傅立叶变换光谱测量技术实验报告

黄润华
Ocean University of China
email@noreply.com

杨超
Ocean University of China
email@somewhere.com

摘要—本实验报告为傅立叶变换光谱测量技术第三次实验报告，本报告采用 Python 仿真对干涉波形补零与加窗环境下的傅里叶变换光谱测量系统的光谱测量曲线。中心波长选择 632.8nm。

Index Terms—Optical spectrum, python, fft

I. 实验目的

- 思考干涉图补零对傅里叶变换光谱测量曲线的影响
- 思考干涉图加不同窗后补零对傅里叶变换光谱测量曲线的影响
- 学习利用 Python 仿真不同影响条件下光谱测量曲线。

II. 实验原理

A. 分辨率和分辨能力

光谱仪的光谱分辨率是其定性区分彼此非常接近的两个光谱峰的能力的量度。通常，光谱分辨率由仪器线形状（ILS）的半峰全宽（FWHM）表示，光谱仪的输出光谱具有纯单色输入辐射。

光谱分辨率 R 由下式定义

$$R = \frac{\sigma_{max}}{\delta\sigma} \tag{1}$$

其中 σ_{max} 是光谱仪设计运行的最大波数。

1）分辨率与最大 OPD 之间的关系：根据公式

$$B_c(\sigma) = B_{reL}(\sigma) + iB_{ioL}(\sigma) = \int_{-\infty}^{\infty} I(x)\,\mathrm{rect}(x)e^{-i2\pi ax}dx \tag{2}$$

$$\begin{cases} cB(\sigma) = 2\,|\int_{-\infty}^{+\infty} I(x)\,\mathrm{rect}(x)e^{-2\pi ax}dx \\ \phi(\sigma) = \arctan\left[\frac{B_{ioL}(\sigma)}{B_{reL}(\sigma)}\right] \end{cases} \quad (\sigma \geq 0) \tag{3}$$

其中矩形函数定义为

$$\mathrm{rect}(x) = \begin{cases} 1 & |x| < L \\ 0 & |x| > L \end{cases}$$

因此，理论上 FTS 的光谱分辨率取决于可移动镜的扫描范围

$$B_e(\sigma) = \begin{cases} llB(\sigma)/2 & (\sigma \geq 0) \\ B(-\sigma)/2 & (\sigma \leq 0) \end{cases} \tag{4}$$

理论上，完美单色线的干涉图可以用来描述

$$I(x) = 2cos(2\pi\sigma_0 x) \tag{5}$$

将此表达式替换为方程 $B(\sigma)$、$\phi(\sigma)$，考虑到实际上 $\sigma > 0$，可以得到 ILS $B(\sigma)$ 的表达式：

$$B_{IIS}(\sigma) = 2L\frac{\sin\left[2\pi\left(\sigma_0+\sigma\right)L\right]}{2\pi\left(\sigma_0+\sigma\right)L} + \frac{\sin\left[2\pi\left(\sigma_0-\sigma\right)L\right]}{2\pi\left(\sigma_0-\sigma\right)L} \tag{6}$$

$$\approx 2L \times \frac{\sin\left[2\pi\left(\sigma_0-\sigma\right)L\right]}{2\pi\left(\sigma_0-\sigma\right)L} \tag{7}$$

$$= 2L\sin c\left[2\pi\left(\sigma_0-\sigma\right)L\right] \tag{8}$$

如图 1所示，光谱从一条线 (δ 函数) 扩展到 *sinc* 函数形状 (仪器线形)。

$$(\delta\sigma)_{linewidth} = \frac{1.21}{2L} \tag{9}$$

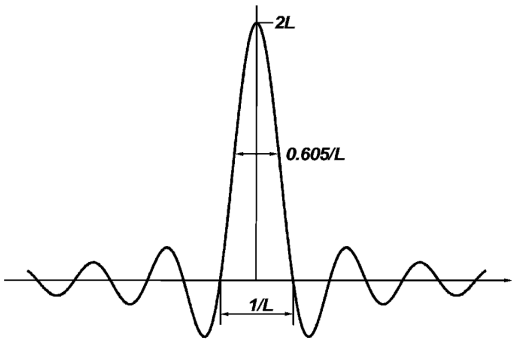

图 1. $B_{IIS}(\sigma)$

2) 分离共振: 分辨率可以通过分离图 2所示光谱中相同强度 (或共振) 的两条单色线 (波数和) 来描述。如果两个线峰值之间的下降幅度大于线峰值的 20%，则可以声称解决了两个共振，此时

$$(\delta\sigma)_{separation} = \frac{1.46}{2L} \tag{10}$$

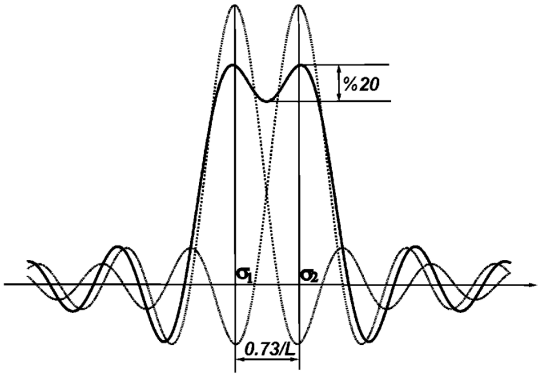

图 2. 分离共振解析结果

3) 瑞利准则: 瑞利准则分离两个 ILS 的峰值，使得一个共振的最大值落在另一个共振的零点处，因此，仪器线形状决定的分辨率是

$$\delta\sigma = \frac{1}{2L} \tag{11}$$

B. 分辨率与发散角度之间的关系

图 3是迈克尔逊干涉仪的等效图，根据图 3与图 4可以计算出 OPD 与夹角 θ 之间的关系

$$OPD = 2 \times \frac{x/2}{\cos\theta} - x\tan\theta\sin\theta = x\cos\theta \tag{12}$$

