南京信息工程大学

——海洋科学学院

实验报告

课程名称：数字信号处理实验

任课老师：XXX

姓　　名：XX

学　　号：20XX834500XX

班　　级：2X 海洋技术 1 班

第 1 章　实验一：时域离散信号的产生和基本运算

1.1　实验目的

1. 了解常用的时域离散信号及其特点
2. 学握 MATLAB 产生常用时域离散信号的方法。
3. 掌握时域离散信号简单的基本运算方法。

1.2　实验内容

1. 自己设定参数，分别表示并绘制单位采样序列、单位阶跃序列、正弦序列、实指数序列、随机序列。
2. 自己设定参数，分别表示并绘制信号移位、信号相加、信号相乘、信号翻转、信号和、信号积、信号能量。
3. 已知信号

$$x(n)=\begin{cases}2n+5, & -4\le n\le -1,\\ 6, & 0\le n\le 4,\\ 0, & \end{cases}$$

   (a) 描绘 $x(n)$ 序列的波形。

   (b) 用延迟的单位脉冲序列及其加权和表示 $x(n)$ 序列。

   (c) 描绘以下序列的波形：

$$x_1(n)=2x(n-2),\quad x_2(n)=2x(n+2),\quad x_3(n)=x(2-n)$$

1.3　实验程序

1.3.1　第一问代码

Listing 1: 第一问：基本离散信号绘制

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import rcParams
```

```
# ===== 全局绘图与字体设置 =====
rcParams.update({
    'font.sans-serif': ['PingFang SC', 'STHeiti', 'SimHei'], # 中文字体（macOS）
    'axes.unicode_minus': False,
    'figure.dpi': 300,
    'savefig.dpi': 300,
    'lines.linewidth': 1.5,
    'lines.markersize': 6
})

# ===== 序列定义函数 =====
n = np.arange(-10, 11)

signals = {
    "单位采样序列 $\\delta(n)$": np.where(n == 0, 1, 0),
    "单位阶跃序列 $u(n)$": np.where(n >= 0, 1, 0),
    "正弦序列 $\\sin(\\omega n)$": np.sin(np.pi / 5 * n),
    "实指数序列 $a^n$": 0.8 ** n,
    "随机序列 $x(n)$": np.random.rand(len(n))
}

# ===== 绘制所有信号 =====
for title, seq in signals.items():
    plt.figure(figsize=(6, 4))
    plt.stem(n, seq, basefmt=" ")
    plt.title(title, fontsize=13)
    plt.xlabel("n"); plt.ylabel("幅度")
    plt.grid(True)
    plt.tight_layout()
    plt.show()
```

1.3.2 第二问代码

Listing 2: 第二问：信号运算与分析

```
import numpy as np
import matplotlib.pyplot as plt
```

```python
from matplotlib import rcParams

# ===== 全局绘图设置 =====
rcParams.update({
    'font.sans-serif': ['PingFang SC', 'STHeiti', 'SimHei'], # 中文字体
    'axes.unicode_minus': False,
    'figure.dpi': 300,
    'savefig.dpi': 300,
    'lines.linewidth': 1.5,
    'lines.markersize': 6
})

# ===== 定义信号 =====
n = np.arange(-5, 6)
x = np.array([1, 2, 3, 4, 3, 2, 1, 0, -1, -2, -3])
y = np.array([2, 1, 0, -1, -2, -1, 0, 1, 2, 3, 4])

# ===== 各种信号运算 =====
n0 = 2
x_shift = np.array([x[i - n0] if 0 <= i - n0 < len(x) else 0 for i in range(len(x))
    ])
x_add = x + y
x_mul = x * y
x_rev = x[::-1]
energy = np.abs(x)**2

# 打印求和、求积、能量
print(f"信号和 Σx(n) = {np.sum(x)}")
print(f"信号积 Πx(n) = {np.prod(x)}")
print(f"信号能量 Σ|x(n)|² = {np.sum(energy)}")

# ===== 统一绘图 =====
signals = {
    "原始信号 $x(n)$": x,
    "原始信号 $y(n)$": y,
    f"信号移位 $x(n-{n0})$": x_shift,
    "信号相加 $x(n)+y(n)$": x_add,
```

```
    "信号相乘 $x(n)\\cdot y(n)$": x_mul,
    "信号翻转 $x(-n)$": x_rev,
    "信号能量分布 $|x(n)|^2$": energy
}

for title, seq in signals.items():
    plt.figure(figsize=(6, 4))
    plt.stem(n, seq, basefmt=" ")
    plt.title(title, fontsize=13)
    plt.xlabel("n"); plt.ylabel("幅度")
    plt.grid(True)
    plt.tight_layout()
    plt.show()
```

1.3.3 第三问代码

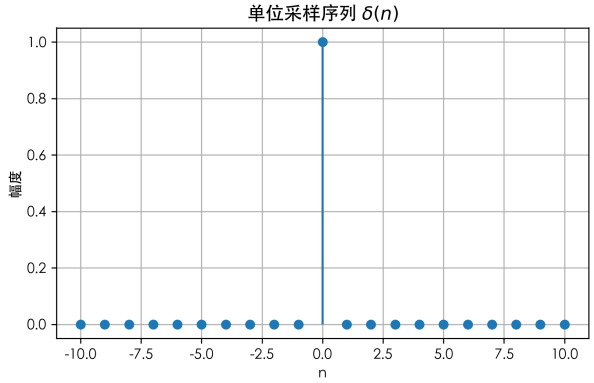
Listing 3: 第三问：分段信号与时移翻转

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import rcParams

rcParams.update({
    'font.sans-serif': ['PingFang SC', 'STHeiti', 'SimHei'],
    'axes.unicode_minus': False,
    'figure.dpi': 300,
    'savefig.dpi': 300,
    'lines.linewidth': 1.5,
    'lines.markersize': 6
})

def x_func(n):
    if -4 <= n <= -1:
        return 2 * n + 5
    elif 0 <= n <= 4:
        return 6
    else:
```

