

**本科生实验报告**

**实验课程 数字信号处理**

**学院名称 核技术与自动化工程学院**

**专业名称 测控技术与仪器**

**学生姓名 芦博宇**

**学生学号 202006010328**

**指导教师 葛青**

**实验地点 6C802**

**实验成绩**

**二〇 年 月 二〇 年 月**

**填写说明**

1. 适用于本科生所有的实验报告（印制实验报告册除外）；
2. 专业填写为专业全称，有专业方向的用小括号标明；
3. 格式要求：
4. 用A4纸双面打印（封面双面打印）或在A4大小纸上用蓝黑色水笔书写。
5. 打印排版：正文用宋体小四号，1.5倍行距，页边距采取默认形式（上下2.54cm，左右2.54cm，页眉1.5cm，页脚1.75cm）。字符间距为默认值（缩放100%，间距：标准）；页码用小五号字底端居中。
6. 具体要求：

**题目**（二号黑体居中）；

**摘要**（“摘要”二字用小二号黑体居中，隔行书写摘要的文字部分，小4号宋体）；

**关键词**（隔行顶格书写“关键词”三字，提炼3-5个关键词，用分号隔开，小4号黑体)；

正文部分采用三级标题；

**第1章** ××(小二号黑体居中，段前0.5行)

**1.1** ×××××小三号黑体×××××（段前、段后0.5行）

**1.1.1**小四号黑体（段前、段后0.5行）

**参考文献**（黑体小二号居中，段前0.5行），参考文献用五号宋体，参照《参考文献著录规则（GB/T 7714－2005）》。

**实验报告中要求写出程序并画出波形**

“数字信号处理”课程主要是学习用数学的方法对信号进行处理,涉及到大量的算法，通过单一的课堂讲授很难掌握其中的基本理论和算法。在理论教学的同时进行实验不仅可加深对基本理论和算法的理解、掌握用计算机解决数字信号处理问题的方法，而且可以培养分析问题、解决问题的能力。

**实验1离散时间信号的时域及频域分析**

1. **实验目的**

加深对常用离散信号的理解；掌握一些常用离散信号在时域中的描述及绘图；掌握离DTFT换的定义、计算及绘图。

**二、内容**

1. 编写程序产生下述信号，并绘出其图形,要求标注xlabel；ylabel；title。

(1) 单位冲激序列δ(n-7)。

(2) 单位阶跃序列u(n-7)；

(3) 正弦函数；

(4) 指数序列0.4n；

1. N=10;

2. t=0:N;

3. x1=zeros(1,N+1);

4. x1(8)=1;

5. subplot(2,2,1),stem(t,x1);

6. title('单位冲激序列');

7. xlabel('n');ylabel('幅值');

8.

9. x2=[0,0,0,0,0,0,0,1,1,1,1];

10. subplot(2,2,2),stem(t,x2);

11. title('单位阶跃序列');

12. xlabel('n');ylabel('幅值');

13.

14. N3=500;

15. t3=0:N3;

16. y3=sin((pi/100).\*t3+pi/2);

17. subplot(2,2,3),plot(t3,y3);axis([-1,N3+1,-1.1,1.1]);

18. title('正弦函数');

19. xlabel('x');ylabel('幅值');

20.

21. N4=10;

22. t4=0:N4;

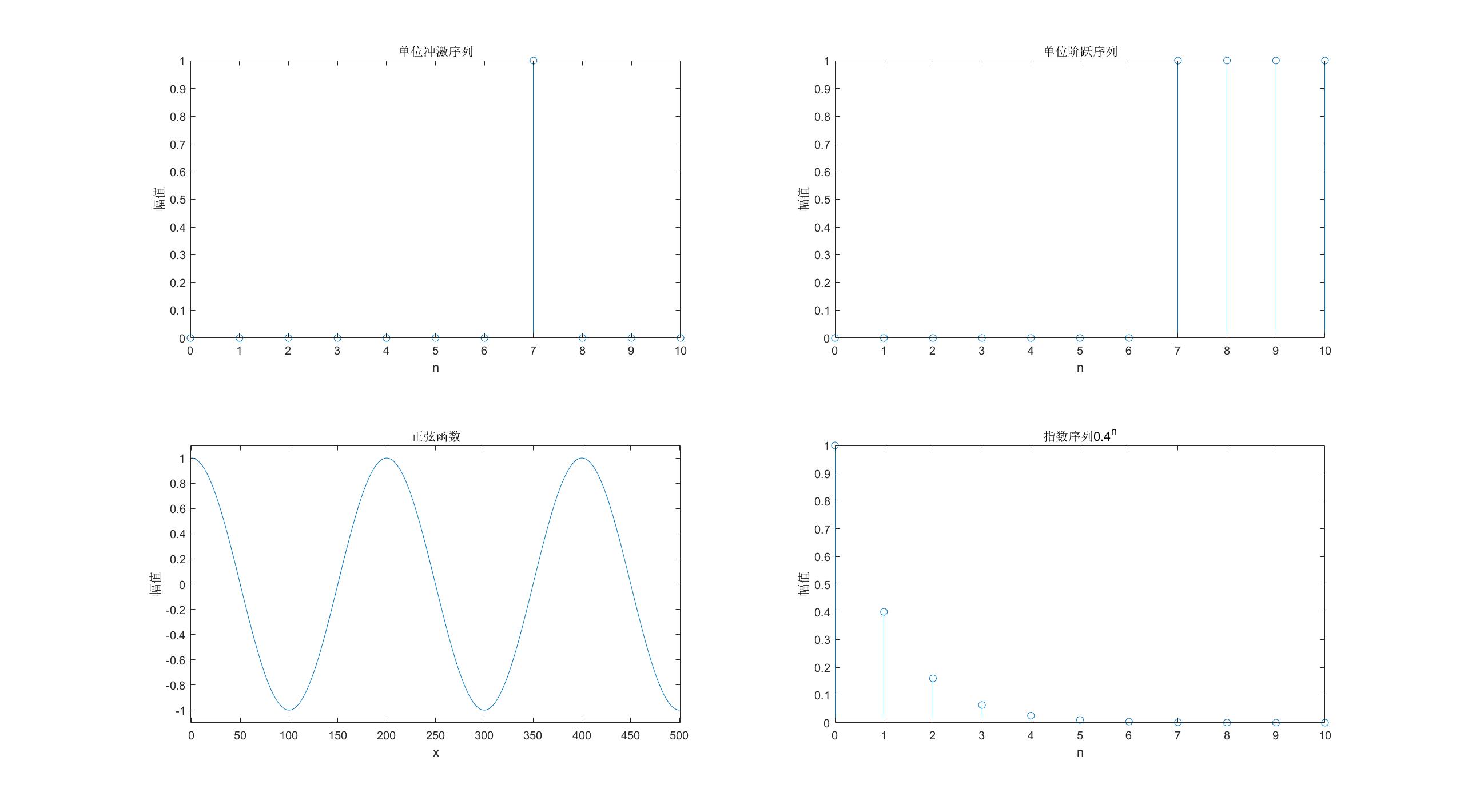
23. y4=0.4.^t4;

24. subplot(2,2,4),stem(t4,y4);

25. % axis([-1,N3+1,-1.1,1.1]);

26. title('指数序列0.4^n');

27. xlabel('n');ylabel('幅值');



2、x1(n)=cos(800πn)，x2(n)=cos(800n)，判断x1(n) 及x2(n) 是否都是周期信号(理论分析)？如果是周期信号，周期是多少？画出这两个信号的波形。

x1是周期信号，周期为1/400；x2不是周期信号。

1. N1=10;

2. t1=0:N1;

3. y1=cos(800\*pi\*t1);

4. subplot(1,2,1),stem(t1,y1);

5. title('cos(800pi\*n)');

6. xlabel('n');ylabel('幅值');