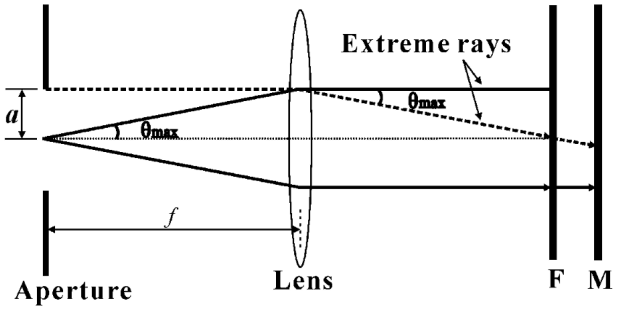

图 3. 迈克尔逊干涉仪的等效图

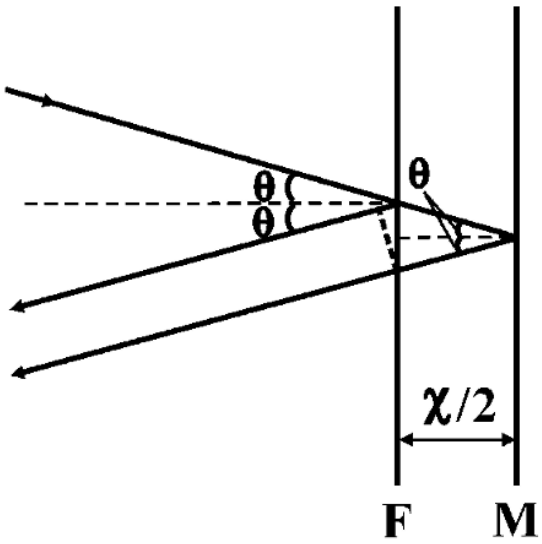

图 4. 迈克尔逊干涉仪接收端放大图

我们可以得到扩展源的归一化干涉图，对着一个立体角 Ω_{max} 是

$$I(x,\Omega_{\max}) = \frac{1}{\Omega_{\max}}\int_0^{\infty} B(\sigma)\int_0^{\Omega_{\text{mix}}} \cos(2\pi\sigma x\cos\theta)d\Omega d\sigma$$

根据立体角的公式，上述式子可以化简为

$$I(x,\Omega_{\max}) = \int_0^{\infty} B(\sigma)sinc\frac{\Omega_{\max}\sigma x}{2}\cos[2\pi\sigma x - \frac{\Omega_{\max}\sigma x}{2}]d\sigma$$

对于单色光源其满足

$$B(\sigma) = \delta(\sigma - \sigma_0)$$

其仪器线轮廓可以写为

$$\begin{cases} B_{Div-1LS}(\sigma) = [\pi/(\sigma_0\Omega_{\max})]\,\text{rect}(\sigma_1,\sigma_2) \\ \qquad\qquad +[\pi/(\sigma_0\Omega_{\max})]\,\text{rect}(-\sigma_2,-\sigma_1) \\ \sigma_2 = \sigma_0 - \sigma_0\Omega_{\max}/2\pi \\ \sigma_1 = \sigma_0 \end{cases}$$

于是可以得到

$$B_{Div-1LS}(\sigma) = [\pi/(\sigma_0\Omega_{\max})]\,\text{rect}(\sigma_1,\sigma_2) \tag{13}$$

$$\bar{\sigma} = \sigma_0[1 - (\Omega_{\max}/4\pi)] \tag{14}$$

总的波数差或分辨率是

$$\delta\sigma = \frac{\sigma_0 \Omega_{max}}{2\pi}$$

考虑到可移动镜的扫描长度有限，实际 FTS 的 ILS 应该是

$$B_{ILS}(\sigma) = B_{Div-ILS}(\sigma) * 2L \sin c[2\pi\sigma L] \tag{15}$$

其中 L 是可移动镜的最大位移，$*$ 是卷积算子。图 4中两个极端射线的 OPD 之间的差异公式如下:

$$\Delta OPD = 2L - 2L\cos\theta_{\max} \approx L\theta_{\max}^2 = L\frac{a^2}{f^2}$$

其中 a 是入口孔径的半径，f 是准直器的焦距。当 ΔOPD 等于 $\lambda/2$ 时，两条光线异相，它们之间会发生破坏性叠加。对于宽带辐射输入，存在的最短波长决定了 FTS 的 ΔOPD 的最大有效值，如下式所示:

$$\Delta OPD \leq \frac{\lambda_{min}}{2} = \frac{1}{2\sigma_{max}} \tag{16}$$

我们可以得到 a 的值应满足以下不等式，以获得大于 R 的分辨率:

$$a \leq \frac{f}{\sqrt{R}} \tag{17}$$

C. 傅里叶变换光谱仪的优点

与色散光谱仪相比，FTS 具有许多广泛宣传的优点。然而，只有吞吐量（Jacquinot）和多路复用（Felgett）优势是 FTS 操作原理所固有的，而不是特定的工程设计。

1) 吞吐量或 Jacquinot 优势: 吞吐量优势是 FTS 可以具有大的圆形入口孔，其面积比分散光谱仪中的狭缝的面积大得多，以获得相同的分辨率。吞吐量定义为，其中是限制孔径的面积，并且是图 5中所示的准直或聚焦光学器件所对应的立体角。

FTS 的最大吞吐量由下式给出

$$T_{FTS} = A\Omega_s = \frac{\pi}{R}A_{mirror} \tag{18}$$

其中 A_{mirror} 是镜子的投影面积。

对于光栅光谱仪，实现分辨率 R 的能量吞吐量受其狭缝区域和准直光学系统的限制，并由下式给出:

$$T_{grating} = \frac{l}{f_g R}A_{grating} \tag{19}$$

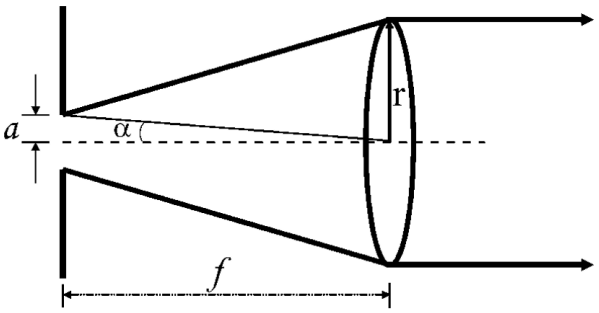

图 5. 孔径和立体角的关系图

其中 f_g 是准直光学系统的焦距，l 是狭缝的高度，$A_{grating}$ 是光栅的投影面积。对于光栅光谱仪，$\frac{l}{f_g R}$ 小于 1/20。

因此，对于相同的分辨率和类似的仪器尺寸，FTS 比光栅光谱仪具有高 60 倍的能量收集能力。因此，Jacquinot 的优势使得 FTS 更适合于弱信号测量，其中检测器噪声占主导地位，频谱信噪比（SNR）与吞吐量成比例地增加。

2) Multiplex 或 Felgett 的优势: 多路复用的优点是 FTS 在扫描周期内同时观察来自给定频谱的整个范围的所有频谱信息。因此，通过使用 FTS 和使用单色仪可获得的 SNR 的比率:

$$\frac{SNR_{FTS}^S}{SNR_{monochromator}} = N^{\frac{1}{2}} \tag{20}$$

FTS 的多重优势仅存在于红外和远红外信号的测量中，并且在可见 - 紫外信号的检测中丢失，因为在红外检测中，与信号电平无关的检测器噪声占主导地位。量子噪声在可见光 - 紫外信号检测中占主导地位。

3) Connes 的优势: FTS 的波数范围来自 He-Ne 激光条纹，其作为每次扫描中采样位置的内部参考。这种激光的波数非常准确，并且非常稳定。因此，干涉仪的波数校准更精确，并且具有比分散仪器的校准好得多的长期稳定性.