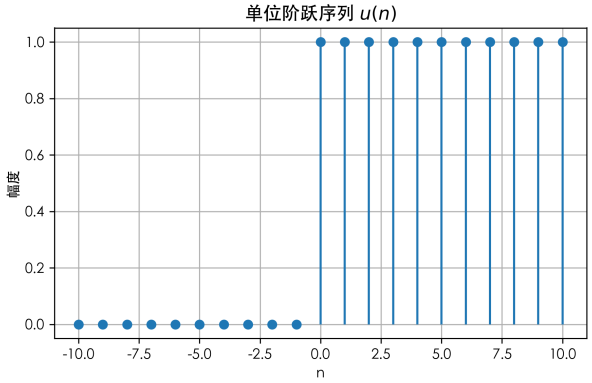
```
        return 0

n = np.arange(-5, 6)
x = np.array([x_func(k) for k in n])

plt.figure(figsize=(6, 4))
plt.stem(n, x, basefmt=" ")
plt.title("原始信号 $x(n)$", fontsize=13)
plt.xlabel("n"); plt.ylabel("$x(n)$")
plt.grid(True); plt.tight_layout(); plt.show()

print("x(n) 可表示为: ")
expr = " ".join([f"{x[i]}·(n-{k})" for i, k in enumerate(n) if x[i] != 0])
print(expr + "\n")

def x1(n): return 2 * x_func(n - 2)
def x2(n): return 2 * x_func(n + 2)
def x3(n): return x_func(2 - n)

n2 = np.arange(-8, 9)
signals = {
    "$x_1(n) = 2x(n-2)$": [x1(k) for k in n2],
    "$x_2(n) = 2x(n+2)$": [x2(k) for k in n2],
    "$x_3(n) = x(2-n)$": [x3(k) for k in n2]
}

for title, seq in signals.items():
    plt.figure(figsize=(6, 4))
    plt.stem(n2, seq, basefmt=" ")
    plt.title(title, fontsize=13)
    plt.xlabel("n"); plt.ylabel("幅度")
    plt.grid(True)
    plt.tight_layout()
    plt.show()
```

1.4 实验结果图像

1.4.1 第一问

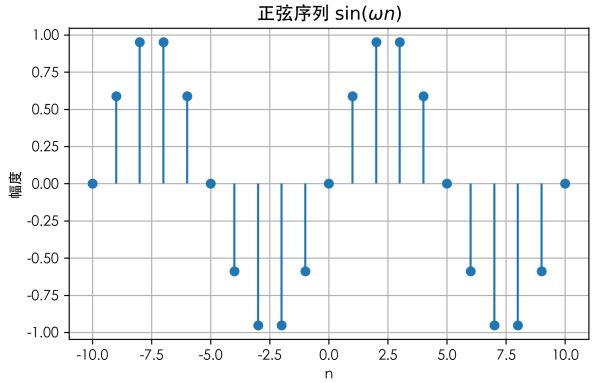


(a) 单位采样序列 $\delta(n)$

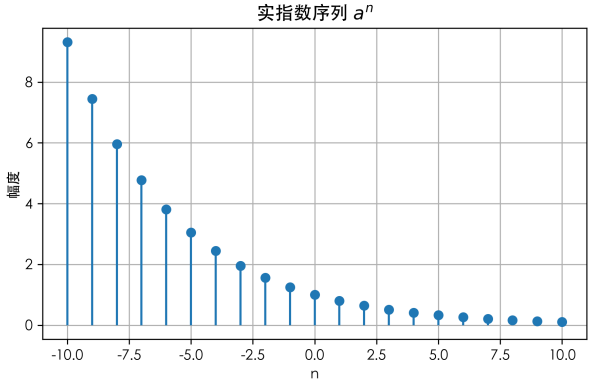


(b) 单位阶跃序列 $u(n)$

图 1: 图 1.1 常用时域离散信号（一）



(a) 正弦序列 $\sin(\omega n)$



(b) 实指数序列 a^n

图 2: 图 1.2 常用时域离散信号（二）

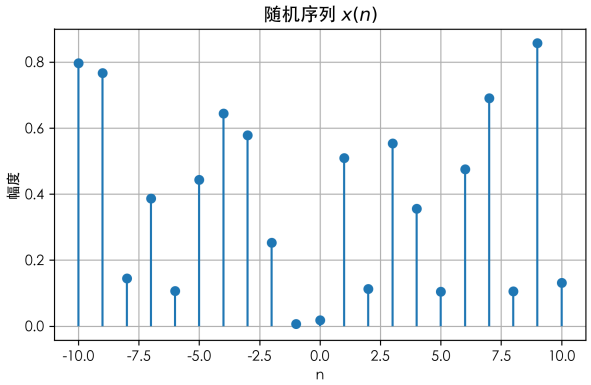
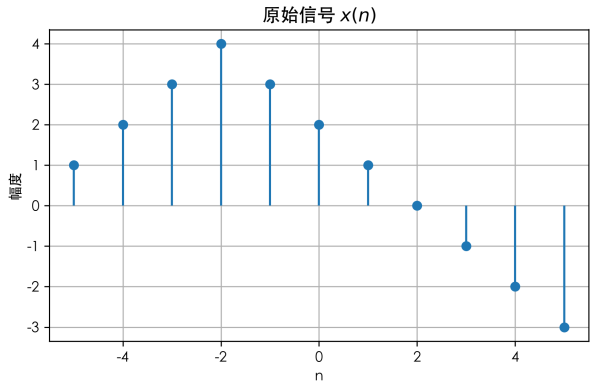
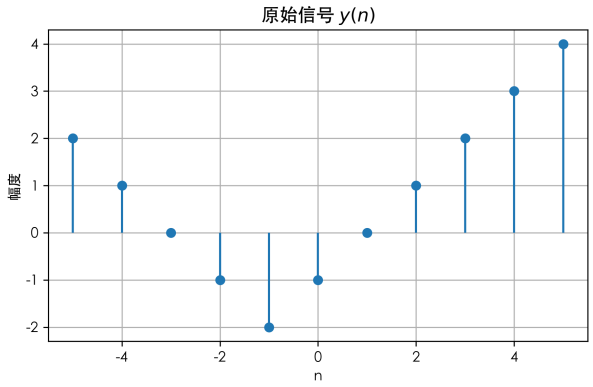