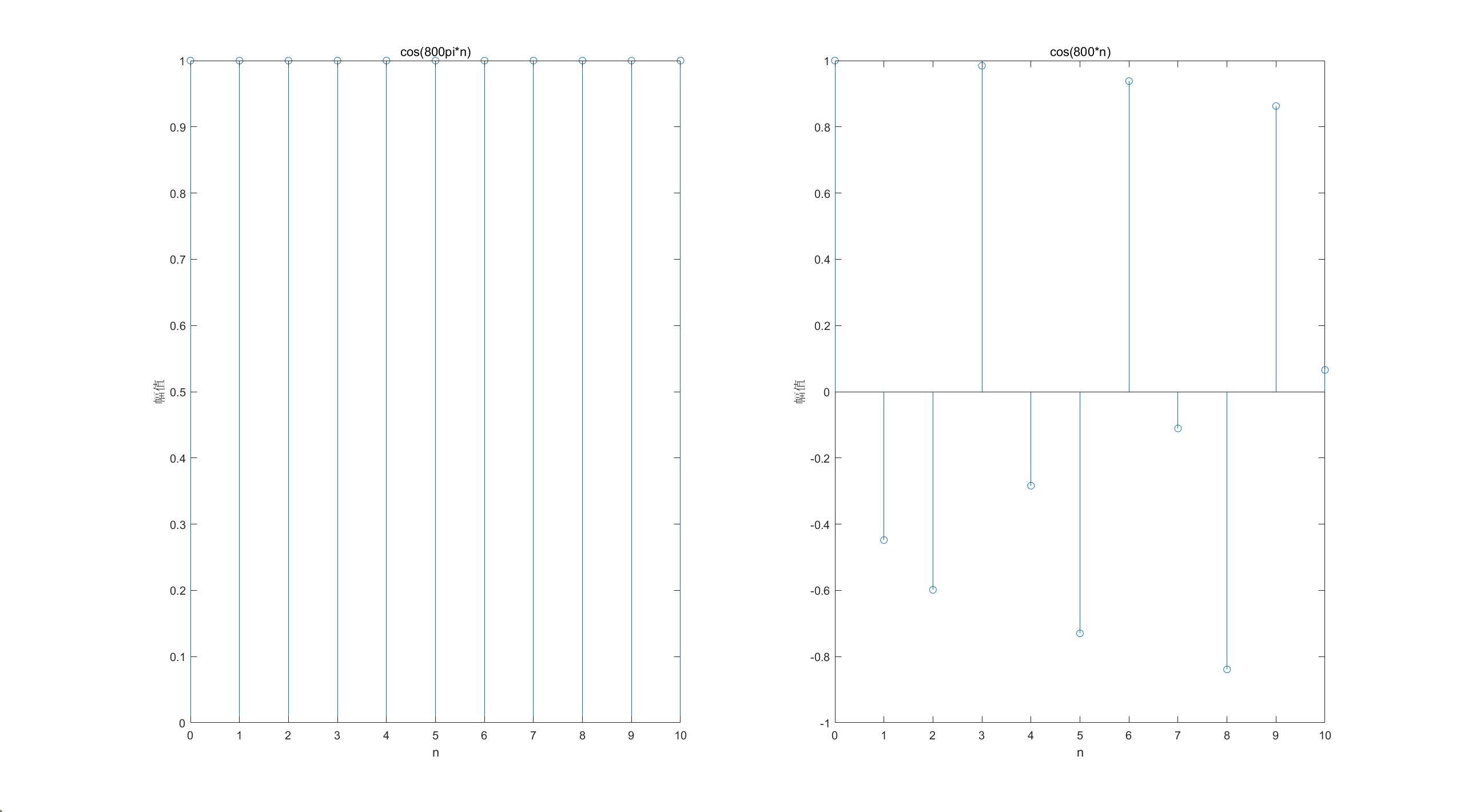
7.

8. y2=cos(800\*t1);

9. subplot(1,2,2),stem(t1,y2);

10. title('cos(800\*n)');

11. xlabel('n');ylabel('幅值');



3、已知 ，画出该信号的时域波形和DTFT(离散时间傅里叶变换)的幅度谱 ？

1. clear;

2. clc;

3. N=10;

4. t1=0:0.01:N;

5. t1=t1(1:N\*100);

6. un=ones(1,N\*100);

7. xn=(6.^t1).\*un;

8. subplot(1,2,1),stem(t1,xn);

9. title('6^n\*u(n)');

10. xlabel('n');ylabel('幅值');

11. b=[1];

12. a=[1,-6];

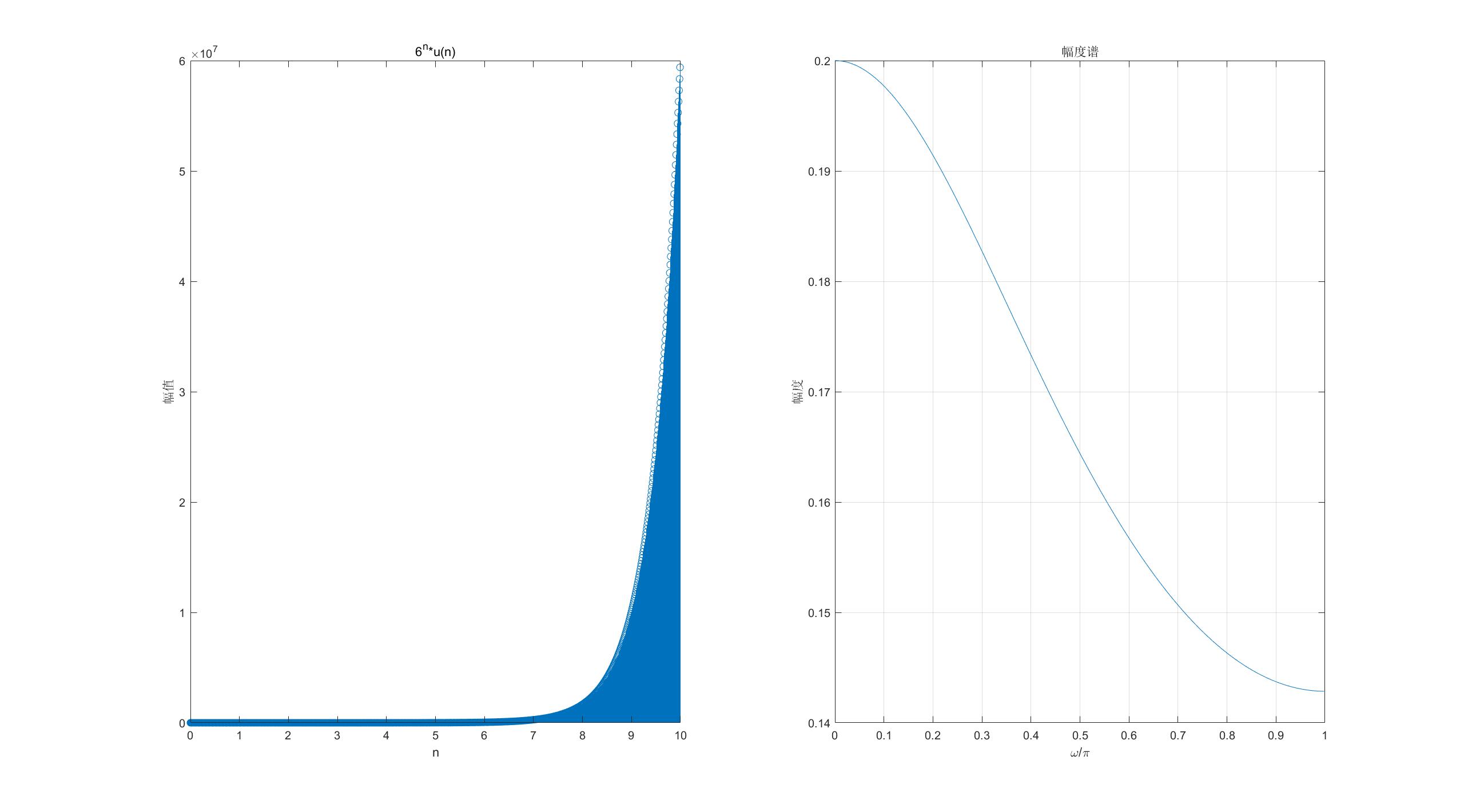
13. [h,w]=freqz(b,a);

