * 10^{-6}} \approx 27.54mm$$

La figure 3 présente sous la forme d'un graphique l'évolution de la dispersion spectrale en fonction de la longueur focale de la lentille 2.

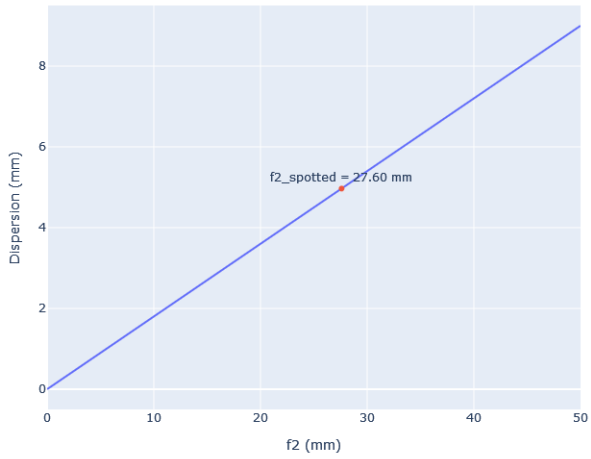

FIGURE 3 – Evolution de la dispersion spectrale en fonction de la longueur focale de la lentille 2

Pour ce qui est de la résolution, la figure 4 présente sous la forme d'un graphique l'évolution de la résolution en fonction de la distance focal de la lentille 1 et de la largeur de la fente dans la direction x.

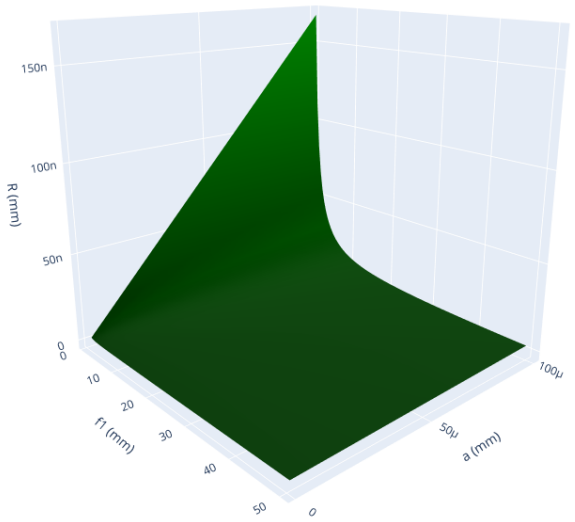

FIGURE 4 – Evolution de la dispersion spectrale en fonction de la longueur focale de la lentille 2

Ainsi plus la valeur de f_1 sera grande et celle de a petite, meilleure sera la résolution. Toujours en prenant en compte la condition de l'équation 2. Finalement, la figure 5 présente la figure de diffraction obtenue pour $f_2 \approx 27.54$ mm.

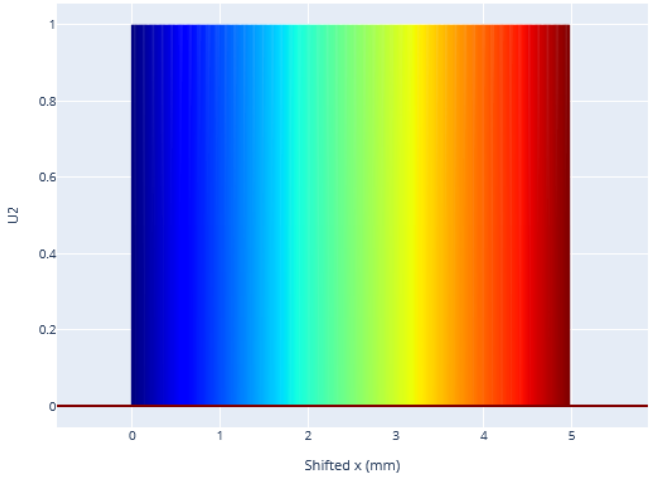

FIGURE 5 – Figure de diffraction obtenue pour $f_2 \approx 27.54$ mm

4 Discussion

4.1 Dispositif d'entrée

Pour optimiser la résolution en longueur d'onde, la taille d'ouverture préférable est une ouverture ayant une largeur de a la plus petite possible. Ne rien utiliser en entrée correspond à avoir une ouverture d'une largeur infinie, et est donc à éviter. La caméra possède une valeur de bruit intrinsèque, le "Read Noise", donc plus l'ouverture est petite, plus l'énergie lumineuse transmise à la caméra est faible, et ainsi plus le ratio signal sur bruit est faible pour la mesure d'intensité sur un pixel. Aussi, plus la largeur de la fente est petite, plus la divergence du faisceau sera élevée, ce qui peut mener à des pertes supplémentaires d'énergie si le faisceau devient plus large que les lentilles ou que le réseau de diffraction. Ces deux objectifs (minimiser la résolution en longueur d'onde et maximiser le ratio signal sur bruit de la détection de lumière) sont donc en tension. La largeur a a seulement besoin d'être minimisée en x pour améliorer la résolution en longueur d'onde, la forme d'ouverture optimale en entrée est donc une fente verticale.

