

**ECE 278C Assignment 2**

Peicheng Wu NetID：X311088

2024.02.03

Catalog

[1. Platform: VScode 3](#_Toc157889953)

[2. Question 3](#_Toc157889954)

[3. Solutions 4](#_Toc157889955)

[4. Results 5](#_Toc157889956)

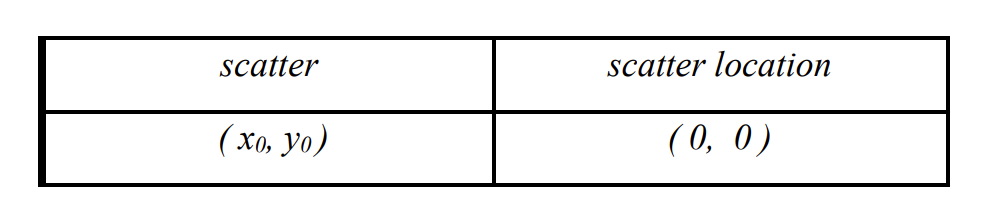
[5. Appendix 6](#_Toc157889957)

# Platform: VScode

Visual Studio Code, often referred to as VSCode, is a free and open-source code editor developed by Microsoft. It's renowned for its lightweight design, speed, and robust capabilities. Available for Windows, macOS, and Linux, VSCode offers built-in Git integration, an integrated terminal, and a debugger. Its power lies in its extensibility, with a vast marketplace of extensions that add support for various programming languages, debuggers, and tools. The editor's appearance and behavior are highly customizable, and its Intellisense feature provides smart code completions. Thanks to its open-source nature, it has a vibrant community that continually contributes to its development and enhancement.

# Question

Consider a single centered point source,

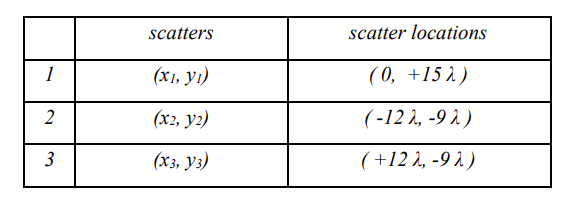


The horizontal receiver aperture is organized in the form of a centered linear receiver array with a span of 60λ (from x= -30λ to x = +30λ). This horizontal linear receiver array is placed at

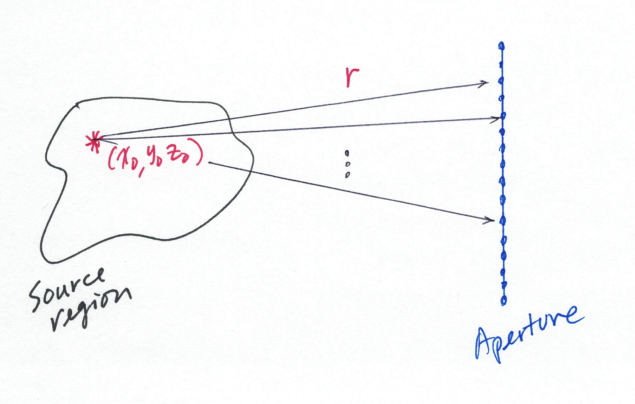
With quarter-wavelength spacing (λ/4) spacing, there are 241 complex wavefield data samples in total over the 60λ-long linear aperture.

* Perform image reconstruction of the (60λ x 60λ) 2D source region. The source region is a square area centered at (0, 0) and bounded by x = ± 30λ and y = ± 30λ. For consistency, use quarter-wavelength spacing as the sample spacing in both directions.
* Obtain and show the 512 x 512 FFT spectrum of the complex image. (Apply the same procedure you used for Assignment 1.)
* Plot the magnitude distribution of your reconstructed image.