14. subplot(1,2,2),

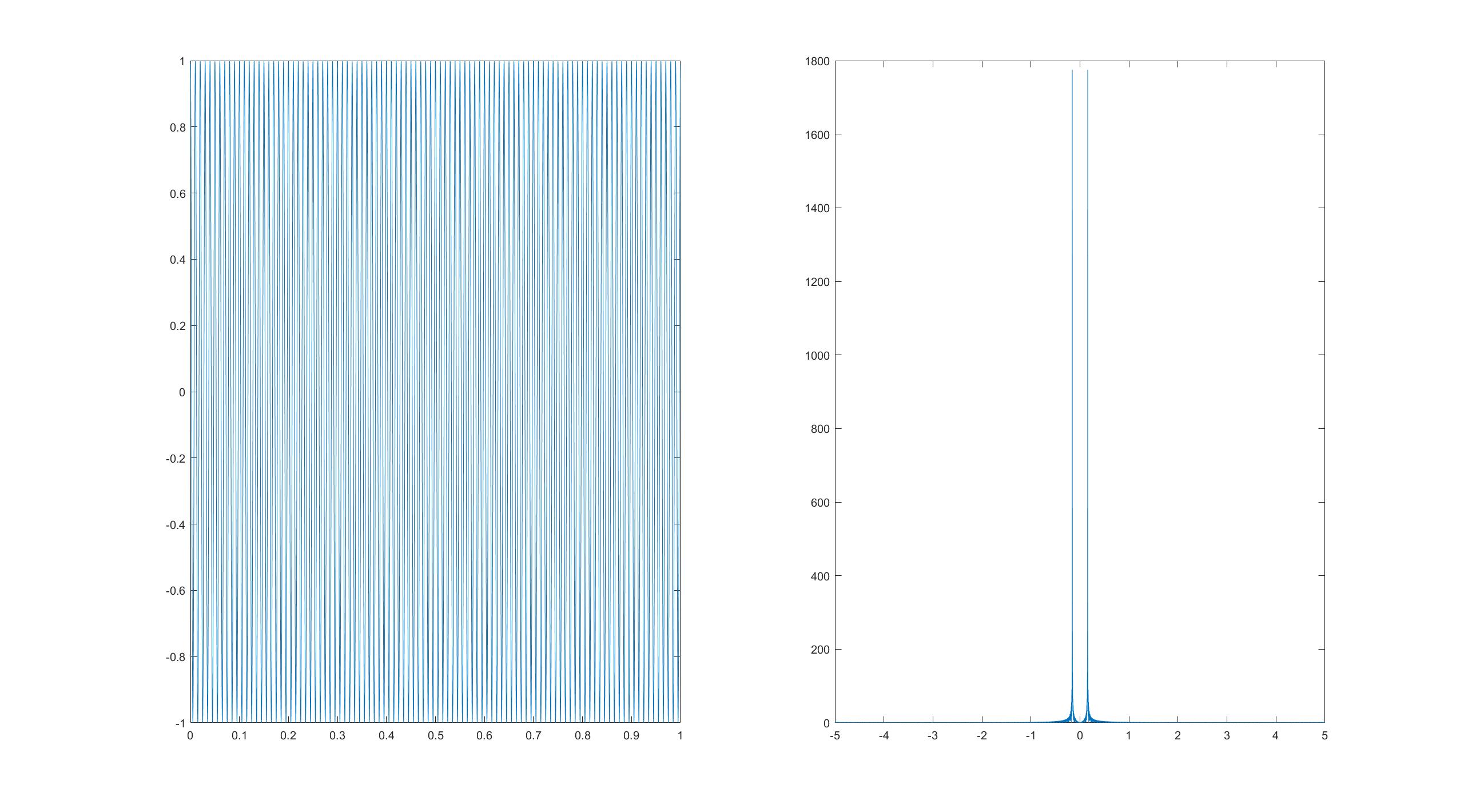
15. plot(w/pi,abs(h));grid

16. title('幅度谱')

17. xlabel('\omega/\pi');ylabel('幅度')



4、已知连续时间信号 , 以采样频率4000Hz对x(t)进行采样，画出采样信号x(n)的时域波形以及采样信号DTFT的幅度谱 。



1. function X = myDTFT(x, w)

2. N = length(x);

3. X = zeros(size(w));

4. for k = 1:length(w)

5. for n = 1:N

6. X(k) = X(k) + x(n)\*exp(-1i\*w(k)\*(n-1));

7. end

8. end

9. end

**思考题：**

* + - 1. 说明DTFT（离散时间傅里叶变换）与z变换的关系？
      2. 说明对连续时间信号进行采样时，采样频率有什么要求？

1.z变换是DTFT的一种拓展形式。DTFT是对x(t)进行离散化处理得到x(n)，进而对其进行傅里叶变换。

2.根据奈奎斯特采样定理，采样率要大于信号最高频率的二倍。

**实验2 FFT及卷积的计算**

1. **实验目的**

掌握信号的DFT的计算及绘图；掌握DTFT与DFT之间的关系；掌握线性卷积及圆周卷积的定义、计算及在MATLAB中的实现方法；掌握用圆周卷积代替线性卷积的条件。

**二. 实验内容**

1、 ，分别画出 的DTFT及8点、16点和128点的DFT，对比实验结果，说明在 后补零的作用。



1. clc;

2. clear;

3. n=1000;

4. x=[ones(1,7),zeros(1,n-7)];

5. w=linspace(0,2\*pi,n);

6. X=myDTFT(x,w);

7. subplot(2,2,1),

8. plot(w,X);

9. title('DTFT');

10.

11. N=8;

12. xx=[1 1 1 1];

13. X\_8=abs((fft(xx,8)));

14. w=0:2\*pi/(N-1):2\*pi;

15. subplot(2,2,2),

16. stem(w/pi,X\_8);

17. title('8点DFT');

18.

19. N=16;

20. X\_16=abs((fft(xx,16)));

21. w=0:2\*pi/(N-1):2\*pi;

22. subplot(2,2,3),

23. stem(w/pi,X\_16);

24. title('16点DFT');

25.

26. N=128;

27. X\_128=abs((fft(xx,128)));

28. w=0:2\*pi/(N-1):2\*pi;

29. subplot(2,2,4),

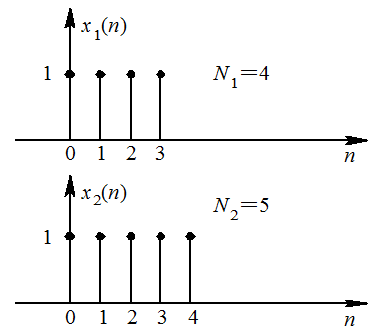
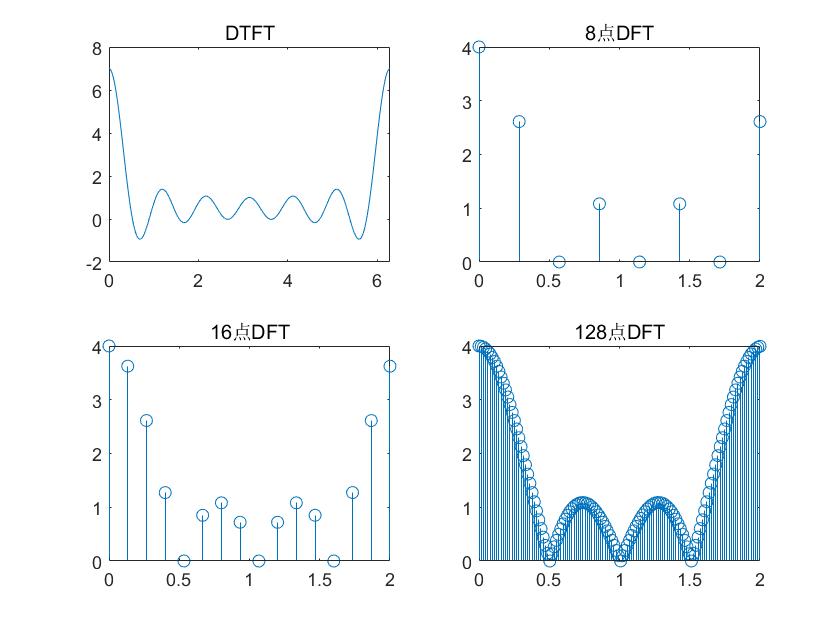
30. stem(w/pi,X\_128);

31. title('128点DFT');

32.

