BME2104 HW#1 **Imaging Foundation**

Zhihao Zhang, 2024291074

zhangzhh12024@shanghaitech.edu.cn

Mar. 27, 2025

Problem 1.

A cycle of a square wave can be expressed as following equation

$$f(x) = \begin{cases} 1, & 0 \leq x < 1/2 \\ 0, & 1/2 \leq x < 1 \end{cases}$$

Using Fourier Series, please do analytical derivation to prove that this square wave can be represented as a linear combination of sine waves of different frequencies.

Then, using a programming environment of your choice (MATLAB/Python), please build a numerical model to demonstrate that the above square wave is indeed a linear combination of sine waves. Please use plots to help explain.

Solution 1

For a periodic function with period 1, the Fourier series is given by:

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} [a_n \cos(2\pi n x) + b_n \sin(2\pi n x)]$$

Where the coefficients are:

$$a_0 = 2\int_0^1 f(x)dx \tag{1}$$
$$a_n = 2\int_0^1 f(x)\cos(2\pi nx)dx \tag{2}$$
$$b_n = 2\int_0^1 f(x)\sin(2\pi nx)dx \tag{3}$$

For the given square wave:

$$a_0 = 2\int_0^1 f(x)dx = 2\int_0^{1/2} 1\,dx + 2\int_{1/2}^1 0\,dx = 1 \tag{4}$$
$$a_n = 2\int_0^{1/2} \cos(2\pi nx)dx = \frac{\sin(\pi n)}{\pi n} = 0 \tag{5}$$
$$b_n = 2\int_0^{1/2} \sin(2\pi nx)dx = \frac{1-\cos(\pi n)}{\pi n} = \begin{cases} \frac{2}{\pi n}, & \text{for odd } n \\ 0, & \text{for even } n \end{cases} \tag{6}$$

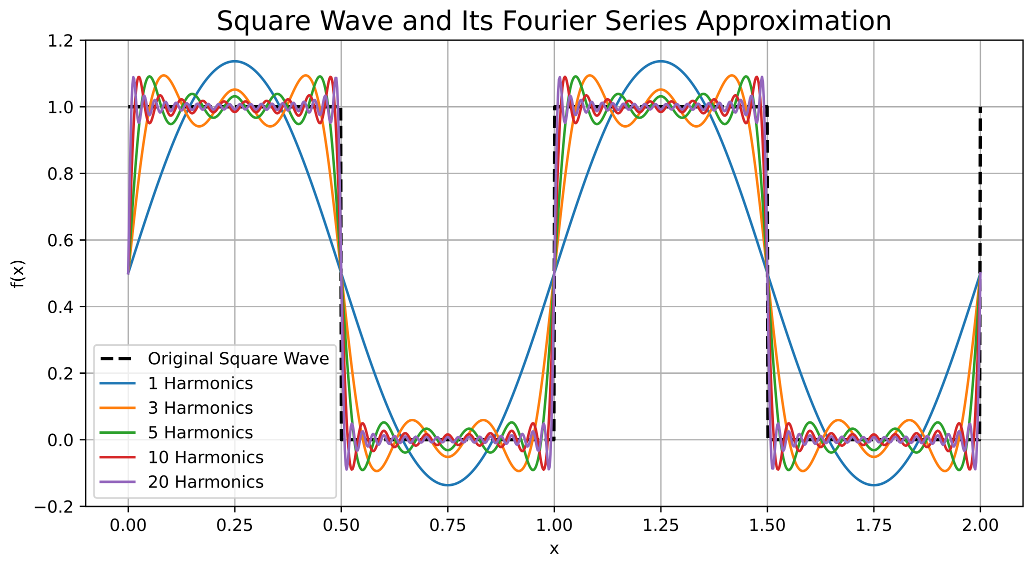
Therefore, the Fourier series representation is:

$$f(x) = \frac{1}{2} + \frac{2}{\pi}\sum_{k=0}^{\infty} \frac{1}{2k+1}\sin(2\pi(2k+1)x)$$

Python implementation to visualize the Fourier approximation:

```python
def fourier_square(x, harmonics):
# Start with the constant term a0 = 1/2
f_approx = 0.5 * np.ones_like(x)
for k in range(harmonics):
    n = 2 * k + 1 # odd only
    f_approx += (2 / (np.pi * n)) * np.sin(2 * np.pi
        * n * x)
return f_approx
```

From both the analytical derivation and numerical demonstration, we've shown that the square wave can be represented as a linear combination of sine waves of different frequencies. We observe that:



- Only odd-numbered harmonics contribute to the series
- The amplitude of each harmonic decreases as $1/n$
- More terms lead to better approximation, especially at discontinuities
- The Gibbs phenomenon causes oscillations near the discontinuities

Problem 2.