Then try the three source scatters:



# Solutions



In the field we have equation:

And now we just assume source as the impulse function:

Thus, for each receiver, we can write the received signal

According the backward propagation:

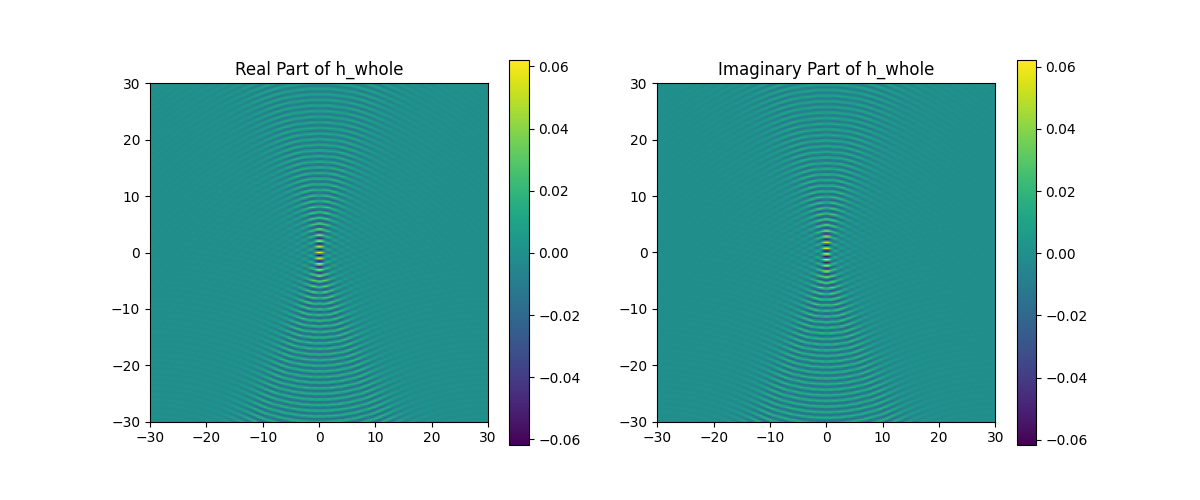
Where r means the distance from source scatter to the receiver and r’ means the distance from source region point to the receiver.

So we can just write the equation of the reconstructed image:

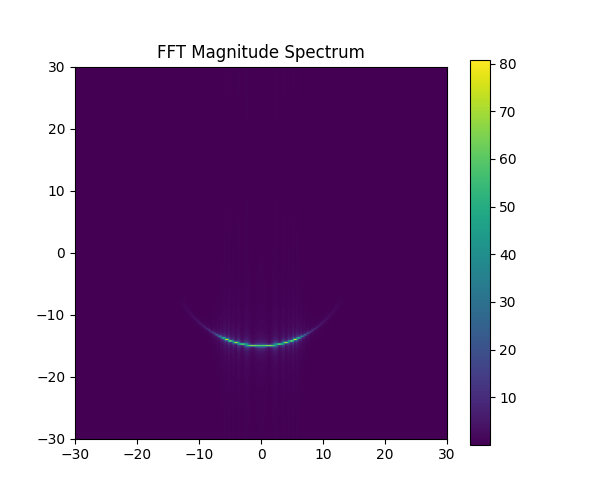
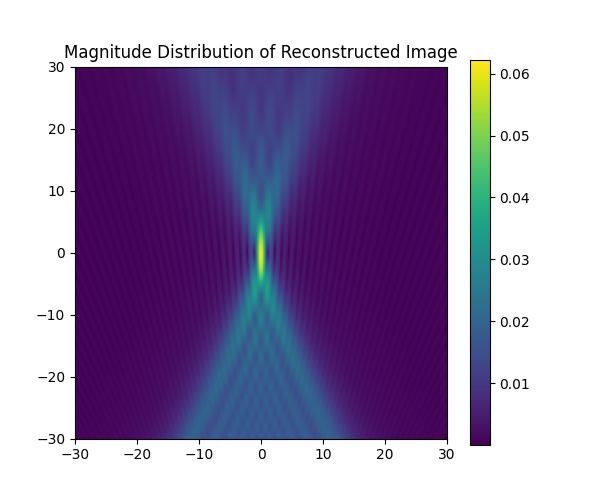
Then we can use this equation calculate all the points in the source region.