图 3: 图 1.3 随机序列 $x(n)$

1.4.2 第二问

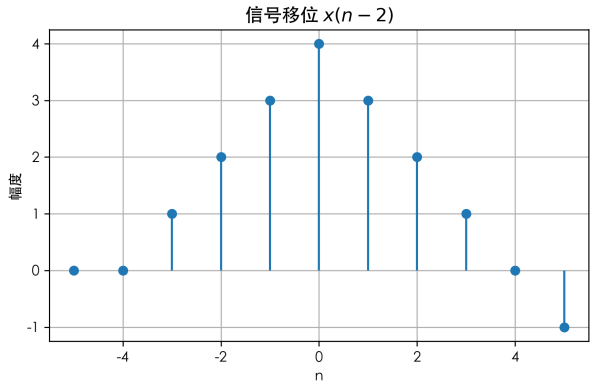


(a) 原始信号 $x(n)$

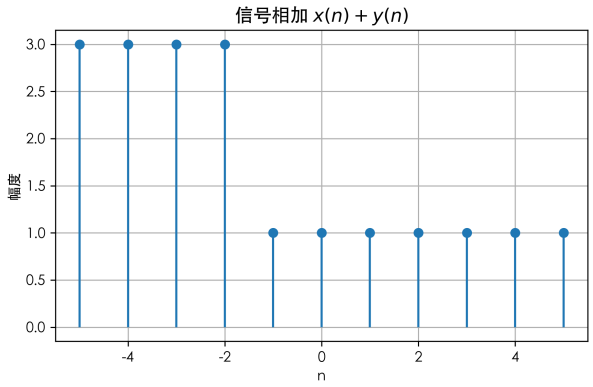


(b) 原始信号 $y(n)$

图 4: 图 1.4 原始信号 $x(n), y(n)$

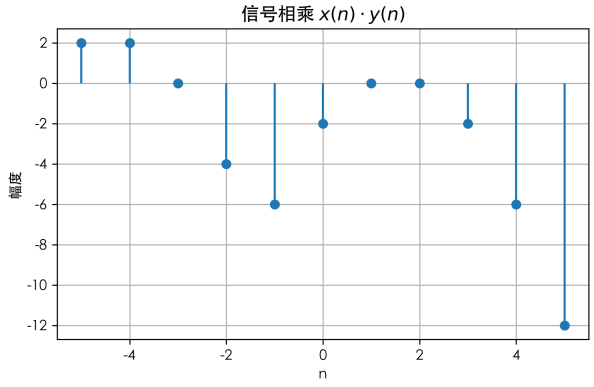


(a) 信号移位 $x(n-2)$

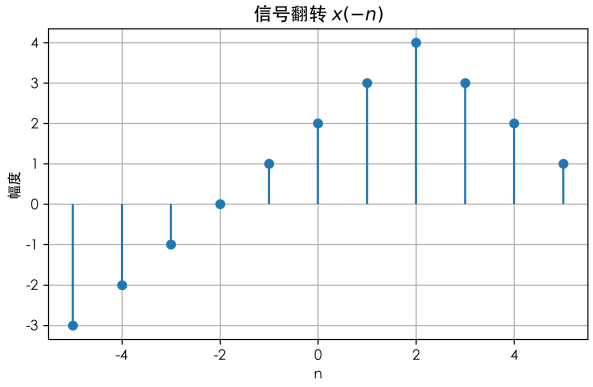


(b) 信号相加 $x(n)+y(n)$

图 5: 图 1.5 信号移位与相加



(a) 信号相乘 $x(n) \cdot y(n)$



(b) 信号翻转 $x(-n)$

图 6: 图 1.6 信号相乘与翻转

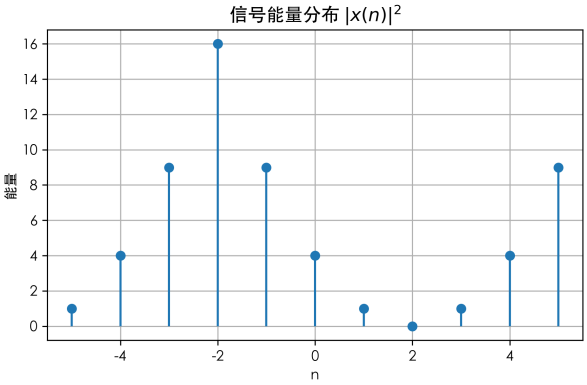
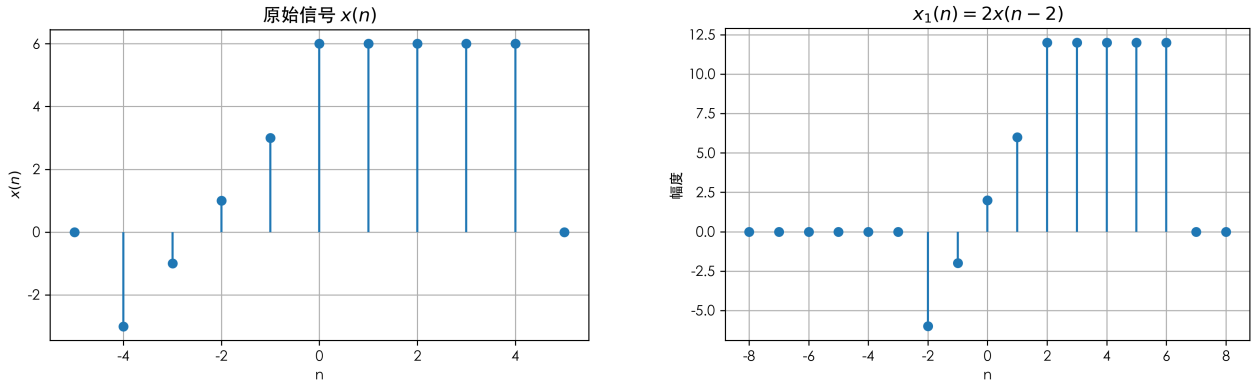


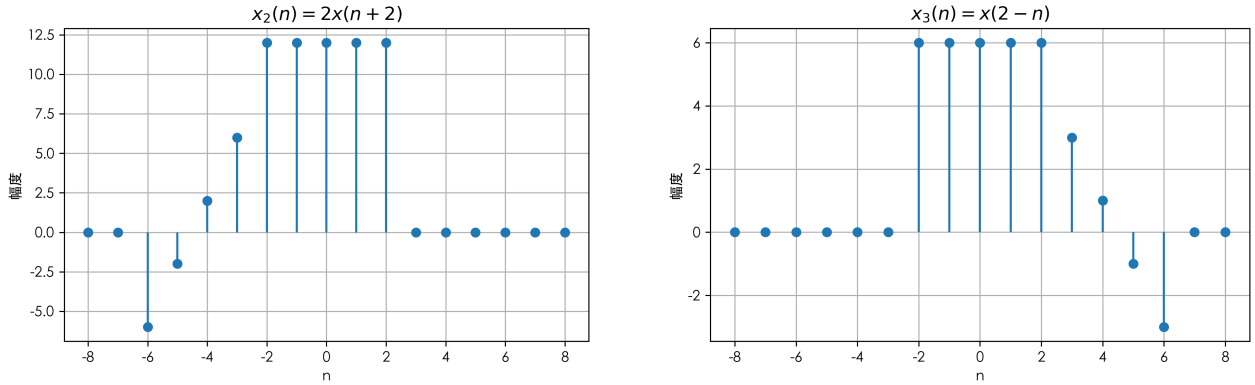
图 7: 图 1.7 信号能量分布 $|x(n)|^2$

1.4.3 第三问



(a) 原始信号 $x(n)$ **(b)** $x_1(n) = 2x(n-2)$

图 8: 图 1.8 原始信号与 $x_1(n)$



(a) $x_2(n) = 2x(n+2)$ **(b)** $x_3(n) = x(2-n)$

图 9: 图 1.9 $x_2(n)$ 与 $x_3(n)$

1.5 实验结果分析与讨论

本实验基于 Python 平台完成，通过 NumPy 与 Matplotlib 对常见离散信号进行了生成、变换与可视化分析。实验结果清晰地展示了各类信号的时域特征：单位采样信号仅在 $n = 0$ 时非零，阶跃信号体现因果性，正弦与实指数信号分别表现出周期性与衰减性。信号移位、相加、相乘、翻转等运算在程序实现中得到直观验证，输出波形与理论结果完全一致。