4.2 Distance focale choisie pour la lentille 1

Plus la distance focale f_1 de la première lentille est élevée, plus la résolution en longueur d'onde du spectromètre est optimale. On veut donc une distance focale très élevée. Cependant, plus f_1 est élevé, plus le spectromètre sera grand, ce qui augmente le prix de fabrication et diminue la praticité de l'appareil. Au vu des composants disponibles dans le laboratoire de cours, une lentille de 50 mm de longueur focale parait donc être la solution la plus appropriée.

4.3 Distance focale choisie pour la lentille 2

Le but est de faire correspondre la dispersion (l'écart entre les pics de 400nm et 700nm) à la largeur du récepteur de la caméra. Pour ce faire, une longueur focale de 27.6mm pour la deuxième lentille du système 4f donne le résultat voulu. Des lentilles de cette distance focale précise ne sont pas aisément accessibles, et il est préférable que l'entièreté du spectre soit capté par le récepteur de la caméra, la focale la plus appropriée est donc de 25mm. L'évolution de la dispersion en fonction de f_2 présentée dans la figure 3 étant positive, cette valeur choisies, inférieur à la valeur théorique calculée, conforte ainsi l'approximation effectué dans l'équation 3.

4.4 Limitations et paramètres négligés

4.4.1 Approximation paraxiale

L'approximation paraxiale a été utilisée tout au long des démarches de ce rapport. Si le faisceau a une divergence élevée, il peut éclairer une large surface des

optiques, y compris des points loin de l'axe optique. Il est donc aussi pertinent de ne pas trop réduire la largeur a de la fente en entrée pour cette raison.

4.4.2 Chromatisme des lentilles

Le chromatisme des lentilles, c'est à dire la dépendance de la longueur focale en fonction de la longueur d'onde de la lumière, n'a pas été pris en compte. Les lentilles aisément accessibles comme celles de Thorlabs n'ont pas un chromatisme très fort, et l'écart de longueurs d'onde étudié, de 400nm à 700nm, est relativement faible. Il est donc raisonnable de négliger cet effet.

4.4.3 Faisceau non-gaussien

Il a aussi été assumé que le profil d'intensité du faisceau lumineux en entrée du système optique correspondait à un faisceau gaussien. En pratique, pour des sources de lumière naturelles et pour des lasers de piètre qualité, le profil d'intensité n'est généralement pas gaussien.

Références

[1] « Visible Ruled Reflective Diffraction Gratings. » (), adresse : https://www.thorlabs.com/newgrouppage9.cfm?objectgroup_id=8626&pn=GR25-0605 (visité le 27/10/2024).

[2] « Visible Ruled Reflective Diffraction Gratings. » (), adresse : https://www.thorlabs.com/catalogpages/obsolete/2019/DCC3260C.pdf (visité le 27/10/2024).

[3] « An Introduction to a Spectrometer - Diffraction Grating. » (), adresse : https://www.azom.com/article.aspx?ArticleID=13368 (visité le 27/10/2024).

[4] J. Harvey et R. Pfisterer, *Understanding diffraction grating behavior : including conical diffraction and Rayleigh anomalies from transmission gratings*, août 2019. adresse : https://www.researchgate.net/publication/335454256_Understanding_diffraction_grating_behavior_including_conical_diffraction_and_Rayleigh_anomalies_from_transmission_gratings.