# Results

For the one target:

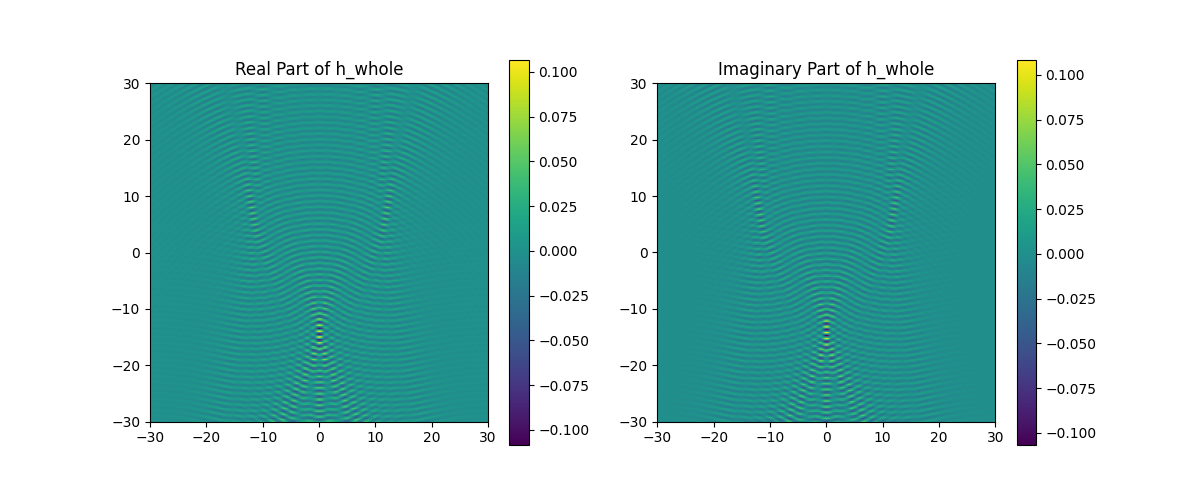


**Fig.1 Reconstructed image**

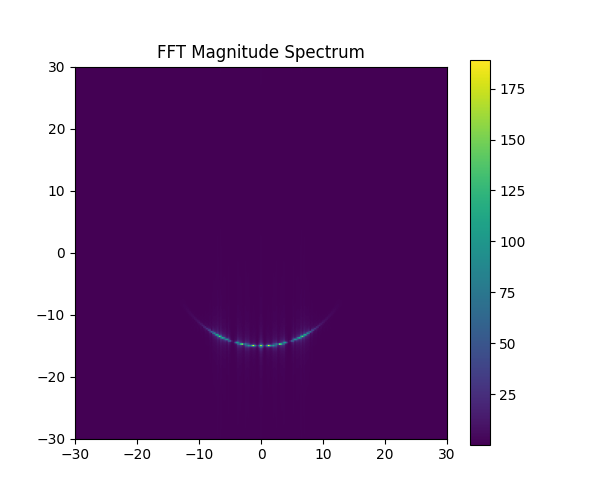
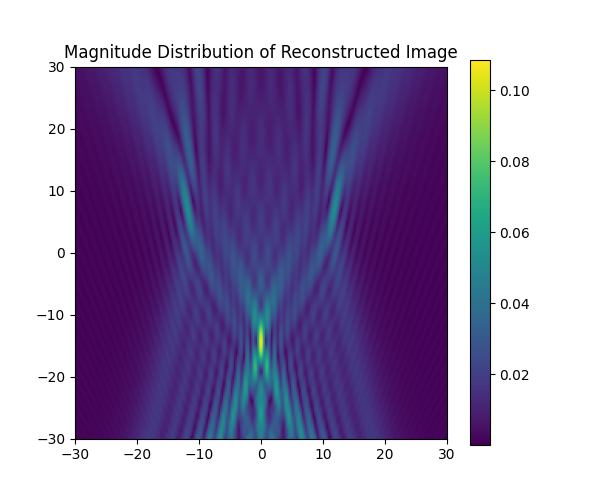


**Fig.2 Magnitude distribution and FFT**

For the three targets:



**Fig.3 Reconstructed image**



**Fig.4 Magnitude distribution and FFT**

# Appendix

import numpy as np

import matplotlib.pyplot as plt

from scipy.fft import fft2, fftshift

import argparse

def generate\_receivers\_locations(lambda\_0, y\_offset, span):

    delta\_x = lambda\_0 / 4

    num\_points = int(span / delta\_x) + 1

    x\_positions = np.linspace(-span / 2, span / 2, num\_points)

    scatter\_locations = [(x, y\_offset) for x in x\_positions]