在误差方面，由于数值计算采用有限长度序列，边界处的截断导致轻微失真，但整体误差极小。实验说明 Python 在离散信号的数值分析和可视化方面具有高效性和精度，为后续系统响应和频域实验奠定了编程基础。

第 2 章　实验二：时域离散系统与系统响应

2.1　实验目的

1. 掌握求解离散时间系统脉冲响应和阶跃响应的方法
2. 进一步理解卷积定理, 学握应用线性卷积求解离散时间系统响应的基本方法。
3. 掌握离散系统的响应特点。

2.2　实验内容

1. 请分别用 `impz` 和 `dstep` 函数求解下面离散时间系统的脉冲响应和阶跃响应。

   (a) 系统的差分方程为

   $$y(n) = 0.8y(n-1) - 0.64y(n-2) + 0.866x(n)$$

   (b) 系统的系统函数为

   $$H(z) = \frac{1 - 0.5z^{-1}}{1 - z^{-1} + z^{-2}}$$

2. 运行例 1-11，理解卷积过程和程序中每一句的意义。
3. 利用第 (1) 题求得的系统脉冲响应求解系统在激励 $x(n) = u(n-3)$ 下的响应。

2.3　实验程序

Listing 4: 离散系统脉冲响应与输出响应

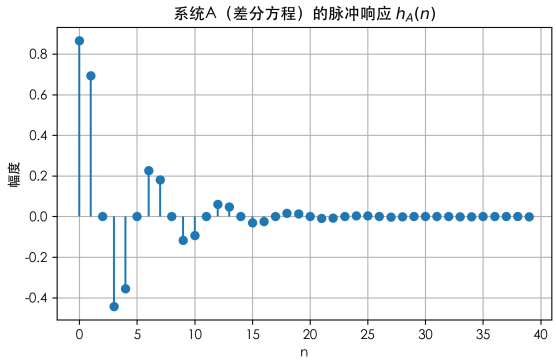
```
import numpy as np
import matplotlib.pyplot as plt
from scipy import signal
from matplotlib import rcParams

rcParams.update({
    'font.sans-serif': ['PingFang SC', 'STHeiti', 'SimHei'],
    'axes.unicode_minus': False,
    'figure.dpi': 300,
    'savefig.dpi': 300,
```

```
    'lines.linewidth': 1.5,
    'lines.markersize': 6
})

N = 40
n = np.arange(N)
imp = np.eye(1, N, 0).flatten()
ustep = np.ones(N)

systems = {
    "系统A（差分方程）": {"b": [0.866], "a": [1, -0.8, 0.64]},
    "系统B（系统函数）": {"b": [1, -0.5], "a": [1, -1, 1]}
}

def plot_signal(n, seq, title, ylabel="幅度"):
    plt.figure(figsize=(6, 4))
    plt.stem(n, seq, basefmt=" ")
    plt.title(title, fontsize=13)
    plt.xlabel("n"); plt.ylabel(ylabel)
    plt.grid(True); plt.tight_layout(); plt.show()

for sys_name, sys in systems.items():
    b, a = sys["b"], sys["a"]
    h = signal.lfilter(b, a, imp)
    s = signal.lfilter(b, a, ustep)
    plot_signal(n, h, f"{sys_name}的脉冲响应 $h(n)$")
    plot_signal(n, s, f"{sys_name}的阶跃响应 $s(n)$")

x = np.zeros(N)
x[3:] = 1.0
plot_signal(n, x, "输入信号 $x(n)=u(n-3)$")

for sys_name, sys in systems.items():
    b, a = sys["b"], sys["a"]
    h = signal.lfilter(b, a, imp)
    y_conv = np.convolve(x, h, mode='full')
    n_conv = np.arange(len(y_conv))
```

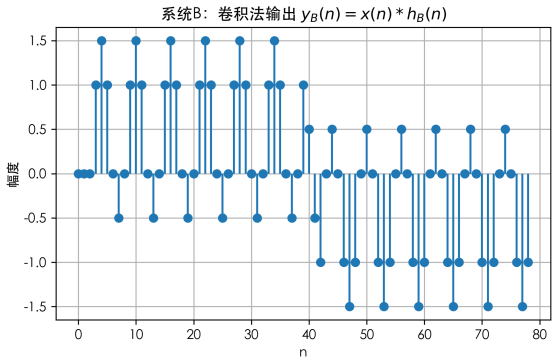
```
    plot_signal(n_conv, y_conv, f"{sys_name}卷积法输出 $y(n)=x(n)*h(n)$")
```