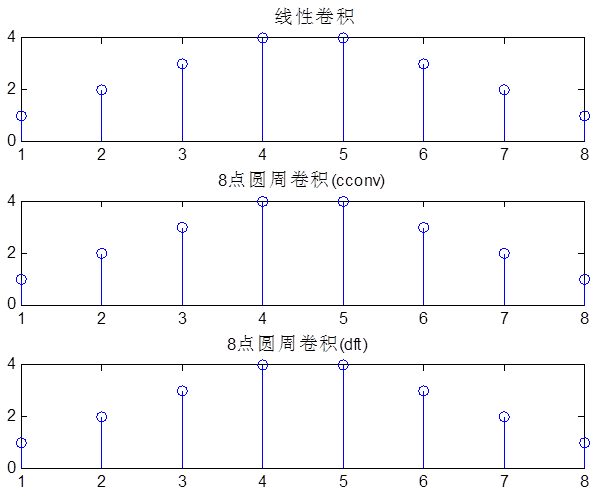
33. % 补零可以让结果变得平滑

2、分别编程计算两个矩形（见图所示）的线性卷积和圆周卷积，圆周卷积L分别取7、8、10（圆周卷积的计算采用两种方法，cconv及DFT，其波形模式见图2-2所示)，画出线性卷积、圆周卷积的波形；



**注意：**

波形的标题见下图所示，分别为：线性卷积、7点圆周卷积(cconv)、7点圆周卷积(DFT)；线性卷积、8点圆周卷积(cconv)、8点圆周卷积(DFT)；线性卷积、10点圆周卷积 (cconv)、10点圆周卷积(DFT)；



1. clc;

2. clear;

3. x1=[1,1,1,1];

4. x2=[1,1,1,1,1];

5. y1=conv(x1,x2);

6. n=length(y1);

7. subplot(3,1,1),

8. stem(1:n-1,y1(1:7));

9. title('线性卷积');

10. y\_7=cconv(x1,x2,7);

11. subplot(3,1,2),

12. stem(1:7,y\_7);

13. title('7点圆周卷积cconv');

14. X1E=fft(x1,n-1);

15. X2E=fft(x2,n-1);

16. y\_7\_dft=ifft(X1E.\*X2E);

17. subplot(3,1,3),

18. stem(1:7,y\_7\_dft);

19. title('7点圆周卷积DFT');

20.

21. figure;

22. subplot(3,1,1),

23. stem(1:n,y1);

24. title('线性卷积');

25. y\_8=cconv(x1,x2,8);

26. subplot(3,1,2),

27. stem(1:8,y\_8);

28. title('8点圆周卷积cconv');

29. X1E=fft(x1,n);

30. X2E=fft(x2,n);

31. y\_8\_dft=ifft(X1E.\*X2E);

32. subplot(3,1,3),

33. stem(1:8,y\_8\_dft);

34. title('8点圆周卷积DFT');

35.

36. figure;

37. y1=[y1 0 0 ];

38. subplot(3,1,1),

39. stem(1:10,y1);

40. title('线性卷积');

41. y\_10=cconv(x1,x2,10);

42. subplot(3,1,2),

43. stem(1:10,y\_10);

44. title('10点圆周卷积cconv');

45. X1E=fft(x1,10);

46. X2E=fft(x2,10);

47. y\_10\_dft=ifft(X1E.\*X2E);

48. subplot(3,1,3),

49. stem(1:10,y\_10\_dft);

50. title('10点圆周卷积DFT');

|  |  |  |
| --- | --- | --- |
|  |  |  |

3. 若已知有限长序列，求该信号的8点、16点DFT，再由所得结果求IDFT，**说明对信号进行DFT至少要进行多少点的？**



1. clc;

2. clear;

3. X=[1 2 -1 3];

4. XE\_8=fft(X,8)

5. XE\_16=fft(X,16)

6. Xi\_8=ifft(XE\_8)

7. Xi\_16=ifft(XE\_16)

XE\_8 =

5.0000 + 0.0000i 0.2929 - 2.5355i 2.0000 + 1.0000i 1.7071 - 4.5355i -5.0000 + 0.0000i 1.7071 + 4.5355i 2.0000 - 1.0000i 0.2929 + 2.5355i

XE\_16 =

5.0000 + 0.0000i 3.2887 - 2.8299i 0.2929 - 2.5355i -0.2992 + 0.0074i 2.0000 + 1.0000i 3.7134 - 1.4068i 1.7071 - 4.5355i -2.7029 - 4.2441i -5.0000 + 0.0000i -2.7029 + 4.2441i 1.7071 + 4.5355i 3.7134 + 1.4068i 2.0000 - 1.0000i -0.2992 - 0.0074i 0.2929 + 2.5355i 3.2887 + 2.8299i

Xi\_8 =

1.0000 2.0000 -1.0000 3.0000 0 -0.0000 0 -0.0000

Xi\_16 =

1.0000 2.0000 -1.0000 3.0000 0 -0.0000 0 -0.0000 0 0 0 0.0000 0 -0.0000 0 -0.0000

**思考题：**

1. 线性卷积和圆周卷积的结果有何不同？要使圆周卷积等于线性卷积而不产生混叠的必要条件是什么？说明用DFT计算线性卷积需满足什么条件？

线性卷积和圆周卷积的结果长度不同。要使圆周卷积等于线性卷积而不产生混淆，必要条件是：L≥N+M-1。用DFT计算线性卷积也需满足此条件。

1. 说明DTFT与DFT的关系？

DTFT是离散时间傅里叶变换，DFT是离散傅里叶变换。DTFT变换后的图形中的频率是一般连续的，而DFT是DTFT的等间隔抽样，是离散的点。

1. 为了能在数字计算机上分析连续信号的频谱，常常用DFT 来逼近连续时间信号的傅里叶变换，但同时也产生以下问题：混叠现象、频谱泄漏、栅栏效应，说明这几类现象产生的原因及减小该现象的措施。

混叠现象是由于采样频率不足导致的，增加采样频率可以减小该现象；

频谱泄漏是由于信号不是带限信号或采样频率不满足采样定理导致的，增加采样点数可以减小该现象；

而栅栏效应则是由于FFT点数不足导致的，使用窗函数可以减小该现象。

**实验3 基于FPGA的FFT及IFFT的实现**

1. **实验目的**
2. 掌握基2按时间抽取FFT算法的原理及在FPGA中实现的方法；
3. 掌握IFFT算法的原理及在FPGA中实现的方法；

**二. 实验内容**

**在FPGA中实现下面的运算**

**FFT**

**x(n)**

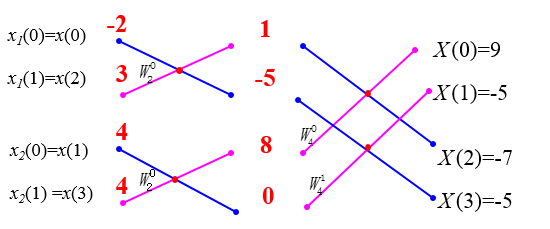
**IFFT**

**X(k)**

**x(n)**

**x(n)={-2 , 4 , 3 , 4 }**

**x(n)={-2 , 4 , 3 , 4 }**



**1. 基于FPGA的FFT的实现**

（1）将待变换的信号在MATLAB中进行FFT运算（用于与FPGA中计算结果进行对比）；

（2）将待变换的信号x(n)存储到FPGA的存储器中；

（3）在FPGA中将存储器中的数据读出并在FPGA中进行FFT运算，得到X(k)；

（4）在MODELSIM中进行仿真，并将仿真结果与MATLAB中运算的结果进行对比，验证运算是否正确。

**2. 基于FPGA的IFFT的实现**

（1）对1中计算的X(k)进行IFFT运算；

（4）在中进行仿真，并将仿真结果x(n)进行对比，验证运算是否正确。

FPGA代码：

1. module FFT(

2. input clk,

3. input rst\_n,

4. input data\_in\_en,

5.

6.

7. output reg data\_out\_en,

8. output reg signed [4:0] fft\_data\_out\_re0,

9. output reg signed [4:0] fft\_data\_out\_im0,

10. output reg signed [4:0] fft\_data\_out\_re1,

11. output reg signed [4:0] fft\_data\_out\_im1,

12. output reg signed [4:0] fft\_data\_out\_re2,

13. output reg signed [4:0] fft\_data\_out\_im2,

14. output reg signed [4:0] fft\_data\_out\_re3,

15. output reg signed [4:0] fft\_data\_out\_im3,

16.