III. 实验内容

1. 仿真对于干涉图补零后的傅里叶变换光谱测量曲线，观察其变化。
2. 仿真对于干涉图加窗函数并补零后的傅里叶变换光谱测量曲线，观察期变化。（以 632.8nm 的 He-Ne 激光为例，光程差采样间隔为 79.1nm）

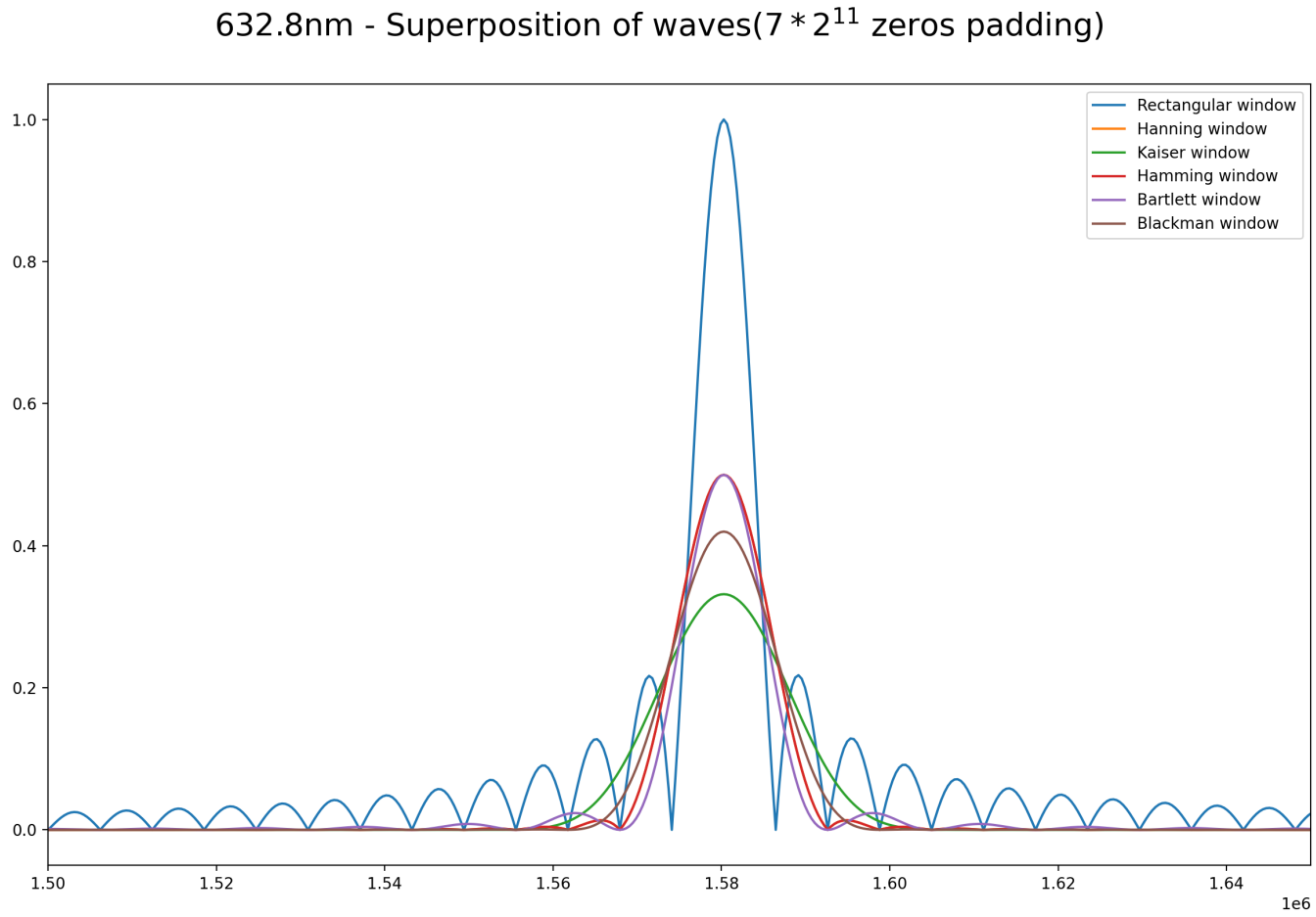

图 6. 加窗后与未加窗的光谱测量曲线的比较

IV. 实验结果

A. 加窗实验结果

图片 6显示了加窗后与未加窗的光谱测量曲线的比较结果。加窗前与加窗后干涉图补零的个数均为 7×2^{11} 个。更详细具体的加窗图片如图 7至 11所示，每张图片显示了对干涉图加特殊窗后与对干涉图加矩形窗（未加窗）的比较结果。图片左侧显示了干涉图加窗的波形，右侧图片为光谱测量曲线的波形，右侧黄色部分的线表示光谱测量曲线的半波全宽。

图 7至 11表明对干涉图加窗函数后将导致原先光谱测量曲线的幅值部分下降，其中加 Kaiser 窗对光谱测量曲线的幅值影响最大，但 Kaiser 窗对光谱测量曲线的边频分量的抑制效果最佳。

对干涉图加窗函数还将导致光谱测量曲线半波全宽变宽的结果。几乎所有窗函数处理后的干涉图的光谱

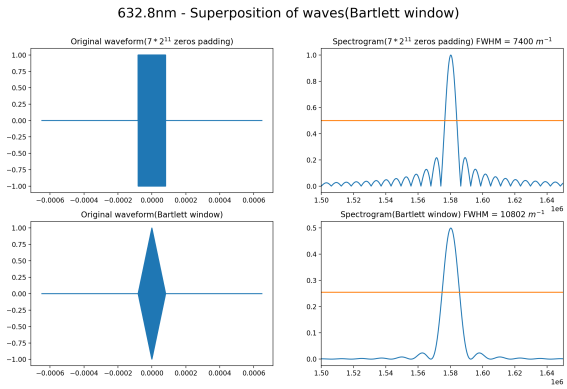

图 7. 对干涉图分别加 Bartlett 窗与矩形窗的比较，图片左侧显示了干涉图加窗的波形，右侧图片为光谱测量曲线的波形，右侧黄色部分的线表示光谱测量曲线的半波全宽

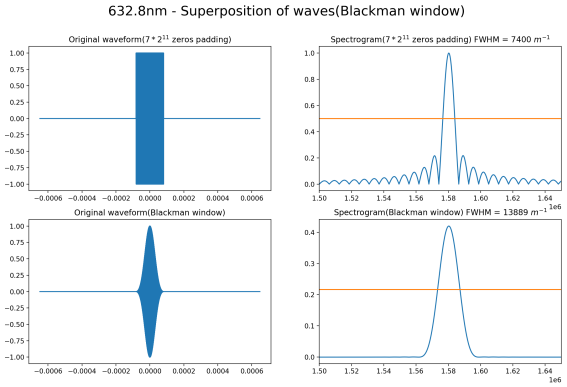

图 8. 对干涉图分别加 Blackman 窗与矩形窗的比较，图片左侧显示了干涉图加窗的波形，右侧图片为光谱测量曲线的波形，右侧黄色部分的线表示光谱测量曲线的半波全宽