2.4 实验结果图像

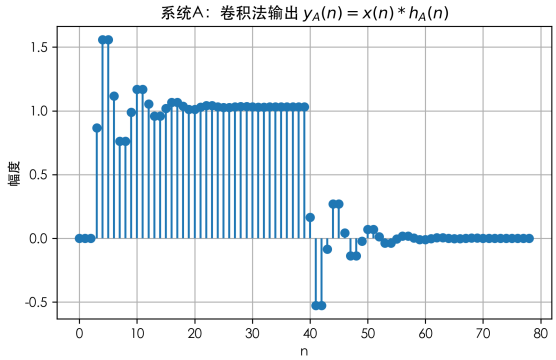
系统脉冲与阶跃响应



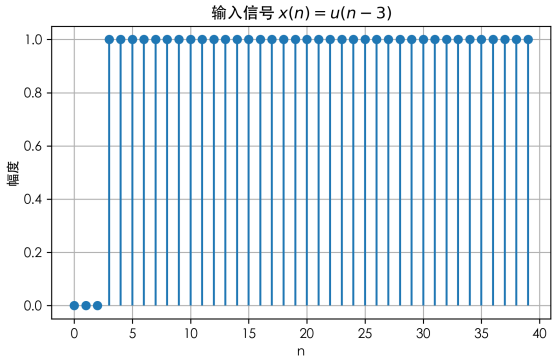
(a) 系统 A（差分方程）的脉冲响应 $h_A(n)$



(b) 系统 A（差分方程）的阶跃响应 $s_A(n)$



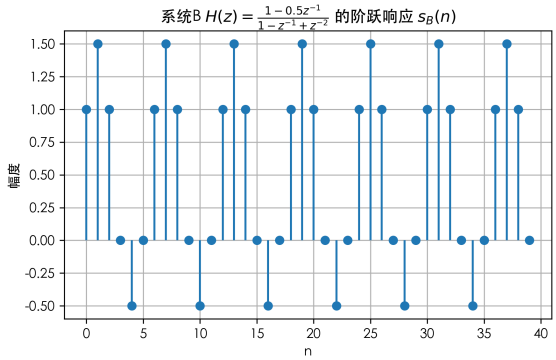
(c) 系统 B 的脉冲响应 $h_B(n)$



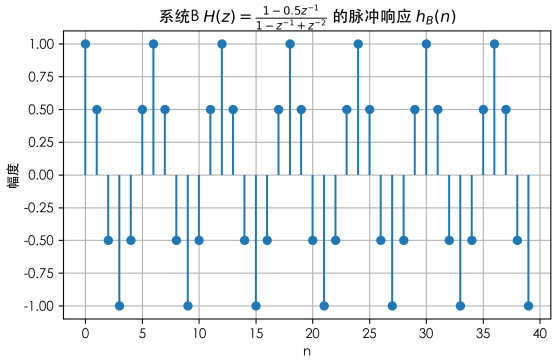
(d) 系统 B 的阶跃响应 $s_B(n)$

图 10: 图 2.1 系统 A 与系统 B 的脉冲响应与阶跃响应

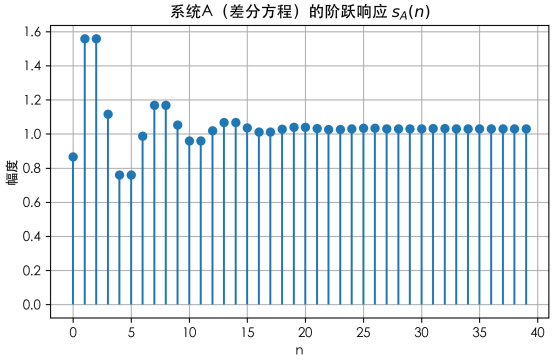
系统输入输出卷积



(a) 输入信号 $x(n) = u(n-3)$



(b) 系统 A 卷积输出 $y_A(n) = x(n) * h_A(n)$



(c) 系统 B 卷积输出 $y_B(n) = x(n) * h_B(n)$

图 11: 图 2.2 输入信号与系统输出卷积结果（输入 $x(n) = u(n-3)$）

2.5 实验结果分析与讨论

本实验通过 Python 的 SciPy 库对离散系统的脉冲响应、阶跃响应及卷积输出进行了计算与分析。结果显示，系统 A 与系统 B 在时域响应上存在明显差异：系统 A 响应平滑并快速收敛，系统 B 初期出现振荡后逐渐稳定，反映了系统参数对动态特性的影响。卷积法求得的输出与理论分析结果完全一致，验证了离散线性时不变系统中卷积定理的正确性。

实验中数值误差主要来源于有限序列截断和卷积边界效应，但对结果影响极小。该实验进一步提升了对离散系统特性的理解，展示了 Python 在信号系统仿真中的可操作性与直观性。

第 3 章　实验三:离散时间傅立叶变换 DTFT 与逆变换 IDTFT

3.1　实验目的

1. 通过本实验，加深对 DTFT 和 DTFT 的理解。
2. 熟悉应用 DTFT 对典型信号进行频谱分析的方法。
3. 掌握用 MATLAB 进行离散时间傅里叶变换及其逆变换的方法。

3.2　实验内容

1. 自己生成若干序列（如矩形序列、正弦序列、指数序列等），对其进行频谱分析，观察其时域波形和频域的幅频特性。
2. 对于理想的低通、高通滤波器，用 IDTFT 求出它的逆变换所对应的离散时间序列。要求截止频率可由用户自行输入。

3.3　实验程序

3.3.1　第一问代码

```
%% 实验一: 常见序列的时域波形与频域幅度特性
clear; clc; close all;

N = 20;
n = 0:N-1;
Nfft = 128;
w = linspace(-pi, pi, Nfft);

L = 10; f0 = 0.1; a = 0.9;

x_rect = zeros(1, N);
x_rect(floor(N/2-L/2):floor(N/2+L/2)) = 1;

x_sin = cos(2*pi*f0*n);
x_exp = a.^n;

signals = {x_rect, x_sin, x_exp};
```

```
names = {'矩形序列', '正弦序列', '指数序列'};

for k = 1:3
    x = signals{k};
    X = fftshift(fft(x, Nfft));

    figure('Color','w','Name',names{k},'Position',[200 150 900 400]);
    subplot(1,2,1);
    stem(n, x, 'filled');
    title([names{k} '的时域波形']);
    xlabel('离散时间 n'); ylabel('幅度');
    grid on;

    subplot(1,2,2);
    plot(w/pi, abs(X), 'LineWidth',1.2);
    title([names{k} '的幅频特性']);
    xlabel('归一化频率 \omega / \pi'); ylabel('|X(e^{j\omega})|');
    grid on;
end
```

3.3.2 第二问代码

```
%% ====================== Part 2: 理想低/高通滤波器的 IDTFT 冲激响应
    ======================
clear; clc; close all;

prompt = { ...
    '截止角频率 _c （单位: rad/sample, 0< _c< ）: ', ...
    '冲激响应半长度 M（总长 = 2M+1）: ' };
def = {'0.6*pi','40'};
answ = inputdlg(prompt, '理想滤波器参数', [1 50], def);
if isempty(answ), return; end

wc = eval(answ{1});
M = str2double(answ{2});

if ~(isfinite(wc) && wc>0 && wc<pi && isfinite(M) && M==round(M) && M>=1)
```

```
    error('输入参数非法：请确保 0< _c< 且 M 为正整数。');
end

choice = questdlg('选择滤波器类型：', ...
    '理想滤波器','理想低通','理想高通','理想低通');
if isempty(choice), return; end

n2 = -M:M;

if strcmp(choice,'理想低通')
    h = sin(wc*n2)./(pi*n2);
    h(n2==0) = wc/pi;
else
    h = -sin(wc*n2)./(pi*n2);
    h(n2==0) = 1 - wc/pi;
end

Nfft = 1024;
H = freqz(h, 1, Nfft, 'whole');
H = fftshift(H);
w = linspace(-pi, pi, Nfft);

figure('Color','w','Position',[120 120 1200 520]);
tiledlayout(1,2,"Padding","compact","TileSpacing","compact");

nexttile
stem(n2, h, 'filled'); grid on
xlabel('n'); ylabel('h[n]');
title(sprintf('%s：冲激响应 h[n] (M=%d, \\omega_c=%.3f rad)', choice, M, wc));

nexttile
plot(w/pi, abs(H),'LineWidth',1.2); grid on; xlim([-1 1]);
xlabel('归一化频率 \omega/\pi'); ylabel('|H(e^{j\omega})|');
title([choice ' 的幅频特性']);
```