17. output reg signed [4:0] ifft\_data\_out\_re0,

18. output reg signed [4:0] ifft\_data\_out\_im0,

19. output reg signed [4:0] ifft\_data\_out\_re1,

20. output reg signed [4:0] ifft\_data\_out\_im1,

21. output reg signed [4:0] ifft\_data\_out\_re2,

22. output reg signed [4:0] ifft\_data\_out\_im2,

23. output reg signed [4:0] ifft\_data\_out\_re3,

24. output reg signed [4:0] ifft\_data\_out\_im3

25. );

26. //输入

27. reg signed [4:0] data\_in\_re0;

28. reg signed [4:0] data\_in\_im0;

29. reg signed [4:0] data\_in\_re1;

30. reg signed [4:0] data\_in\_im1;

31. reg signed [4:0] data\_in\_re2;

32. reg signed [4:0] data\_in\_im2;

33. reg signed [4:0] data\_in\_re3;

34. reg signed [4:0] data\_in\_im3;

35. //一级

36. reg fft1\_en;

37. reg signed [10:0] fft1\_re0;

38. reg signed [10:0] fft1\_im0;

39. reg signed [10:0] fft1\_re1;

40. reg signed [10:0] fft1\_im1;

41. reg signed [10:0] fft1\_re2;

42. reg signed [10:0] fft1\_im2;

43. reg signed [10:0] fft1\_re3;

44. reg signed [10:0] fft1\_im3;

45. initial begin//赋初值

46. data\_in\_re0<=-5'd2;

47. data\_in\_im0<=5'd1;

48. data\_in\_re1<=5'd4;

49. data\_in\_im1<=5'd2;

50. data\_in\_re2<=5'd3;

51. data\_in\_im2<=5'd0;

52. data\_in\_re3<=5'd4;

53. data\_in\_im3<=5'd0;

54. end

55. always @(posedge clk or negedge rst\_n)begin

56. if(!rst\_n)begin

57. fft1\_en<=1'b0;

58. fft1\_re0<=1'b0;

59. fft1\_im0<=1'b0;

60. fft1\_re1<=1'b0;

61. fft1\_im1<=1'b0;

62. fft1\_re2<=1'b0;

63. fft1\_im2<=1'b0;

64. fft1\_re3<=1'b0;

65. fft1\_im3<=1'b0;

66. end

67. else if(data\_in\_en)begin

68. fft1\_re0<=data\_in\_re0+data\_in\_re2;

69. fft1\_im0<=data\_in\_im0+data\_in\_im2;

70. fft1\_re1<=data\_in\_re0-data\_in\_re2;

71. fft1\_im1<=data\_in\_im0-data\_in\_im2;

72. fft1\_re2<=data\_in\_re1+data\_in\_re3;

73. fft1\_im2<=data\_in\_im1+data\_in\_im3;

74. fft1\_re3<=data\_in\_re1-data\_in\_re3;

75. fft1\_im3<=data\_in\_im1-data\_in\_im3;

76. fft1\_en<=1'b1;

77. end

78. else

79. fft1\_en<=1'b0;

80. end

81. //二级

82. reg ifft1\_en;

83. reg signed [10:0] fft2\_re0;

84. reg signed [10:0] fft2\_im0;

85. reg signed [10:0] fft2\_re1;

86. reg signed [10:0] fft2\_im1;

87. reg signed [10:0] fft2\_re2;

88. reg signed [10:0] fft2\_im2;

89. reg signed [10:0] fft2\_re3;

90. reg signed [10:0] fft2\_im3;

91. always @(posedge clk or negedge rst\_n)begin

92. if(!rst\_n)begin

93. fft2\_re0<=1'b0;

94. fft2\_im0<=1'b0;

95. fft2\_re1<=1'b0;

96. fft2\_im1<=1'b0;

97. fft2\_re2<=1'b0;

98. fft2\_im2<=1'b0;

99. fft2\_re3<=1'b0;

100. fft2\_im3<=1'b0;

101. end

102. else if(fft1\_en)begin

103. fft2\_re0<=fft1\_re0+fft1\_re2;

104. fft2\_im0<=fft1\_im0+fft1\_im2;//x0

105. fft2\_re1<=fft1\_re1+fft1\_im3;

106. fft2\_im1<=fft1\_im1-fft1\_re3;

107. fft2\_re2<=fft1\_re0-fft1\_re2;

108. fft2\_im2<=fft1\_im0-fft1\_im2;

109. fft2\_re3<=fft1\_re1-fft1\_im3;

110. fft2\_im3<=fft1\_im1+fft1\_re3;

111. ifft1\_en<=1'b1;

112. end

113. else

114. ifft1\_en<=1'b0;

115. end

116.

117. //ifft

118. reg ifft2\_en;

119. reg signed [10:0] ifft1\_re0;

120. reg signed [10:0] ifft1\_im0;

121. reg signed [10:0] ifft1\_re1;

122. reg signed [10:0] ifft1\_im1;

123. reg signed [10:0] ifft1\_re2;

124. reg signed [10:0] ifft1\_im2;

125. reg signed [10:0] ifft1\_re3;

126. reg signed [10:0] ifft1\_im3;

127. always @(posedge clk or negedge rst\_n)begin

128. if(!rst\_n)begin

129. ifft2\_en<=1'b0;

130. ifft1\_re0<=1'b0;

131. ifft1\_im0<=1'b0;

132. ifft1\_re1<=1'b0;

133. ifft1\_im1<=1'b0;

134. ifft1\_re2<=1'b0;

135. ifft1\_im2<=1'b0;

136. ifft1\_re3<=1'b0;

137. ifft1\_im3<=1'b0;

138. end

139. else if(ifft1\_en)begin

140. ifft1\_re0<=fft2\_re0+fft2\_re2;

141. ifft1\_im0<=fft2\_im0+fft2\_im2;

142. ifft1\_re1<=fft2\_re0-fft2\_re2;

143. ifft1\_im1<=fft2\_im0-fft2\_im2;

144. ifft1\_re2<=fft2\_re1+fft2\_re3;

145. ifft1\_im2<=fft2\_im1+fft2\_im3;

146. ifft1\_re3<=fft2\_re1-fft2\_re3;

147. ifft1\_im3<=fft2\_im1-fft2\_im3;

148. ifft2\_en<=1'b1;

149. end

150. else

151. ifft2\_en<=1'b0;

152. end

153.

154. reg signed [10:0] ifft2\_re0;

155. reg signed [10:0] ifft2\_im0;

156. reg signed [10:0] ifft2\_re1;

157. reg signed [10:0] ifft2\_im1;

158. reg signed [10:0] ifft2\_re2;

159. reg signed [10:0] ifft2\_im2;

160. reg signed [10:0] ifft2\_re3;

161. reg signed [10:0] ifft2\_im3;

162. always @(posedge clk or negedge rst\_n)begin

163. if(!rst\_n)begin

164. ifft2\_re0<=1'b0;

165. ifft2\_im0<=1'b0;

166. ifft2\_re1<=1'b0;

167. ifft2\_im1<=1'b0;

168. ifft2\_re2<=1'b0;

169. ifft2\_im2<=1'b0;

170. ifft2\_re3<=1'b0;

171. ifft2\_im3<=1'b0;

172. end

173. else if(ifft2\_en)begin

174. ifft2\_re0<=ifft1\_re0+ifft1\_re2;

175. ifft2\_im0<=ifft1\_im0+ifft1\_im2;

176. ifft2\_re1<=(ifft1\_re1+ifft1\_im3);

177. ifft2\_im1<=(-ifft1\_re3+ifft1\_im1);

178. ifft2\_re2<=ifft1\_re0-ifft1\_re2;

179. ifft2\_im2<=ifft1\_im0-ifft1\_im2;

180. ifft2\_re3<=(ifft1\_re1-ifft1\_im3);

181. ifft2\_im3<=(ifft1\_im1+ifft1\_re3);

182. data\_out\_en<=1'b1;

183. end

184. else

185. data\_out\_en<=1'b0;

186. end

187. //赋值

188. always @(posedge clk or negedge rst\_n)begin

189. if(!rst\_n)begin

190. fft\_data\_out\_re0<=1'b0;

191. fft\_data\_out\_im0<=1'b0;

192. fft\_data\_out\_re1<=1'b0;

193. fft\_data\_out\_im1<=1'b0;

194. fft\_data\_out\_re2<=1'b0;

195. fft\_data\_out\_im2<=1'b0;

196. fft\_data\_out\_re3<=1'b0;

197. fft\_data\_out\_im3<=1'b0;

198. end

199. else if(data\_out\_en)begin

200. fft\_data\_out\_re0<=fft2\_re0;

201. fft\_data\_out\_im0<=fft2\_im0;

202. fft\_data\_out\_re1<=fft2\_re1;

203. fft\_data\_out\_im1<=fft2\_im1;

204. fft\_data\_out\_re2<=fft2\_re2;

205. fft\_data\_out\_im2<=fft2\_im2;

206. fft\_data\_out\_re3<=fft2\_re3;

207. fft\_data\_out\_im3<=fft2\_im3;

208.