    # print(scatter\_locations)

    return scatter\_locations

def generate\_source\_region(lambda\_0, X\_length, Y\_length):

    delta\_x = lambda\_0 / 4

    num\_points\_x = int(X\_length / delta\_x) + 1

    num\_points\_y = int(Y\_length / delta\_x) + 1

    x\_positions = np.linspace(-X\_length / 2, X\_length / 2, num\_points\_x)

    y\_positions = np.linspace(-Y\_length / 2, Y\_length / 2, num\_points\_y)

    source\_locations = [(x, y) for y in y\_positions for x in x\_positions]

    # print(source\_locations)

    return source\_locations

def calculate(source\_scatter):

    lambda\_0 = 1

    X\_length = 60 \* lambda\_0

    Y\_length = 60 \* lambda\_0

    y\_offset = 60 \* lambda\_0

    span = 60 \* lambda\_0

    source\_locations = generate\_source\_region(lambda\_0, X\_length, Y\_length)

    receivers\_locations = generate\_receivers\_locations(lambda\_0, y\_offset, span)

    num\_points\_x = int(X\_length / (lambda\_0 / 4)) + 1

    num\_points\_y = int(Y\_length / (lambda\_0 / 4)) + 1

    h\_whole = np.zeros((num\_points\_y, num\_points\_x), dtype=complex)

    temp = np.zeros((num\_points\_y, num\_points\_x), dtype=complex)

    source\_scatter = [(0, 0)]

    for x\_cc, y\_cc in source\_scatter:

        for idx, (x\_s, y\_s) in enumerate(source\_locations):

            y\_idx = idx // num\_points\_x

            x\_idx = idx % num\_points\_x

            h = 0

            for x\_g, y\_g in receivers\_locations:

                r\_g = np.sqrt((x\_g - x\_cc)\*\*2 + (y\_g - y\_cc)\*\*2)

                r\_s = np.sqrt((x\_s - x\_g)\*\*2 + (y\_s - y\_g)\*\*2)

                h\_n = (1/(lambda\_0\*\*2 \* r\_g \* r\_s)) \* np.exp(1j \* 2 \* np.pi \* (r\_g - r\_s) / lambda\_0)

                h += h\_n

            h\_whole[y\_idx, x\_idx] = h

        temp += h\_whole

    return temp

if \_\_name\_\_ == "\_\_main\_\_":

    parser = argparse.ArgumentParser(description='Different question in the assignment')

    parser.add\_argument('--Q1', action='store\_true', help='Process one target')

    parser.add\_argument('--Q2', action='store\_true', help='Process three target')

    args = parser.parse\_args()

    if args.Q1:

        source\_scatter = [(0, 0)]

    elif args.Q2:

        source\_scatter = [(0, 15), (-12, -9), (12, -9)]

    h\_whole = calculate(source\_scatter)

    # 绘制 h\_whole 的实部和虚部

    plt.figure(figsize=(12, 5))

    plt.subplot(1, 2, 1)

    plt.imshow(h\_whole.real, extent=(-30, 30, -30, 30))

    plt.title('Real Part of h\_whole')

    plt.colorbar()

    plt.subplot(1, 2, 2)

    plt.imshow(h\_whole.imag, extent=(-30, 30, -30, 30))

    plt.title('Imaginary Part of h\_whole')

    plt.colorbar()

    plt.show()

    # FFT 变换

    fft\_image = fftshift(fft2(h\_whole))

    # 绘制 FFT 幅度分布图

    plt.figure(figsize=(6, 5))

    plt.imshow(np.abs(fft\_image), extent=(-30, 30, -30, 30))

    plt.title('FFT Magnitude Spectrum')

    plt.colorbar()

    plt.show()

    # 绘制重建图像的幅度分布

    plt.figure(figsize=(6, 5))

    plt.imshow(np.abs(h\_whole), extent=(-30, 30, -30, 30))

    plt.title('Magnitude Distribution of Reconstructed Image')

    plt.colorbar()

    plt.show()

You can also find the code on the github: [UCSB-ECE-278C/assignment2 at main · percyance/UCSB-ECE-278C (github.com)](https://github.com/percyance/UCSB-ECE-278C/tree/main/assignment2)

When you want to run: please type python assignment2.py --Q1 to Q2