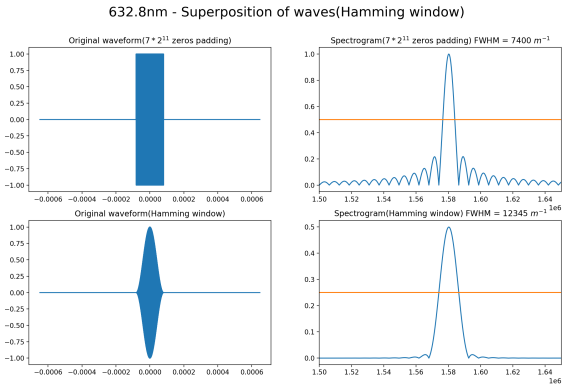

图 9. 对干涉图分别加 Hamming 窗与矩形窗的比较，图片左侧显示了干涉图加窗的波形，右侧图片为光谱测量曲线的波形，右侧黄色部分的线表示光谱测量曲线的半波全宽

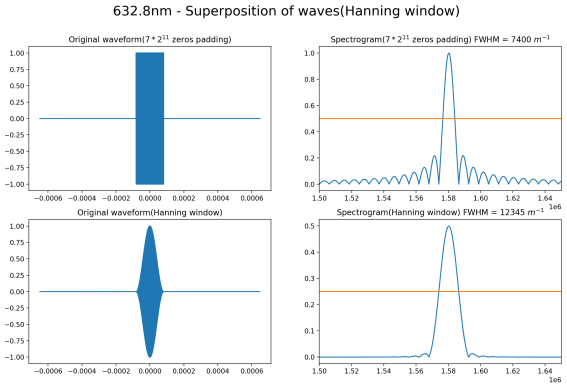

图 10. 对干涉图分别加 Hanning 窗与矩形窗的比较，图片左侧显示了干涉图加窗的波形，右侧图片为光谱测量曲线的波形，右侧黄色部分的线表示光谱测量曲线的半波全宽

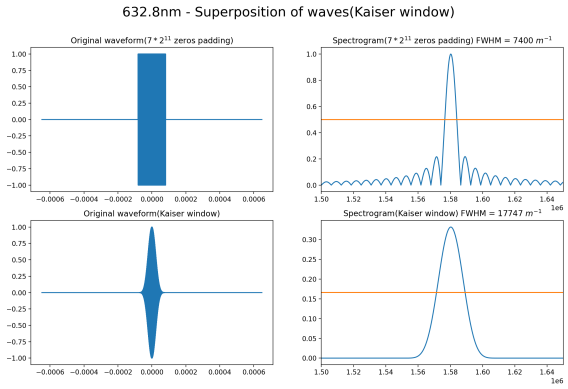

图 11. 对干涉图分别加 Kaiser 窗与矩形窗的比较，图片左侧显示了干涉图加窗的波形，右侧图片为光谱测量曲线的波形，右侧黄色部分的线表示光谱测量曲线的半波全宽

测量曲线的半波宽度均大于使用矩形窗函数（未加窗）处理后的干涉图的光谱测量曲线的半波宽度（图片 6），其中加 Kaiser 窗对光谱测量曲线的半波宽度影响最大，Hamming 窗与 Bartlett 窗对光谱测量曲线的半波宽度影响较小，但 Bartlett 窗对光谱测量曲线的边频分量的抑制效果较差。

对干涉图加窗函数的优势在于通过加窗后，干涉图对应的光谱测量曲线的边频分量可以得到有效的抑制。用 Kaiser 窗处理后的干涉图对应的光谱测量曲线的边频分量的抑制效果最为明显，Balckman 窗处理后的效果次之。

B. 补零实验结果

图片 12显示了补 2^{11} 个零与未补零的光谱测量曲线的比较结果。图片 13显示了补 3×2^{11} 个零与未补零的光谱测量曲线的比较结果。图片 14显示了补 7×2^{11} 个零与未补零的光谱测量曲线的比较结果。三个图中，黄色的线代表光谱测量曲线的半波全宽。

图 12，图 13与图 14显示了对干涉图补零不会影响光谱测量曲线的半波全宽，从频域角度分析，对干涉图补零不会影响频谱的分辨率，补零操作增大了系统的采样点个数，这种操作对 FFT 分辨率有较好的提升，同时带来了频域波形更加光滑的结果。

C. 仿真结论

通过上述对仿真结果的讨论，可以得到如下结论：

1) 补 0 不会影响分辨率，但会使旁瓣增大。
2) 补 0 的数量越多，波形越圆润。

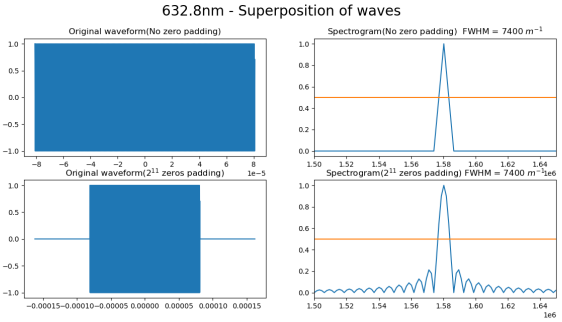

图 12. 补 2^{11} 个零与未补零光谱测量曲线的比较

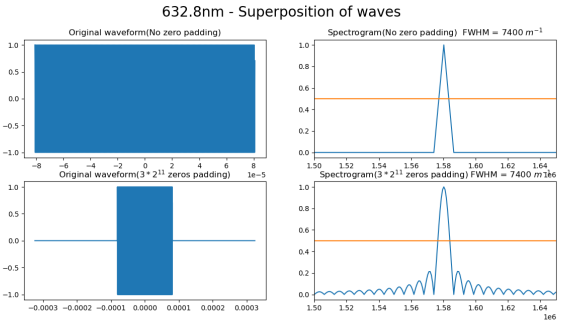

图 13. 补 3×2^{11} 个零与未补零光谱测量曲线的比较

3) 加窗可以有效减小旁瓣，但会使分辨率降低。

V. 结语

本实验完成了利用 Python 分别仿真对干涉图补零与对干涉图加窗两因素影响下的傅里叶变换光谱测量系统的光谱测量曲线的实验。

附录

实验代码

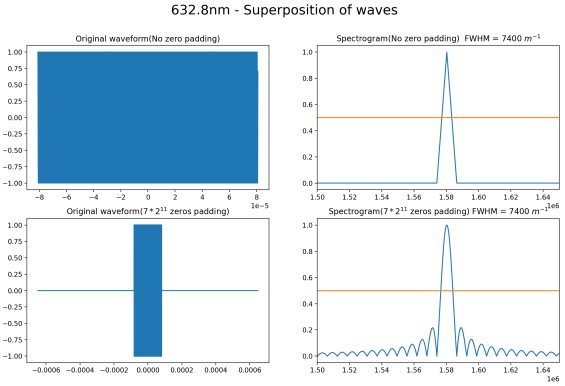

图 14. 补 7×2^{11} 个零与未补零光谱测量曲线的比较

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.fftpack import fft, ifft
from matplotlib.pylab import mpl
from pylab import *
from scipy import signal


def find_best_I_average_value(array,
    average):
    best_I_average_value = 0
    min_dif = 1
    for i in array:
        if abs(i-average) < min_dif:
            min_dif = abs(i-average)
            best_I_average_value = i

    return best_I_average_value

def find_index(array1, array2):
    first_index = 0
    last_index = 0
    current_index = 0
    for i, j in zip(array1,array2):
        if i < j:
            first_index = current_index
            break
        current_index = current_index +
            1
    array2 = array2[::-1]
    array1 = array1[::-1]
    current_index = 0
    for i, j in zip(array1,array2):
        if i < j:
            last_index = current_index
            break
        current_index = current_index +
            1
    print("array1.size - last_index -
        first_index = %d" %(array1.size
        - last_index - first_index))
```