209. ifft\_data\_out\_re0<=ifft2\_re0/4'd4;

210. ifft\_data\_out\_im0<=ifft2\_im0/4'd4;

211. ifft\_data\_out\_re1<=ifft2\_re1/4'd4;

212. ifft\_data\_out\_im1<=ifft2\_im1/4'd4;

213. ifft\_data\_out\_re2<=ifft2\_re2/4'd4;

214. ifft\_data\_out\_im2<=ifft2\_im2/4'd4;

215. ifft\_data\_out\_re3<=ifft2\_re3/4'd4;

216. ifft\_data\_out\_im3<=ifft2\_im3/4'd4;

217. end

218. else begin

219. fft\_data\_out\_re0<=fft\_data\_out\_re0;

220. fft\_data\_out\_im0<=fft\_data\_out\_im0;

221. fft\_data\_out\_re1<=fft\_data\_out\_re1;

222. fft\_data\_out\_im1<=fft\_data\_out\_im1;

223. fft\_data\_out\_re2<=fft\_data\_out\_re2;

224. fft\_data\_out\_im2<=fft\_data\_out\_im2;

225. fft\_data\_out\_re3<=fft\_data\_out\_re3;

226. fft\_data\_out\_im3<=fft\_data\_out\_im3;

227.

228.

229. ifft\_data\_out\_re0<=ifft\_data\_out\_re0;

230. ifft\_data\_out\_im0<=ifft\_data\_out\_im0;

231. ifft\_data\_out\_re1<=ifft\_data\_out\_re1;

232. ifft\_data\_out\_im1<=ifft\_data\_out\_im1;

233. ifft\_data\_out\_re2<=ifft\_data\_out\_re2;

234. ifft\_data\_out\_im2<=ifft\_data\_out\_im2;

235. ifft\_data\_out\_re3<=ifft\_data\_out\_re3;

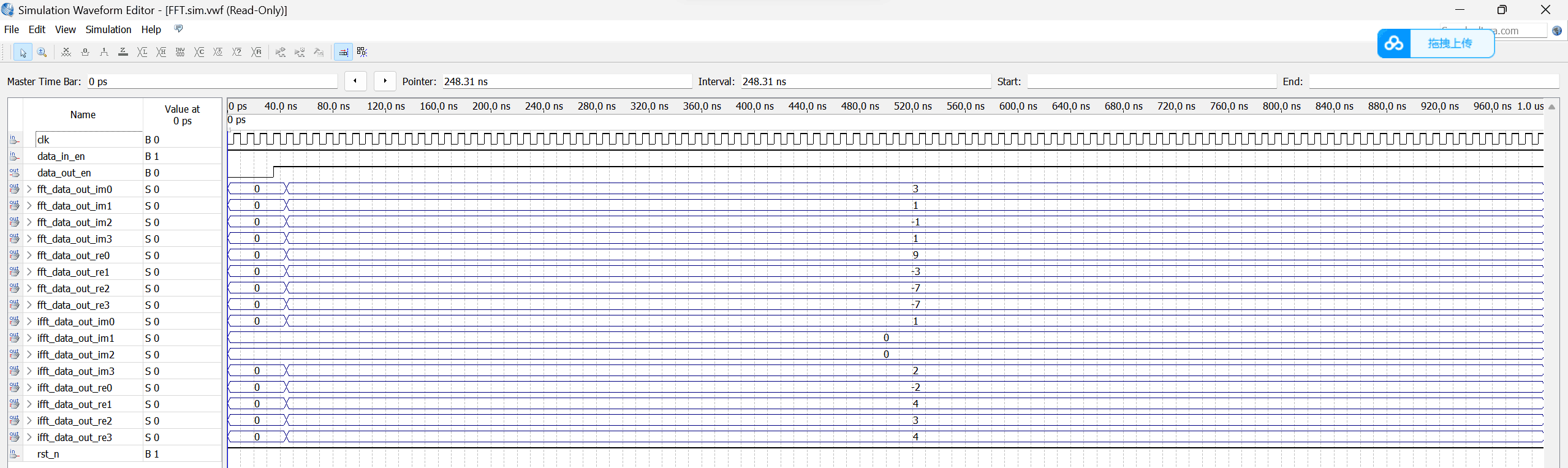
236. ifft\_data\_out\_im3<=ifft\_data\_out\_im3;

237. end

238. end

239. endmodule

FPGA仿真结果：



Matlab代码：

1. clc;

2. clear;

3. x=[-2+1i 4+2i 3 4];

4. XE=fft(x,4)

5. Xi=ifft(XE)

Matlab计算结果：

1. XE =

2. 9.0000 + 3.0000i -3.0000 + 1.0000i -7.0000 - 1.0000i -7.0000 + 1.0000i

3. Xi =

4. -2.0000 + 1.0000i 4.0000 + 2.0000i 3.0000 + 0.0000i 4.0000 + 0.0000i

**思考题：**

比较FFT运算与IFFT运算的异同点

相同点：都使用了蝶形运算，都用作数字信号处理。

不同点：IFFT是FFT的反运算，需要将FFT的旋转因子阶数取相反数，再将结果乘以1/N得到。

**实验4 IIR滤波器的设计**

1. **实验目的**

理解数字滤波器的基本概念，掌握IIR滤波器设计方法，掌握IIR滤波器性能分析方法。

**二. 实验内容**

假定信号的采样频率是600 Hz，输入信号是频率为100 Hz，180 Hz和250 Hz的合成正弦波信号f(t)=sin(200πt)+sin(360πt)+sin(500πt)，

（1）若要保留100 Hz信号，如何设计滤波器？用巴特沃兹滤波器实现；画出滤波器的频谱图及滤波前后信号的频谱图。

1. clc;

2. clear;

3. fs=600;%采样频率

4. f1=100;

5. f2=180;

6. f3=250;

7. N=1024;%100、180和250的最小公倍数是900，N取2的整数次幂

8. n=linspace(1,N,1024);

9. t=n/fs;

10. y=sin(200\*pi\*t)+sin(360\*pi\*t)+sin(500\*pi\*t);

11. Y=fft(y,N);

12. amplitude=abs(Y);

13. f=n\*fs/N;

14. subplot(3,1,1),plot(f,amplitude);

15. title('滤波前');xlabel('f');ylabel('幅值');

16. Wp=f1/(fs/2);

17. Ws=f2/(fs/2);

18. [nn,Wc]=buttord(Wp,Ws,1,100);%nn是巴特沃斯滤波器阶数

19. [B,A]=butter(nn,Wc);%传递函数的参数

20. z=filter(B,A,y);%对y进行滤波

21. Z=fft(z,1024);

22. amplitude=abs(Z);

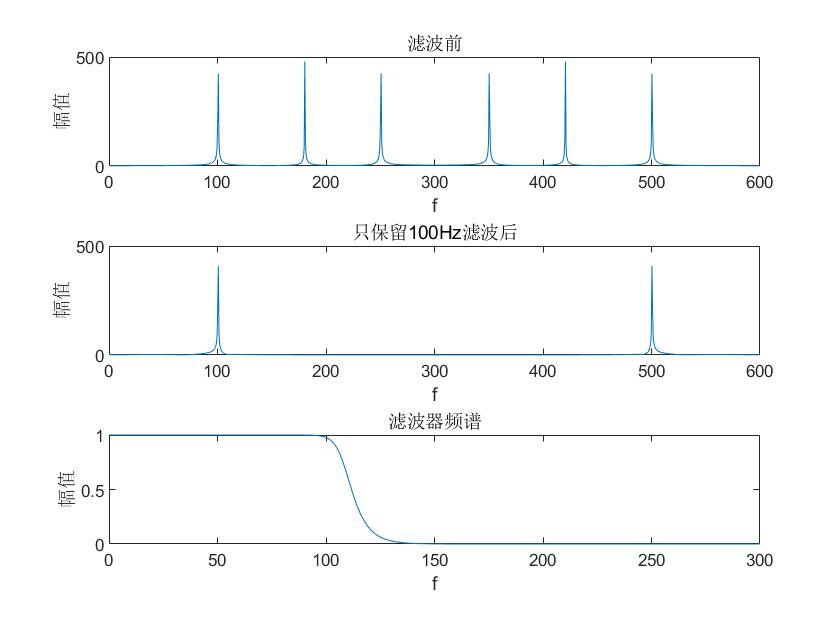
23. subplot(3,1,2),plot(f,amplitude);

24. title('只保留100Hz滤波后');xlabel('f');ylabel('幅值');

25. [h1,w1]=freqz(B,A,1024);

26. subplot(3,1,3),plot(w1/pi\*(fs/2),abs(h1));

27. title('滤波器频谱');xlabel('f');ylabel('幅值');



（2）若要保留250 Hz信号，如何设计滤波器？用巴特沃兹滤波器实现；画出滤波器的频谱图及滤波前后信号的频谱图。

1. clc;

2. clear;