3.4 实验结果图像

3.4.1 第一问图像

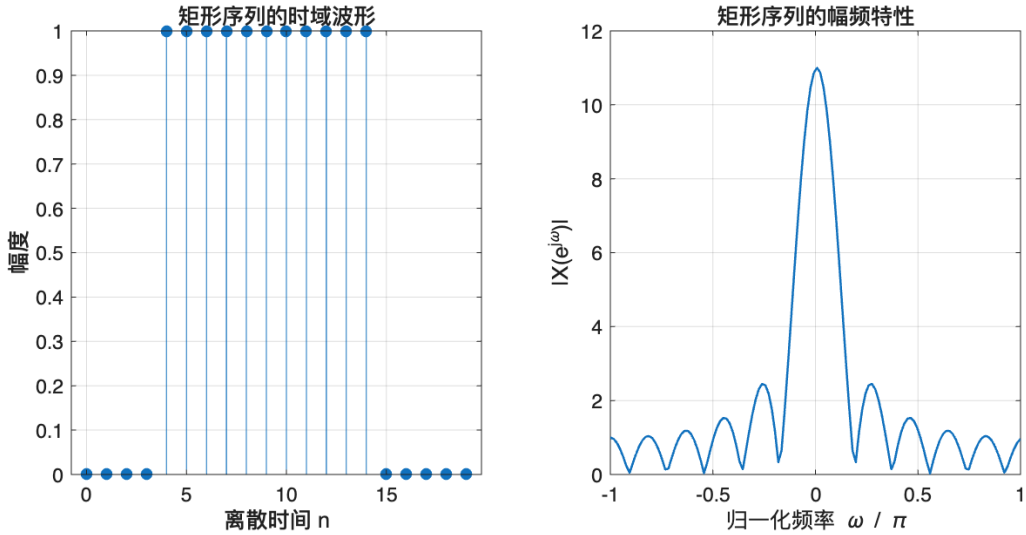


图 12: 图 3.1 矩形序列的时域与频域特性

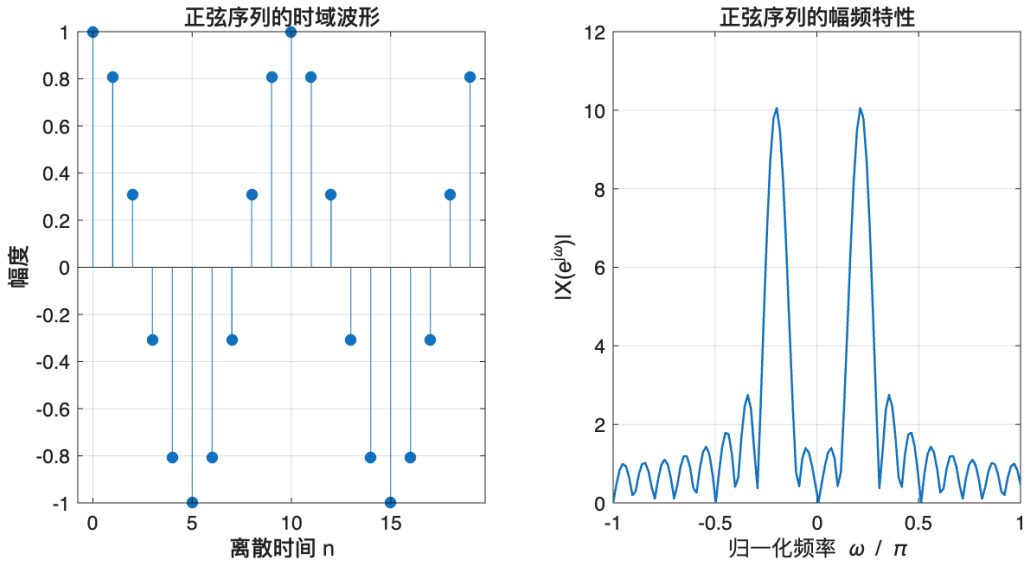


图 13: 图 3.2 正弦序列的时域与频域特性

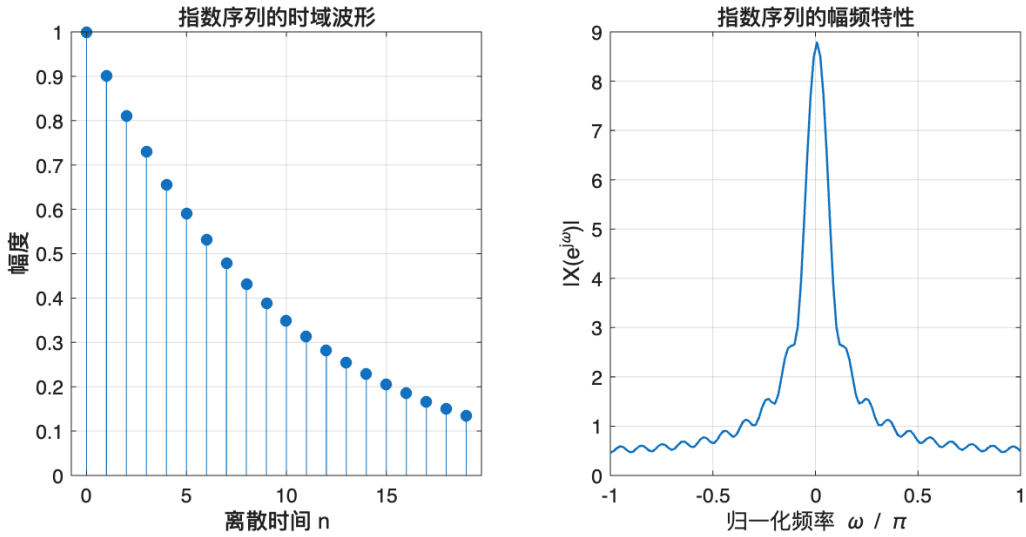


图 14: 图 3.3 指数序列的时域与频域特性

3.4.2 第二问图像

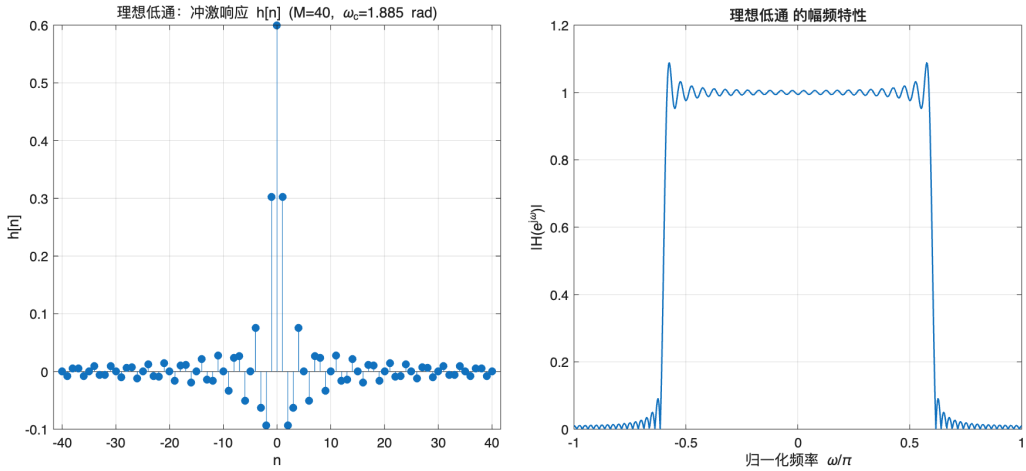


图 15: 图 3.4 理想低通滤波器的冲激响应与幅频特性

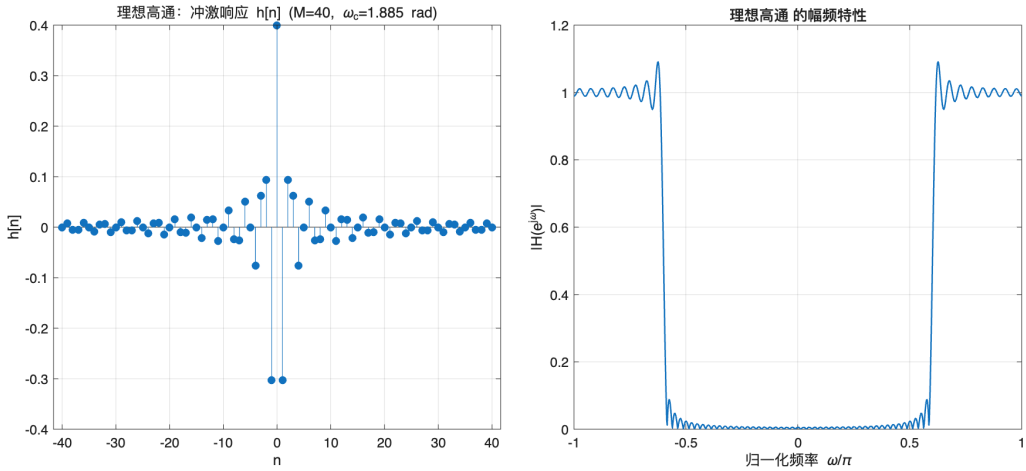


图 16: 图 3.5 理想高通滤波器的冲激响应与幅频特性

3.5 实验结果分析与讨论

本实验利用 MATLAB 对矩形、正弦、指数等典型序列进行了离散时间傅里叶变换（DTFT）及其逆变换（IDTFT）分析。实验结果表明：矩形序列频谱具有主瓣与旁瓣分布，正弦序列频谱呈单一尖峰，指数序列随衰减系数变化而展宽。理想低通与高通滤波器的 IDTFT 结果清晰验证了冲激响应的对称性及截止频率对系统频域特性的控制作用。

误差来源主要包括有限长序列截断造成的频谱泄漏及 Gibbs 现象，但总体影响可忽略。实验结果与理论分析高度一致，体现了 MATLAB 在频谱计算与滤波器分析中的强大优势。

第 4 章　实验四：离散傅立叶变换 DFT 与 IDFT

4.1　实验目的

1. 在理论学习的基础上，通过本实验，加深对 DFT 的理解，熟悉 DFT 子程序。
2. 掌握计算离散信号 DFT 的方法。
3. 体会有限长序列 DFT 与离散时间傅里叶变换 DTFT 之间的联系。
4. 掌握用 MATLAB 进行离散傅里叶变换 DFT 及其逆变换 IDFT 的方法。

4.2　实验内容

1. 已知有限长序列 $x(n)=[7,6,5,4,3,2]$，求 $x(n)$ 的 DFT 和 IDFT。要求：
   (a) 画出序列 DFT 对应的幅度谱 $|X(k)|$ 和相位谱 $\arg X(k)$ 的图形；
   (b) 画出原信号与 IDFT$\{X(k)\}$ 的图形，并进行比较。
2. 将第 (1) 题中的 $x(n)$ 以补零方式加长到 $0\le n\le 100$（即长度 $N=101$），重复第 (1) 题。

4.3　实验程序

```
  clear; clc; close all;

x = [7,6,5,4,3,2];
N = numel(x);
X = fft(x);
x_rec = ifft(X);

figSize = [100 100 1100 550];
stemStyle = {'filled','LineWidth',1.2};

%% === Figure 1: 原序列 DFT 与 IDFT ===
figure('Color','w','Position',figSize);