```
    return first_index, array1.size -
        last_index

# 设置中心波长为632.8nm
laimda0 = 632.8*10**(-9)
#用79.1nm的采样间隔
i = 79.1*10**(-9)
# 设置中心点的采样频率
sigma0 = 1/laimda0

p1 = (-1)*(2**10)*79.1*10**(-9)
p2 = (2**10)*79.1*10**(-9)
p3 = (2**11)
p4 = 2**10

#####################################
# 设置干涉图的采样点
# (无补零的情况)
#####################################
x0 = np.arange(p1, p2, i)
# 计算采样点的个数
print("没有补零前x0的长度为 %d" %len(x0
    ))
# 做傅里叶的点数，取下一个2的次幂
# n0 = 2**(int(np.log2(len(x0)))+1)
n0 = len(x0)
print("n0 length = %d" %n0)
# 干涉图函数
I0=(np.cos(2*np.pi*sigma0*x0))
print("没有补零前I0的长度为 %d" %len(I0
    ))

# 这里对I0做n0个点的FFT，为了求得正确的
    幅度需要除以len(x0)
# 真实的点个数，不是除以傅立叶变换的点
    数
Y0 = 2*abs(fft(I0,n0))/len(x0)
# 得到的Y0是周期函数，为了使用 FWHM函数
    需要提取出前一半图像
Y0 = Y0[:int(n0/2)]

# 设置频谱图的横坐标
fs = 1/i*np.arange(n0/2)/n0

best_Y0_average_range = 0.5*np.ones((Y0
    .size, 1))

#####################################
# Zero padding begins here
# Case 1.
#   1*2^11 zeros
#####################################
zero_numbers_1= p3/2
x1 = np.arange((-1)*(p4+zero_numbers_1)
    *i, (p4+zero_numbers_1)*i, i)
print("补零后x1的长度为 %d" %len(x1))
# 做傅里叶的点数，取下一个2的次幂
n1 = 2**(int(np.log2(len(x1)))+1)
print("n1 length = %d" %n1)
I1=(np.cos(2*np.pi*sigma0*x0));

I1 = np.pad(I1,(0,int(zero_numbers_1)),
    'constant',constant_values = (0,0))
I1 = np.pad(I1,(int(zero_numbers_1),0),
    'constant',constant_values = (0,0))
print("补零后I1的长度为 %d" %len(I1))


Y1 = 2*abs(fft(I1,n1))/len(x0)
Y1 = Y1[:int(n1/2)]


# 设置频谱图的横坐标
fs_1 = 1/i*np.arange(n1/2)/n1

best_Y1_average_range = 0.5*np.ones((Y1
    .size, 1))

figure(1)
subplot(2,2,1)
plt.plot(x0, I0)
```

```
plt.title("Original waveform(No zero
    padding)")

subplot(2,2,2)
plt.plot(fs, Y0)
plt.plot(fs,best_Y0_average_range)
plt.title("Spectrogram(No zero padding)
     FWHM = 7400 $m^{-1}$")
plt.xlim(1.50*(10**6), 1.65*(10**6))

subplot(2,2,3)
plt.plot(x1, I1)
plt.title("Original waveform($2^{11}$
    zeros padding)")

subplot(2,2,4)
plt.plot(fs_1, Y1)
plt.plot(fs_1,best_Y1_average_range)
plt.title("Spectrogram($2^{11}$ zeros
    padding) FWHM = 7400 $m^{-1}$")
plt.xlim(1.50*(10**6), 1.65*(10**6))

plt.suptitle("632.8nm - Superposition
    of waves", fontsize = 20)

##############################
# Zero padding case 2.
#   3*2^11 zeros
##############################
zero_numbers_2= 3*p3/2
x2 = np.arange((-1)*(p4+zero_numbers_2)
    *i, (p4+zero_numbers_2)*i, i)
print("补零后x1的长度为 %d" %len(x1))
# 做傅里叶的点数, 取下一个2的次幂
n2 = 2**(int(np.log2(len(x2)))+1)
print("n2 length = %d" %n2)
I2=(np.cos(2*np.pi*sigma0*x0));

I2 = np.pad(I2,(0,int(zero_numbers_2)),
    'constant',constant_values = (0,0))
I2 = np.pad(I2,(int(zero_numbers_2),0),
    'constant',constant_values = (0,0))
print("补零后I2的长度为 %d" %len(I2))


Y2 = 2*abs(fft(I2,n2))/len(x0)
Y2 = Y2[:int(n2/2)]

# 设置频谱图的横坐标
fs_2 = 1/i*np.arange(n2/2)/n2

best_Y2_average_range = 0.5*np.ones((Y2
    .size, 1))

figure(2)
subplot(2,2,1)
plt.plot(x0, I0)
plt.title("Original waveform(No zero
    padding)")

subplot(2,2,2)
plt.plot(fs, Y0)
plt.plot(fs,best_Y0_average_range)
plt.title("Spectrogram(No zero padding)
     FWHM = 7400 $m^{-1}$")
plt.xlim(1.50*(10**6), 1.65*(10**6))

subplot(2,2,3)
plt.plot(x2, I2)
plt.title("Original waveform($3*2^{11}$
     zeros padding)")

subplot(2,2,4)
plt.plot(fs_2, Y2)
plt.plot(fs_2,best_Y2_average_range)
plt.title("Spectrogram($3*2^{11}$ zeros
     padding) FWHM = 7400 $m^{-1}$")
plt.xlim(1.50*(10**6), 1.65*(10**6))

plt.suptitle("632.8nm - Superposition
    of waves", fontsize = 20)
```

```
##############################
# Zero padding case 3.
#   5*2^11 zeros
##############################
zero_numbers_3= 5*p3/2
x3 = np.arange((-1)*(p4+zero_numbers_3)
   *i, (p4+zero_numbers_3)*i, i)
print("补零后x1的长度为 %d" %len(x3))
# 做傅里叶的点数, 取下一个2的次幂
n3 = 2**(int(np.log2(len(x3)))+1)
print("n3 length = %d" %n1)
I3=(np.cos(2*np.pi*sigma0*x0));