3. fs=600;%采样频率

4. f1=100;

5. f2=180;

6. f3=250;

7. N=1024;%100、180和250的最小公倍数是900，N取2的整数次幂

8. n=linspace(1,N,1024);

9. t=n/fs;

10. y=sin(200\*pi\*t)+sin(360\*pi\*t)+sin(500\*pi\*t);

11. Y=fft(y,N);

12. amplitude=abs(Y);

13. f=n\*fs/N;

14. subplot(3,1,1),plot(f,amplitude);

15. title('滤波前');xlabel('f');ylabel('幅值');

16. Wp=f3/(fs/2);

17. Ws=f2/(fs/2);

18. [nn,Wc]=buttord(Wp,Ws,1,100);%nn是巴特沃斯滤波器阶数

19. [B,A]=butter(nn,Wc,'high');%传递函数的参数

20. z=filter(B,A,y);%对y进行滤波

21. Z=fft(z,1024);

22. amplitude=abs(Z);

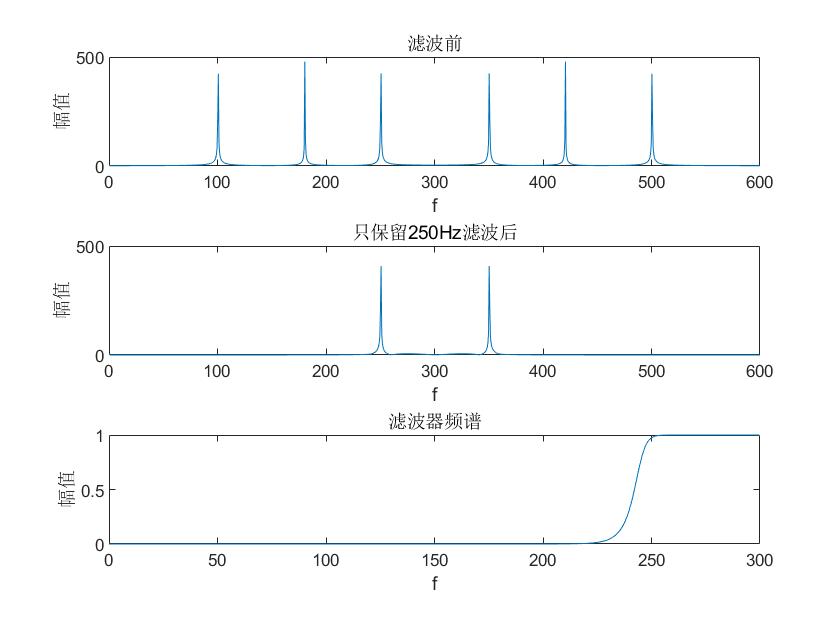
23. subplot(3,1,2),plot(f,amplitude);

24. title('只保留250Hz滤波后');xlabel('f');ylabel('幅值');

25. [h1,w1]=freqz(B,A,1024);

26. subplot(3,1,3),plot(w1/pi\*(fs/2),abs(h1));

27. title('滤波器频谱');xlabel('f');ylabel('幅值');



（3）若要保留100 Hz，250 Hz两个信号，如何设计滤波器？

用巴特沃兹滤波器实现；画出滤波器的频谱图及滤波前后信号的频谱图。

1. clc;

2. clear;

3. fs=600;%采样频率

4. f1=100;

5. f2=180;

6. f3=250;

7. N=1024;%100、180和250的最小公倍数是900，N取2的整数次幂

8. n=linspace(1,N,1024);

9. t=n/fs;

10. y=sin(200\*pi\*t)+sin(360\*pi\*t)+sin(500\*pi\*t);

11. Y=fft(y,N);

12. amplitude=abs(Y);

13. f=n\*fs/N;

14. subplot(3,1,1),plot(f,amplitude);

15. title('滤波前');xlabel('f');ylabel('幅值');

16. Wp=[(f1+10)/(fs/2) (f3-10)/(fs/2)];

17. Ws=[(f2-10)/(fs/2) (f2+10)/(fs/2)];

18. [nn,Wc]=buttord(Wp,Ws,1,100);%nn是巴特沃斯滤波器阶数

19. [B,A]=butter(nn,Wc,'stop');%传递函数的参数

20. z=filter(B,A,y);%对y进行滤波

21. Z=fft(z,1024);

22. amplitude=abs(Z);

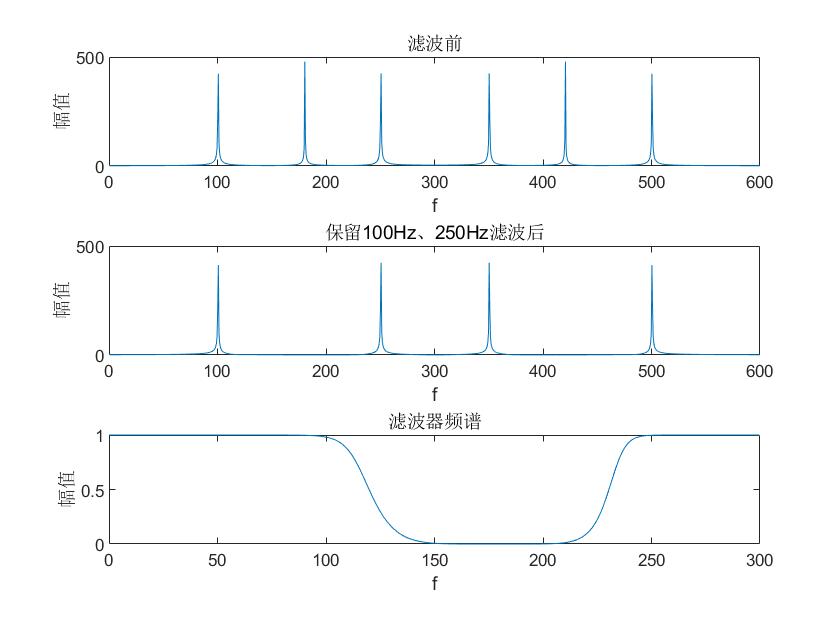
23. subplot(3,1,2),plot(f,amplitude);

24. title('不保留180Hz滤波后');xlabel('f');ylabel('幅值');

25. [h1,w1]=freqz(B,A,1024);

26. subplot(3,1,3),plot(w1/pi\*(fs/2),abs(h1));

27. title('滤波器频谱');xlabel('f');ylabel('幅值');



（4）若仅保留180 Hz的正弦信号如何设计滤波器？

用巴特沃兹滤波器实现；画出滤波器的频谱图及滤波前后信号的频谱图。

1. clc;

2. clear;

3. fs=600;%采样频率

4. f1=100;

5. f2=180;

6. f3=250;

7. N=1024;%100、180和250的最小公倍数是900，N取2的整数次幂

8. n=linspace(1,N,1024);

9. t=n/fs;

10. y=sin(200\*pi\*t)+sin(360\*pi\*t)+sin(500\*pi\*t);

11. Y=fft(y,N);

12. amplitude=abs(Y);

13. f=n\*fs/N;

14. subplot(3,1,1),plot(f,amplitude);

15. title('滤波前');xlabel('f');ylabel('幅值');

16. Wp=[(f1+10)/(fs/2) (f3-10)/(fs/2)];

17. Ws=[(f2-10)/(fs/2) (f2+10)/(fs/2)];

18. [nn,Wc]=buttord(Wp,Ws,1,100);%nn是巴特沃斯滤波器阶数

19. [B,A]=butter(nn,Wc);%传递函数的参数

20. z=filter(B,A,y);%对y进行滤波

21. Z=fft(z,1024);

22. amplitude=abs(Z);

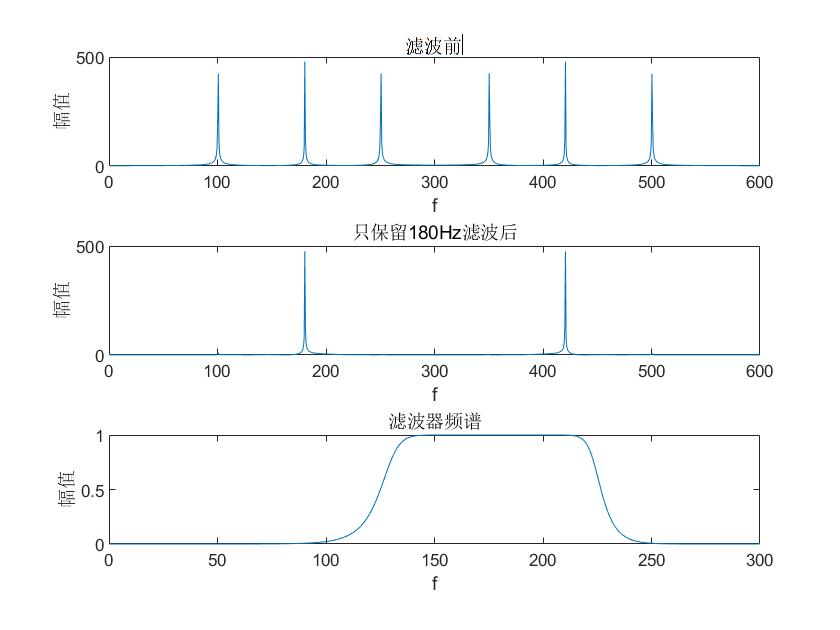
23. subplot(3,1,2),plot(f,amplitude);