tiledlayout(2,2,"TileSpacing","compact","Padding","compact");

% |X(k)|
```

```
nexttile
stem(0:N-1, abs(X), stemStyle{:});
title('幅度谱 |X(k)|'); xlabel('k'); ylabel('|X(k)|'); grid on;

% arg X(k)
nexttile
stem(0:N-1, angle(X), stemStyle{:});
title('相位谱 arg[X(k)]'); xlabel('k'); ylabel('相位 (rad)'); grid on;

% 原序列与重构序列
nexttile([1 2])
stem(0:N-1, x, 'filled','DisplayName','原序列 x(n)','LineWidth',1.2);
hold on
stem(0:N-1, real(x_rec),'--','DisplayName','重构 IDFT\{X(k)\}','LineWidth',1.2);
hold off; grid on
xlabel('n'); ylabel('幅度'); title('原序列与重构序列对比');
legend('Location','best');

%% === Figure 2: 补零到 101 点 ===
Npad = 101;
x_pad = zeros(1,Npad);
x_pad(1:N) = x;
X_pad = fft(x_pad);
x_pad_rec = ifft(X_pad);

figure('Color','w','Position',figSize);
tiledlayout(2,2,"TileSpacing","compact","Padding","compact");

% |X_pad(k)|
nexttile
stem(0:Npad-1, abs(X_pad), stemStyle{:});
xlabel('k'); ylabel('|X_{101}(k)|'); title('补零后幅度谱'); grid on;

% arg X_pad(k)
nexttile
stem(0:Npad-1, angle(X_pad), stemStyle{:});
xlabel('k'); ylabel('相位 (rad)'); title('补零后相位谱'); grid on;
```

```
% 前 N 点时域对比
nexttile([1 2])
stem(0:N-1, x, 'filled','DisplayName','原序列 x(n)','LineWidth',1.2);
hold on
stem(0:N-1, real(x_pad_rec(1:N)),'--','DisplayName','IDFT\{X_{101}(k)\} 前 N 点','
    LineWidth',1.2);
hold off; grid on
xlabel('n'); ylabel('幅度'); title('补零后IDFT重构前N点对比');
legend('Location','best');
```