I3 = np.pad(I3,(0,int(zero_numbers_3)),
   'constant',constant_values = (0,0))
I3 = np.pad(I3,(int(zero_numbers_3),0),
   'constant',constant_values = (0,0))
print("补零后I3的长度为 %d" %len(I3))



Y3 = 2*abs(fft(I3,n3))/len(x0)
Y3 = Y3[:int(n3/2)]

# 设置频谱图的横坐标
fs_3 = 1/i*np.arange(n3/2)/n3

best_Y3_average_range = 0.5*np.ones((Y3
   .size, 1))

figure(3)
subplot(2,2,1)
plt.plot(x0, I0)
plt.title("Original waveform(No zero
   padding)")

subplot(2,2,2)
plt.plot(fs, Y0)
plt.plot(fs,best_Y0_average_range)
plt.title("Spectrogram(No zero padding)
    FWHM = 7400 $m^{-1}$")
plt.xlim(1.50*(10**6), 1.65*(10**6))

subplot(2,2,3)
plt.plot(x3, I3)
plt.title("Original waveform($5*2^{11}$
    zeros padding)")

subplot(2,2,4)
plt.plot(fs_3, Y3)
plt.plot(fs_3,best_Y3_average_range)
plt.title("Spectrogram($5*2^{11}$ zeros
    padding) FWHM = 7400 $m^{-1}$")
plt.xlim(1.50*(10**6), 1.65*(10**6))

plt.suptitle("632.8nm - Superposition
   of waves", fontsize = 20)

figure(10)
pic1, = plt.plot(fs, Y0)
pic2, = plt.plot(fs_1, Y1)
pic3, = plt.plot(fs_2, Y2)
pic4, = plt.plot(fs_3, Y3)
plt.title("Spectrogram All in One
   Picture")
plt.xlim(1.50*(10**6), 1.65*(10**6))

plt.legend(handles=[pic1,pic2,pic3,pic4
   ],labels=['No zero padding', '$1
   *2^{11}$ zeros padding', '$3*2^{11}$
    zeros padding', '$5*2^{11}$ zeros
   padding'], loc='upper right')


plt.suptitle("632.8nm - Superposition
   of waves", fontsize = 20)


##################################
# Window function begins here
# Case1.
#   - hamming
```

```
#####################################

zero_numbers_3= 5*p3/2
x3_hamming = np.arange((-1)*(p4+
    zero_numbers_3)*i, (p4+
    zero_numbers_3)*i, i)
print("补零后x3的长度为 %d" %len(x3))
# 做傅里叶的点数, 取下一个2的次幂
n3_hamming = 2**(int(np.log2(len(
    x3_hamming)))+1)
print("n3 length = %d" %n1)
I3_hamming=(np.cos(2*np.pi*sigma0*x0))
L = len(I3_hamming)

# 定义hamming窗函数
hamming_window = np.hamming(len(
    I3_hamming))
# 对 x3加窗
I3_hamming = I3_hamming *
    hamming_window

I3_hamming = np.pad(I3_hamming,(0,int(
    zero_numbers_3)),'constant',
    constant_values = (0,0))
I3_hamming = np.pad(I3_hamming,(int(
    zero_numbers_3),0),'constant',
    constant_values = (0,0))
print("补零后I3的长度为 %d" %len(I3))




Y3_hamming = 2*abs(fft(I3_hamming,
    n3_hamming))/len(x0)
Y3_hamming = Y3_hamming[:int(n3/2)]

# 设置频谱图的横坐标
fs_3_hamming = 1/i*np.arange(n3_hamming
    /2)/n3_hamming

Y3_hamming_average = (Y3_hamming.min()+
    Y3_hamming.max())/2
print("Y3_hamming min = %24e" %
    Y3_hamming.min())
print("Y3_hamming max = %24e" %
    Y3_hamming.max())
print("Y3_hamming_average = %24e" %
    Y3_hamming_average)

best_Y3_hamming_average_value =
    find_best_I_average_value(Y3_hamming
    , Y3_hamming_average)
print("best_Y3_hamming_average_value =
    %24e" %best_Y3_hamming_average_value
    )

best_Y3_hamming_average_range =
    best_Y3_hamming_average_value*np.
    ones((Y3_hamming.size, 1))
# best_Y3_hamming_average_range = 0.5*
    np.ones((Y3_hamming.size, 1))

first_index, last_index = find_index(
    best_Y3_hamming_average_range,
    Y3_hamming)
hamming_FWHM = fs_3_hamming[last_index]
    - fs_3_hamming[first_index]
print("Hamming FWHM = %d" %int(
    hamming_FWHM))

figure(4)
subplot(2,2,1)
plt.plot(x3, I3)
plt.title("Original waveform($5*2^{11}$
    zeros padding)")

subplot(2,2,2)
plt.plot(fs_3, Y3)
plt.plot(fs_3,best_Y3_average_range)
plt.title("Spectrogram($5*2^{11}$ zeros
    padding) FWHM = 7400 $m^{-1}$")
plt.xlim(1.50*(10**6), 1.65*(10**6))
```

```
plt.suptitle("632.8nm - Superposition
    of waves(Hamming window)", fontsize
    = 20)


subplot(2,2,3)
plt.plot(x3_hamming, I3_hamming)
plt.title("Original waveform(Hamming
    window)")

subplot(2,2,4)
plt.plot(fs_3_hamming, Y3_hamming)
plt.plot(fs_3_hamming,
    best_Y3_hamming_average_range)
plt.title("Spectrogram(Hamming window)
    FWHM = %d $m^{-1}$" %int(
    hamming_FWHM))
plt.xlim(1.50*(10**6), 1.65*(10**6))


####################################
# Window function begins here
# Case2.
#   - kaiser
####################################

zero_numbers_3= 5*p3/2
x3_kaiser = np.arange((-1)*(p4+
    zero_numbers_3)*i, (p4+
    zero_numbers_3)*i, i)
print("补零后x3的长度为 %d" %len(x3))
# 做傅里叶的点数, 取下一个2的次幂
n3_kaiser = 2**(int(np.log2(len(
    x3_kaiser)))+1)
print("n3 length = %d" %n3_kaiser)
I3_kaiser=(np.cos(2*np.pi*sigma0*x0));
L = len(I3_kaiser)

# 定义kaiser窗函数
kaiser_window = np.kaiser(len(I3_kaiser
    ), 14)


# 对 x3加窗
I3_kaiser = I3_kaiser * kaiser_window

I3_kaiser = np.pad(I3_kaiser,(0,int(
    zero_numbers_3)),'constant',
    constant_values = (0,0))
I3_kaiser = np.pad(I3_kaiser,(int(
    zero_numbers_3),0),'constant',
    constant_values = (0,0))
print("补零后I3的长度为 %d" %len(I3))



Y3_kaiser = 2*abs(fft(I3_kaiser,
    n3_kaiser))/len(x0)
Y3_kaiser = Y3_kaiser[:int(n3/2)]