24. title('只保留180Hz滤波后');xlabel('f');ylabel('幅值');

25. [h1,w1]=freqz(B,A,1024);

26. subplot(3,1,3),plot(w1/pi\*(fs/2),abs(h1));

27. title('滤波器频谱');xlabel('f');ylabel('幅值');



**思考题：**

1. 对巴特沃兹、切比雪夫、椭圆滤波器滤波器的滤波性能比较，简述各自的特点。

巴特沃兹滤波器通带最平坦，阻带下降慢。

切比雪夫滤波器通带等纹波，阻带下降较快。

椭圆滤波器比其他滤波器更陡，过渡带宽较窄，但是它在通带和阻带上都有波动。

1. 说明模拟频率、模拟角频率、数字角频率的含义及他们之间的关系

模拟频率是指每秒经历多少个周期，单位Hz，即1/s；模拟角频率是指每秒经历多少弧度，单位rad/s；数字角频率是指每个采样点间隔之间的弧度，单位rad。数字角频率与模拟角频率的关系是w=Ω\*Ts，其中w为数字角频率，Ω为模拟角频率，Ts为采样间隔。

1. 说明说明在MATLAB中下列函数中各参数的含义：

（1）[n,Wn]= buttord(Wp,Ws,Rp,Rs) ；

Wp 和Ws 分别是通带和阻带的截止频率，都是归一化频率,其取值范围为0 至1 之间；Rp 和Rs 分别是通带和阻带区的波纹系数。

（2）[b,a]=butter(n,Wn,'ftype') ；

N是阶数，Wn是3dB 截止频率的归一化值（关于pi归一化），

（3）y=filter(b,a,x)；

ftype 缺省时，Wc只有一个值时，默认为低通；Wc（[Wc1,Wc2]）为两个值，默认为带通。'ftype' = 'high' 高通滤波器，'ftype' = 'stop'带阻滤波器。

4. 说明高通、低通、带通、带阻滤波器对应的Wp和Ws值遵循的规则？

高通滤波器：Wp为截止频率，Ws为通带频率，Wp>Ws。

低通滤波器：Wp为截止频率，Ws为通带频率，Wp<Ws。

带通滤波器：Wp1和Wp2为截止频率，Ws1和Ws2为通带频率，Wp1<Ws1<Wp2<Ws2。

带阻滤波器：Wp1和Wp2为截止频率，Ws1和Ws2为通带频率，Ws1<Wp1<Ws2<Wp2。

**实验5 FIR滤波器设计**

1. **实验目的**

能根据FIR数字滤波器的原理设计滤波器对语音信号进行滤波。

**二. 实验内容**

对给定的语音信号进行加噪，并设计FIR滤波器滤除噪声，要求

（1）对给出的语音信号加噪声；

（2）在同一张图中，画出原始语音信号及加噪信号的时域波形；

（3）在同一张图中，画出原始语音信号及加噪信号的频域波形；

（4）设计FIR滤波器对加噪声的语音信号进行滤波，画出滤波器的频率响应图，及滤波后的时域及频域波形；并回放滤波后的信号，与加噪的语音信号及原始的语音信号进行对比，检验是否滤除了噪声。

1. clc;clear;close all;

2. fs=44100; %语音信号采样频率为22050

3. x1=audioread('我的祖国.wav'); %读取语音信号的数据，赋给变量x1

4. sound(x1,fs); %播放语音信号

5. y1=fft(x1,1024); %对信号做1024点FFT变换

6. f=fs\*(0:511)/1024;

7. figure(1);

8. plot(x1) %做原始语音信号的时域图形

9. title('原始语音信号');xlabel('time n');ylabel('fuzhi n');

10. figure(2);freqz(x1) %绘制原始语音信号的频率响应图

11. title('频率响应图')

12. figure(3);subplot(2,1,1);

13. plot(abs(y1(1:512))) %做原始语音信号的FFT频谱图

14. title('原始语音信号FFT频谱');

15. subplot(2,1,2);

16. plot(f,abs(y1(1:512)));

17. title('原始语音信号频谱')

18. xlabel('频率/Hz');ylabel('幅值');

19. X1=fft(x1);

20. L=length(X1);

21. k=0:L-1;

22. c1=0.1\*sin (2\*pi\*6000\*k/fs);

23. x2=x1+c1';

24. sound(x2,44100)

25. y2=fft(x2,1024);

26. figure(1);

27. plot(k,x2);

28. title('加噪后的信号');

29. xlabel('time n');

30. ylabel('fuzhi n');

31. figure(2);

32. subplot(2,1,1);

33. plot(f,abs(y1(1:512)));

34. title('原始语音信号频谱');

35. xlabel('频率/Hz');ylabel('幅值');hold on

36. subplot(2,1,2);

37. plot(f,abs(y2(1:512)));

38. title('加噪后的信号频谱');

39. xlabel('频率/Hz');ylabel('幅值');hold on

40.

41. wp=0.15\*pi;%实际的通带截止频率为(wp/pi)\*fs/2

42. ws=0.2\*pi;

43. wdelta=ws-wp;

44. wn=(ws+wp)/2;

45. N=ceil(6.2\*pi/wdelta);%选汉明窗

46. Window=hanning(N+1);

47. b=fir1(N,wn/pi,Window);

48. s=filter(b,1,x2);

49. SE=fft(s,1024);

50. figure;

51. plot(f,abs(SE(1:512)));

52. title('去噪后的信号频谱');

53. xlabel('频率/Hz');ylabel('幅值');

|  |  |
| --- | --- |
|  |  |
|  |  |

**思考题：**

1. IIR滤波器及FIR滤波器各自的特点？

IIR滤波器：IIR滤波器是一种递归滤波器，具有更高的计算效率和更窄的转换带宽，但可能会导致不稳定性和相位失真。

FIR滤波器：FIR滤波器是一种非递归滤波器，具有线性相位和稳定性，但需要更多的计算资源和更宽的转换带宽。

1. FIR滤波器线性相位的条件是什么？

如果FIRDF的单位抽样响应h(n)为实数，而且满足偶对称即h(n)=h(N-1-n)或奇对称h(n)=-h(N-1-n)，其对称中心在n=(N-1)/2处，可证明filter就具有严格的线性相位。

1. 说明窗函数的选择原则？主瓣和旁瓣对过渡带和波纹的影响？

主瓣应尽量窄，能量尽可能集中在主瓣内，从而获得高的频率分辨能力。

旁瓣衰减应尽量大，以减少频谱拖尾，但通常都不能同时满足这两个要求。

主瓣是时域信号频率成分的中央，旁瓣接近于0。旁瓣的高度显示了加窗函数对于主瓣周围频率的影响。对强正弦信号的旁瓣响应可能会超过对较近的弱正弦信号主瓣响应。一般而言，低旁瓣会减少FFT的泄漏，但是增加主瓣的带宽。

|  |  |
| --- | --- |
| **学生实验 心得** | 数字信号处理是一门非常精彩的学科，在无数的领域都有着非常重大的意义。通过此次实验，我领悟了很多滤波器的设计方法、一些数字信号处理的算法，并且被FFT的精妙所折服。在信号处理领域、自动控制领域、人工智能领域都有着数字信号处理的身影。  此次试验由我自己独立完成，由于理论水平有所欠缺，故有去询问老师与同学。感谢老师与各位同学的帮助，才有了我这次的实验结果。  学生（签名）：  2023年05月28日 |
| **指导**  **教师**  **评语** | 该同学理论基础 ，实验过程中，主动性  ， 独立完成实验内容， 相关的原理和方法，实验结果 ，对思考题的理解和回答 。  成绩评定：  指导教师（签名）：葛青  2023年 06月 日 |