A micro-CT imaging system' s spatial resolution was characterized with a thin tungsten wire phantom. An axial micro-CT slice image of the phantom is shown here (original image file is uploaded to Blackboard, " MTF-slice" . Note the pixel size is 7.6um).

(1). Please plot the Line Intensity Profile (LIP) of the wire in the center of the image. Please fit the LIP with a Gaussian function. (2). Assume that the diameter of the wire is very small, therefore the LIP can be assumed to be the LSF of the micro-CT. Using the LSF, please calculate the MTF of the micro-CT. Please plot the MTF, and fit your MTF with a Gaussian function. Is your Gaussian function in (2) close to the FT (Fourier Transform) of the Gaussian function in (1)? Find your spatial resolution at 10% MTF? Explain your result.

Solution 2

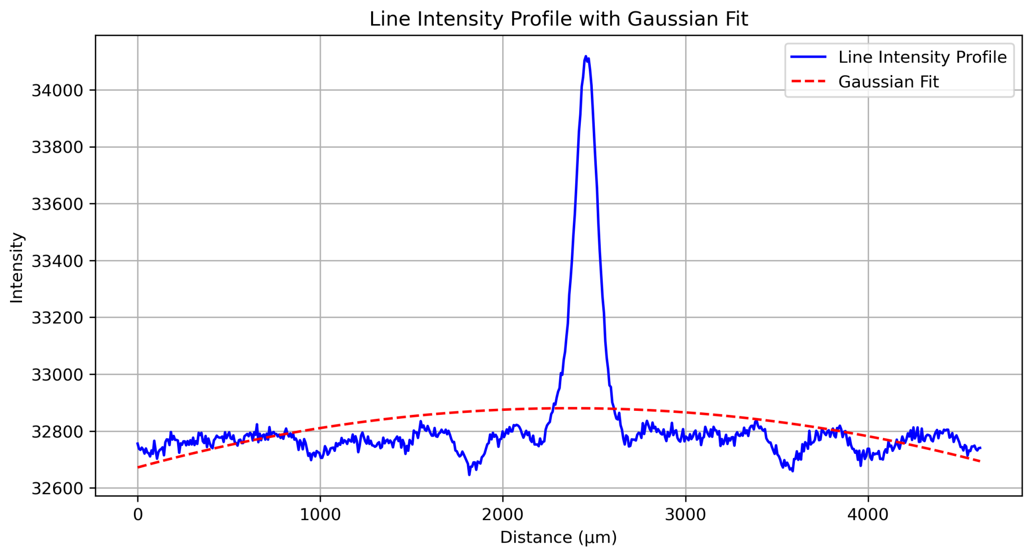
Line Intensity Profile (LIP) is defined as the intensity of the image along a line, which can be represented as a function of distance from the center of the wire.

The key Python code and the result is as follows:

```
# Get the center row of the image
center_row = img[img.shape[0] // 2, :]

# Define Gaussian function for fitting
def gaussian(x, amplitude, mean, sigma):
    return amplitude * np.exp(-((x - mean) ** 2) / (2
        * sigma**2))

# Fit Gaussian to LIP
popt, _ = curve_fit(gaussian, x, center_row)
fitted_gaussian = gaussian(x, *popt)
```



Line Spread Function (LSF) is the response of the imaging system to a point source. The LSF can be derived from the Line Intensity Profile (LIP) by taking the derivative of the Gaussian function fitted to the LIP.

$$LSF(y) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y-y_0)^2}{2\sigma^2}\right)$$

Problem 3.

Start with any noisy digital image (it can be downloaded from the web, synthesized by yourself, or even an image in your smartphone), first convert it into 8-bit grey scale if it is not already so, then re-size it to 512 x 512 using digital down-sampling or up-sampling (combined with cropping/padding if necessary), you will end up with a noisy grey-scale image of 512 x 512. Then, do image denoising using the following two approaches:

(1). Filtering in frequency space: First find the FT version of the image, then multiply the FT version with a low-pass filter, which will give you a filtered FT version. Finally do an inverse FT to yield the denoised image. Please try with low-pass filters of different filtration levels, and show your results.

(2). Filtering in image space: design a ‘smoothing/denoising’ filter kernel, and convolute it with the noisy grey-scale image to receive your denoised image.

(3). For the above two denoising approaches, calculate their respective MSE, PSNR, and SSIM relative to the original image before denoising.

Solution 3