# 设置频谱图的横坐标
fs_3_kaiser = 1/i*np.arange(n3_kaiser
    /2)/n3_kaiser

Y3_kaiser_average = (Y3_kaiser.min()+
    Y3_kaiser.max())/2
print("Y3_kaiser min = %24e" %Y3_kaiser
    .min())
print("Y3_kaiser max = %24e" %Y3_kaiser
    .max())
print("Y3_kaiser_average = %24e" %
    Y3_kaiser_average)

best_Y3_kaiser_average_value =
    find_best_I_average_value(Y3_kaiser,
     Y3_kaiser_average)
print("best_Y3_kaiser_average_value =
    %24e" %best_Y3_kaiser_average_value)

best_Y3_kaiser_average_range =
    best_Y3_kaiser_average_value*np.ones
    ((Y3_kaiser.size, 1))
```

```
# best_Y3_hamming_average_range = 0.5*
    np.ones((Y3_hamming.size, 1))

first_index, last_index = find_index(
    best_Y3_kaiser_average_range,
    Y3_kaiser)
kaiser_FWHM = fs_3_kaiser[last_index] -
     fs_3_kaiser[first_index]
print("kaiser FWHM = %d" %int(
    kaiser_FWHM))

figure(5)
subplot(2,2,1)
plt.plot(x3, I3)
plt.title("Original waveform($5*2^{11}$
     zeros padding)")

subplot(2,2,2)
plt.plot(fs_3, Y3)
plt.plot(fs_3,best_Y3_average_range)
plt.title("Spectrogram($5*2^{11}$ zeros
     padding) FWHM = 7400 $m^{-1}$")
plt.xlim(1.50*(10**6), 1.65*(10**6))

plt.suptitle("632.8nm - Superposition
    of waves(Kaiser window)", fontsize =
     20)



subplot(2,2,3)
plt.plot(x3_kaiser, I3_kaiser)
plt.title("Original waveform(Kaiser
    window)")

subplot(2,2,4)
plt.plot(fs_3_kaiser, Y3_kaiser)
plt.plot(fs_3_kaiser,
    best_Y3_kaiser_average_range)
plt.title("Spectrogram(Kaiser window)
   FWHM = %d $m^{-1}$" %int(kaiser_FWHM
    ))
plt.xlim(1.50*(10**6), 1.65*(10**6))


####################################
# Window function begins here
# Case3.
#   - hanning
####################################

zero_numbers_3= 5*p3/2
x3_hanning = np.arange((-1)*(p4+
    zero_numbers_3)*i, (p4+
    zero_numbers_3)*i, i)
print("补零后x3的长度为 %d" %len(
    x3_hanning))
# 做傅里叶的点数, 取下一个2的次幂
n3_hanning = 2**(int(np.log2(len(
    x3_hanning)))+1)
print("n3 length = %d" %n3_hanning)
I3_hanning=(np.cos(2*np.pi*sigma0*x0));
L = len(I3_hanning)

# 定义kaiser窗函数
hanning_window = np.hanning(len(
    I3_hanning))

# 对 x3加窗
I3_hanning = I3_hanning *
    hanning_window

I3_hanning = np.pad(I3_hanning,(0,int(
    zero_numbers_3)),'constant',
    constant_values = (0,0))
I3_hanning = np.pad(I3_hanning,(int(
    zero_numbers_3),0),'constant',
    constant_values = (0,0))
print("补零后I3的长度为 %d" %len(
    I3_hanning))
```

```
Y3_hanning = 2*abs(fft(I3_hanning,
    n3_hanning))/len(x0)
Y3_hanning = Y3_hanning[:int(n3/2)]

# 设置频谱图的横坐标
fs_3_hanning = 1/i*np.arange(n3_hanning
    /2)/n3_hanning

Y3_hanning_average = (Y3_hanning.min()+
    Y3_hanning.max())/2
print("Y3_hanning min = %24e" %
    Y3_hanning.min())
print("Y3_hanning max = %24e" %
    Y3_hanning.max())
print("Y3_hanning_average = %24e" %
    Y3_hanning_average)

best_Y3_hanning_average_value =
    find_best_I_average_value(Y3_hanning
    , Y3_hanning_average)
print("best_Y3_hanning_average_value =
    %24e" %best_Y3_hanning_average_value
    )

best_Y3_hanning_average_range =
    best_Y3_hanning_average_value*np.
    ones((Y3_hanning.size, 1))
# best_Y3_hamming_average_range = 0.5*
    np.ones((Y3_hamming.size, 1))

first_index, last_index = find_index(
    best_Y3_hanning_average_range,
    Y3_hanning)
hanning_FWHM = fs_3_hanning[last_index]
     - fs_3_hanning[first_index]
print("hanning FWHM = %d" %int(
    hanning_FWHM))

figure(6)
subplot(2,2,1)
plt.plot(x3, I3)
plt.title("Original waveform($5*2^{11}$
     zeros padding)")

subplot(2,2,2)
plt.plot(fs_3, Y3)
plt.plot(fs_3,best_Y3_average_range)
plt.title("Spectrogram($5*2^{11}$ zeros
     padding) FWHM = 7400 $m^{-1}$")
plt.xlim(1.50*(10**6), 1.65*(10**6))

plt.suptitle("632.8nm - Superposition
    of waves(Hanning window)", fontsize
    = 20)



subplot(2,2,3)
plt.plot(x3_hanning, I3_hanning)
plt.title("Original waveform(Hanning
    window)")

subplot(2,2,4)
plt.plot(fs_3_hanning, Y3_hanning)
plt.plot(fs_3_hanning,
    best_Y3_hanning_average_range)
plt.title("Spectrogram(Hanning window)
   FWHM = %d $m^{-1}$" %int(
    hanning_FWHM))
plt.xlim(1.50*(10**6), 1.65*(10**6))



#
    ##########################################

# Window function begins here
# Case4.
#   - blackman
#
    ##########################################

zero_numbers_3= 5*p3/2
```

```
x3_blackman = np.arange((-1)*(p4+
    zero_numbers_3)*i, (p4+
    zero_numbers_3)*i, i)
print("补零后x3的长度为 %d" %len(x3))
# 做傅里叶的点数，取下一个2的次幂
n3_blackman = 2**(int(np.log2(len(
    x3_blackman)))+1)
I3_blackman=(np.cos(2*np.pi*sigma0*x0))
    ;
L = len(I3_blackman)

# 定义blackman窗函数
blackman_window = np.blackman(len(
    I3_blackman))
# 对 x3加窗
I3_blackman = I3_blackman *
    blackman_window

I3_blackman = np.pad(I3_blackman,(0,int
    (zero_numbers_3)),'constant',
    constant_values = (0,0))
I3_blackman = np.pad(I3_blackman,(int(
    zero_numbers_3),0),'constant',
    constant_values = (0,0))
print("补零后I3的长度为 %d" %len(I3))



Y3_blackman = 2*abs(fft(I3_blackman,
    n3_blackman))/len(x0)
Y3_blackman = Y3_blackman[:int(n3/2)]