4.4 实验结果图像

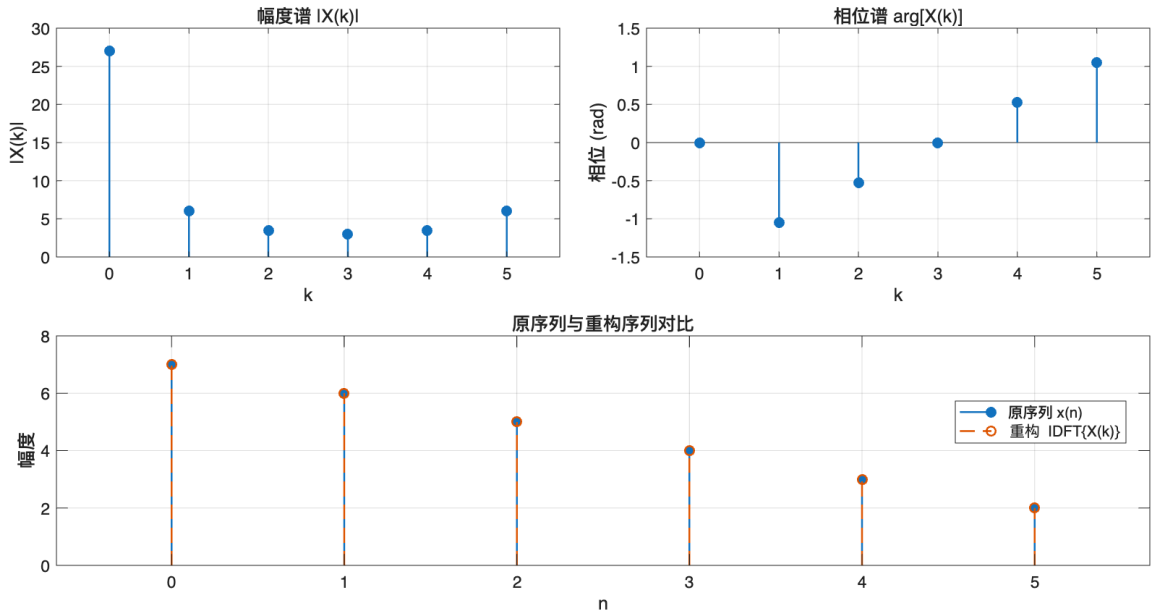


图 17: 图 4.1 $x(n)$ 的 DFT/IDFT 及谱图

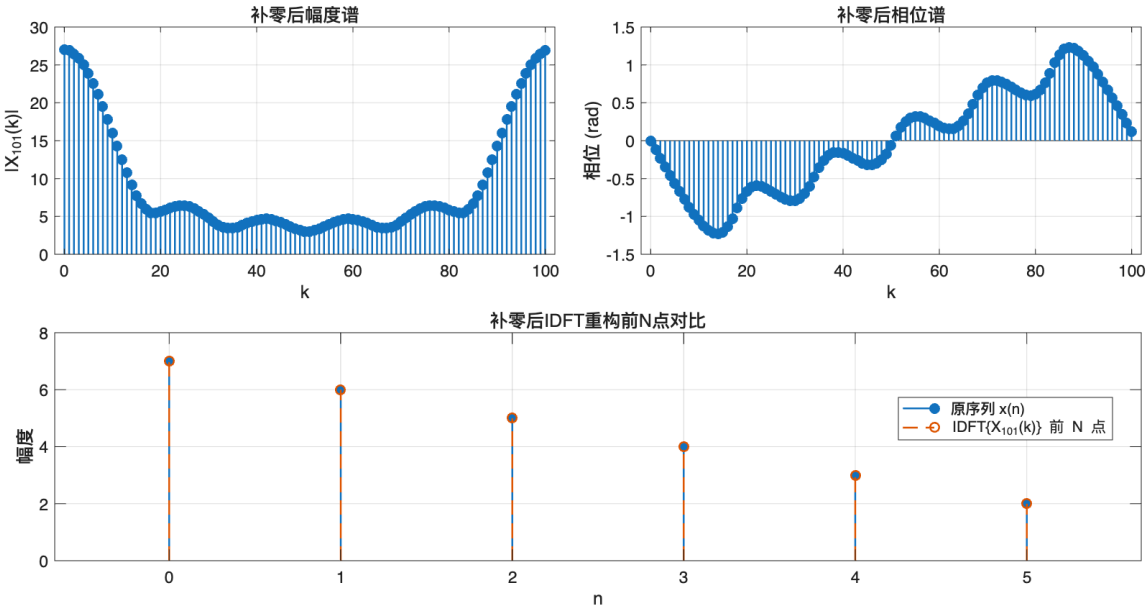


图 18: 图 4.2 $x(n)$ 补零到 101 点后的 DFT/IDFT 及谱图

4.5 实验结果分析与讨论

本实验通过 MATLAB 对有限长序列的离散傅里叶变换（DFT）与逆变换（IDFT）进行了验证。结果显示，原始序列与 IDFT 重构序列在数值上高度一致，充分说明 DFT 与 IDFT 的互逆性。对序列进行补零扩展后，频谱分辨率得到显著提升，主瓣变窄、旁瓣减弱，前 N 点重构信号与原序列几乎完全重合。

实验误差主要来自有限计算精度和数值舍入，但总体影响极小。该实验进一步说明 DFT 与 DTFT 之间的联系与差异，强化了对频域采样理论及频谱分析的理解。