[5] *Mandat 2*, 2024. adresse : https://moodle.polymtl.ca/pluginfile.php/1030673/mod_resource/content/2/PHS3910-Mandat2_2024.pdf.

```python
# Annexe code python

import numpy as np
import plotly.graph_objs as go
from plotly.subplots import make_subplots
import matplotlib.cm as cm

def setup_parameters(f2_value, f1_value=1000, aperture_size=0.070):
    d = 4.968  # mm, camera width
    blaze_angle_deg = 8 + 37/60
    blaze_angle = np.deg2rad(blaze_angle_deg)
    grooves_per_mm = 600
    pitch = 1 / grooves_per_mm  # grating pitch in mm
    wavelengths = np.linspace(400, 700, 300) * 1e-6 # wavelengths in mm
    return {
        'largeur_camera': d,
        'blaze_angle': blaze_angle,
        'wavelengths': wavelengths,
        'f1': f1_value,
        'f2': f2_value,
        'a': aperture_size,
        'pitch': pitch
    }

def calculate_dispersion(params):
    f2 = np.linspace(0, 50, 1000)  # mm
    dispersion = f2 * (max(params['wavelengths']) - min(params['wavelengths'])) / params['pitch']
    f2_spotted = params['largeur_camera'] * params['pitch'] / (max(params['wavelengths']) - min(params['wavelengths']))
    return f2, dispersion, {'f2_spotted': f2_spotted}

def calculate_resolution(params, a_ref=100e-6, f1_ref=50):
    f1 = np.linspace(1, 50, 100)  # mm
    a = np.linspace(1e-6, 100e-6, 100)  # mm
    f1_grid, a_grid = np.meshgrid(f1, a)
    resolution = params['pitch'] * a_grid / f1_grid
    resolution_f1 = params['pitch'] * a_ref / f1
    resolution_a = params['pitch'] * a / f1_ref
    return f1_grid, a_grid, resolution, resolution_f1, resolution_a

def rect(x):
    return np.where(np.abs(x) <= 0.5, 1, 0)

def calculate_u2(params, f1=1000):
    x = np.linspace(5, 12.5, 10000)
    _, _, f2_result = calculate_dispersion(params)
    f2 = f2_result['f2_spotted']
    Lambda = params['pitch']
    a = params['a']
    lambda_wavelengths = params['wavelengths']
    u2_results = []

    # Calculate the first wavelength's peak position to determine shift
    first_lambda = lambda_wavelengths[0]
    u2_rect = rect((x - (first_lambda * f2 / Lambda)) * f1 / (a * f2))
    u2_sinc = np.sinc(x * Lambda / (first_lambda * f2))
    u2_first = np.abs(u2_rect * u2_sinc) ** 2
    peak_index = np.argmax(u2_first)
    shift_value = x[peak_index]  # Calculate the x-shift needed

    # Shift x by the peak position of the first wavelength
    shifted_x = x - shift_value

    for lambda_ in lambda_wavelengths:
        u2_rect = rect((x - (lambda_ * f2 / Lambda)) * f1 / (a * f2))
        u2_sinc = np.sinc(x * Lambda / (lambda_ * f2))
        u2 = np.abs(u2_rect * u2_sinc) ** 2
        norm_u2 = u2 / np.max(np.abs(u2))
        u2_results.append(norm_u2)

    return shifted_x, u2_results, lambda_wavelengths

def main():
    params = setup_parameters(f2_value=11, f1_value=25)
    f2, dispersion, f2_result = calculate_dispersion(params)
    f2_spotted = f2_result['f2_spotted']

    # Dispersion plot
    fig = go.Figure()
    fig.add_trace(go.Scatter(x=f2, y=dispersion, mode='lines', name="Dispersion curve"))
```

```
fig.add_trace(go.Scatter(x=[f2_spotted], y=[params['largeur_camera']], mode='markers+text',
                         text=f"f2_spotted = {f2_spotted:.2f} mm", textposition="top center"))
fig.update_layout(
    title="Dispersion vs f2",
    xaxis_title="f2 (mm)",
    yaxis_title="Dispersion (mm)",
    width=800,  # Width and height adjusted for square shape
    height=600
)

# Resolution plot (3D surface)

# Custom darker green colorscale
moderate_greens = [
    [0, "rgb(0, 50, 0)"],
    [0.2, "rgb(0, 80, 0)"],
    [0.4, "rgb(0, 100, 0)"],
    [0.6, "rgb(0, 120, 0)"],
    [0.8, "rgb(0, 140, 0)"],
    [1, "rgb(0, 160, 0)"]
]

f1_grid, a_grid, resolution, resolution_f1, resolution_a = calculate_resolution(params)
fig3d = go.Figure(data=[go.Surface(z=resolution, x=f1_grid,
                                   y=a_grid, colorscale=moderate_greens)])
fig3d.update_layout(
    scene=dict(
        xaxis_title="f1 (mm)",
        yaxis_title="a (mm)",
        zaxis_title="R (mm)"
    ),
    title="Resolution as a function of f1 and aperture width",
    width=900,
    height=900
)

# Resolution plots for f1 and a individually
fig_f1 = go.Figure(go.Scatter(x=f1_grid[0, :], y=resolution_f1, mode='lines'))
fig_f1.update_layout(
    title="Resolution vs f1 for given aperture",
    xaxis_title="f1 (mm)",
    yaxis_title="R (mm)",
    width=800,
    height=600
)

fig_a = go.Figure(go.Scatter(x=a_grid[:, 0], y=resolution_a, mode='lines'))
fig_a.update_layout(
    title="Resolution vs aperture width for given f1",
    xaxis_title="a (mm)",
    yaxis_title="R (mm)",
    width=800,
    height=600
)

# U2 Function for different wavelengths with first peak-aligned x-axis
shifted_x, u2_results, wavelengths = calculate_u2(params)
fig_u2 = go.Figure()

# Generate colors from cm.jet for the number of wavelengths
jet_colors = [cm.jet(i / len(wavelengths)) for i in range(len(wavelengths))]

for u2, lambda_, color in zip(u2_results, wavelengths, jet_colors):
    rgba_color = f'rgba({color[0] * 255}, {color[1] * 255}, {color[2] * 255}, {color[3]})'
    fig_u2.add_trace(go.Scatter(x=shifted_x, y=u2, mode='lines',
                                name=f'λ = {lambda_ * 1e6:.0f} nm',
                                line=dict(color=rgba_color)))

fig_u2.update_layout(
    title="U2 Function for Different Wavelengths (Aligned at First Peak)",
    xaxis_title="Shifted x (mm)",
    yaxis_title="U2",
    width=800,
    height=600
)


# Display the figure
fig.show()
```

```
    fig3d.show()
    fig_a.show()
    fig_f1.show()
    fig_u2.show()

if __name__ == "__main__":
    main()
```