# 设置频谱图的横坐标
fs_3_blackman = 1/i*np.arange(
    n3_blackman/2)/n3_blackman

Y3_blackman_average = (Y3_blackman.min
    ()+Y3_blackman.max())/2
print("Y3_blackman min = %24e" %
    Y3_blackman.min())
print("Y3_blackman max = %24e" %
    Y3_blackman.max())
print("Y3_blackman_average = %24e" %
    Y3_blackman_average)

best_Y3_blackman_average_value =
    find_best_I_average_value(
    Y3_blackman, Y3_blackman_average)
print("best_Y3_blackman_average_value =
     %24e" %
    best_Y3_blackman_average_value)

best_Y3_blackman_average_range =
    best_Y3_blackman_average_value*np.
    ones((Y3_blackman.size, 1))
# best_Y3_blackman_average_range = 0.5*
    np.ones((Y3_blackman.size, 1))

first_index, last_index = find_index(
    best_Y3_blackman_average_range,
    Y3_blackman)
blackman_FWHM = fs_3_blackman[
    last_index] - fs_3_blackman[
    first_index]
print("blackman FWHM = %d" %int(
    blackman_FWHM))

figure(7)
subplot(2,2,1)
plt.plot(x3, I3)
plt.title("Original waveform($5*2^{11}$
     zeros padding)")

subplot(2,2,2)
plt.plot(fs_3, Y3)
plt.plot(fs_3,best_Y3_average_range)
plt.title("Spectrogram($5*2^{11}$ zeros
     padding) FWHM = 7400 $m^{-1}$")
plt.xlim(1.50*(10**6), 1.65*(10**6))
```

```
plt.suptitle("632.8nm - Superposition
    of waves(Blackman window)", fontsize
    = 20)


subplot(2,2,3)
plt.plot(x3_blackman, I3_blackman)
plt.title("Original waveform(Blackman
    window)")

subplot(2,2,4)
plt.plot(fs_3_blackman, Y3_blackman)
plt.plot(fs_3_blackman,
    best_Y3_blackman_average_range)
plt.title("Spectrogram(Blackman window)
    FWHM = %d $m^{-1}$" %int(
    blackman_FWHM))
plt.xlim(1.50*(10**6), 1.65*(10**6))




#########################################
# Window function begins here
# Case5.
#   - bartlett
#########################################

zero_numbers_3= 5*p3/2
x3_bartlett = np.arange((-1)*(p4+
    zero_numbers_3)*i, (p4+
    zero_numbers_3)*i, i)
print("补零后x3的长度为 %d" %len(
    x3_bartlett))
# 做傅里叶的点数, 取下一个2的次幂
n3_bartlett = 2**(int(np.log2(len(
    x3_bartlett)))+1)
print("n3 length = %d" %n3_bartlett)
I3_bartlett=(np.cos(2*np.pi*sigma0*x0))
    ;
L = len(I3_bartlett)


# 定义kaiser窗函数
bartlett_window = np.bartlett(len(
    I3_bartlett))

# 对 x3加窗
I3_bartlett = I3_bartlett *
    bartlett_window

I3_bartlett = np.pad(I3_bartlett,(0,int
    (zero_numbers_3)),'constant',
    constant_values = (0,0))
I3_bartlett = np.pad(I3_bartlett,(int(
    zero_numbers_3),0),'constant',
    constant_values = (0,0))
print("补零后I3的长度为 %d" %len(
    I3_bartlett))



Y3_bartlett = 2*abs(fft(I3_bartlett,
    n3_bartlett))/len(x0)
Y3_bartlett = Y3_bartlett[:int(n3/2)]

# 设置频谱图的横坐标
fs_3_bartlett = 1/i*np.arange(
    n3_bartlett/2)/n3_bartlett

Y3_bartlett_average = (Y3_bartlett.min
    ()+Y3_bartlett.max())/2
print("Y3_bartlett min = %24e" %
    Y3_bartlett.min())
print("Y3_bartlett max = %24e" %
    Y3_bartlett.max())
print("Y3_bartlett_average = %24e" %
    Y3_bartlett_average)

best_Y3_bartlett_average_value =
    find_best_I_average_value(
    Y3_bartlett, Y3_bartlett_average)
```

```
print("best_Y3_bartlett_average_value =
    %24e" %
    best_Y3_bartlett_average_value)

best_Y3_bartlett_average_range =
    best_Y3_bartlett_average_value*np.
    ones((Y3_bartlett.size, 1))
# best_Y3_blackman_average_range = 0.5*
    np.ones((Y3_blackman.size, 1))

first_index, last_index = find_index(
    best_Y3_bartlett_average_range,
    Y3_bartlett)
bartlett_FWHM = fs_3_bartlett[
    last_index] - fs_3_bartlett[
    first_index]
print("bartlett FWHM = %d" %int(
    bartlett_FWHM))

figure(8)
subplot(2,2,1)
plt.plot(x3, I3)
plt.title("Original waveform($5*2^{11}$
    zeros padding)")

subplot(2,2,2)
plt.plot(fs_3, Y3)
plt.plot(fs_3,best_Y3_average_range)
plt.title("Spectrogram($5*2^{11}$ zeros
    padding) FWHM = 7400 $m^{-1}$")
plt.xlim(1.50*(10**6), 1.65*(10**6))

plt.suptitle("632.8nm - Superposition
    of waves(Bartlett window)", fontsize
    = 20)


subplot(2,2,3)
plt.plot(x3_bartlett, I3_bartlett)
plt.title("Original waveform(Bartlett
    window)")

subplot(2,2,4)
plt.plot(fs_3_bartlett, Y3_bartlett)
plt.plot(fs_3_bartlett,
    best_Y3_bartlett_average_range)
plt.title("Spectrogram(Bartlett window)
    FWHM = %d $m^{-1}$" %int(
    bartlett_FWHM))
plt.xlim(1.50*(10**6), 1.65*(10**6))


#######################################
# 将所有加窗后波形汇总在一个图里面
#######################################
figure(9)
l1, = plt.plot(fs_3, Y3)
l2, = plt.plot(fs_3_hanning, Y3_hanning
    )
l3, = plt.plot(fs_3_kaiser, Y3_kaiser)
l4, = plt.plot(fs_3_hamming, Y3_hamming
    )
l5, = plt.plot(fs_3_bartlett,
    Y3_bartlett)
l6, = plt.plot(fs_3_blackman,
    Y3_blackman)
plt.xlim(1.50*(10**6), 1.65*(10**6))


plt.legend(handles=[l1,l2,l3,l4,l5,l6],
    labels=['Rectangular window','
    Hanning window', 'Kaiser window','
    Hamming window','Bartlett window','
    Blackman window'], loc='upper right'
    )

plt.suptitle("632.8nm - Superposition
    of waves", fontsize = 20)

plt.subplots_adjust(left=0.125,
                    bottom=0.1,
                    right=0.9,
```

```
                    top=0.9,
                    wspace=0.2,
                    hspace=0.35)
plt.show()
```