****

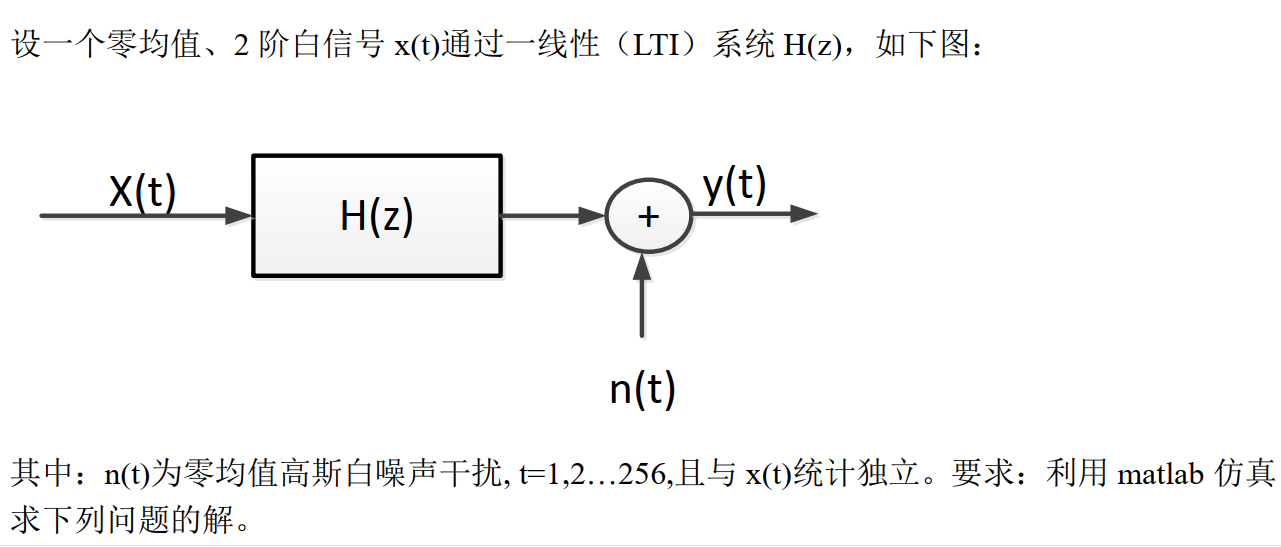
**随机信号大作业报告**

|  |  |
| --- | --- |
| 姓名 | 李鹏飞 |
| 学号 | 22309068 |
| 院系 | 电子与信息工程学院 |
| 专业 | 通信工程专业 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 2024 | 年 | 7 | 月 | 2 | 日 |

引言：本作业通过 MATLAB 仿真研究了随机信号通过线性时不变（LTI）系统的特性，重点考虑了高斯白噪声干扰对信号传输的影响。具体地，根据不同题设条件下的系统参数和相关量，计算并估计了输入信号 x(t)x(t)x(t) 和输出信号 y(t)y(t)y(t) 的自相关函数和互相关函数，从而获取系统的特性参数。最终，通过多种方法估计并比较了不同信噪比（SNR）条件下系统输出信号的功率谱密度，以便进行全面的横向比较。

题目：



题目给出的情形是一个零均值的二阶白信号x(t)通过线性系统H(z)后，与一零均值且与x(t)统计独立的高斯白噪声n(t)相加后得到输出信号y(t)

**第1题：**

1. 已知，要求计算信噪比SNR=10dB，20dB时的x(t)和y(t)的自相关函数和互相关函数

算法分析：首先利用randn函数来生成白噪声，利用filter函数进行二阶滤波后得到二阶白信号x(t)

系统函数，由此得到系统函数中

再用filter函数生成x(t)经过滤波器H(z)后的信号 y\_clean（无噪声），与n(t)相加后得到输出信号y(t)

计算x(t)和y(t)的自相关函数时使用直接估计法即

观察这个式子，不难发现，求和都是序列的两项相乘，且两项相隔距离固定，因此可以用matlab中的对两个序列进行点乘来简化计算，互相关函数计算也是同理，只需把上面式子中的一项x换成y即可。

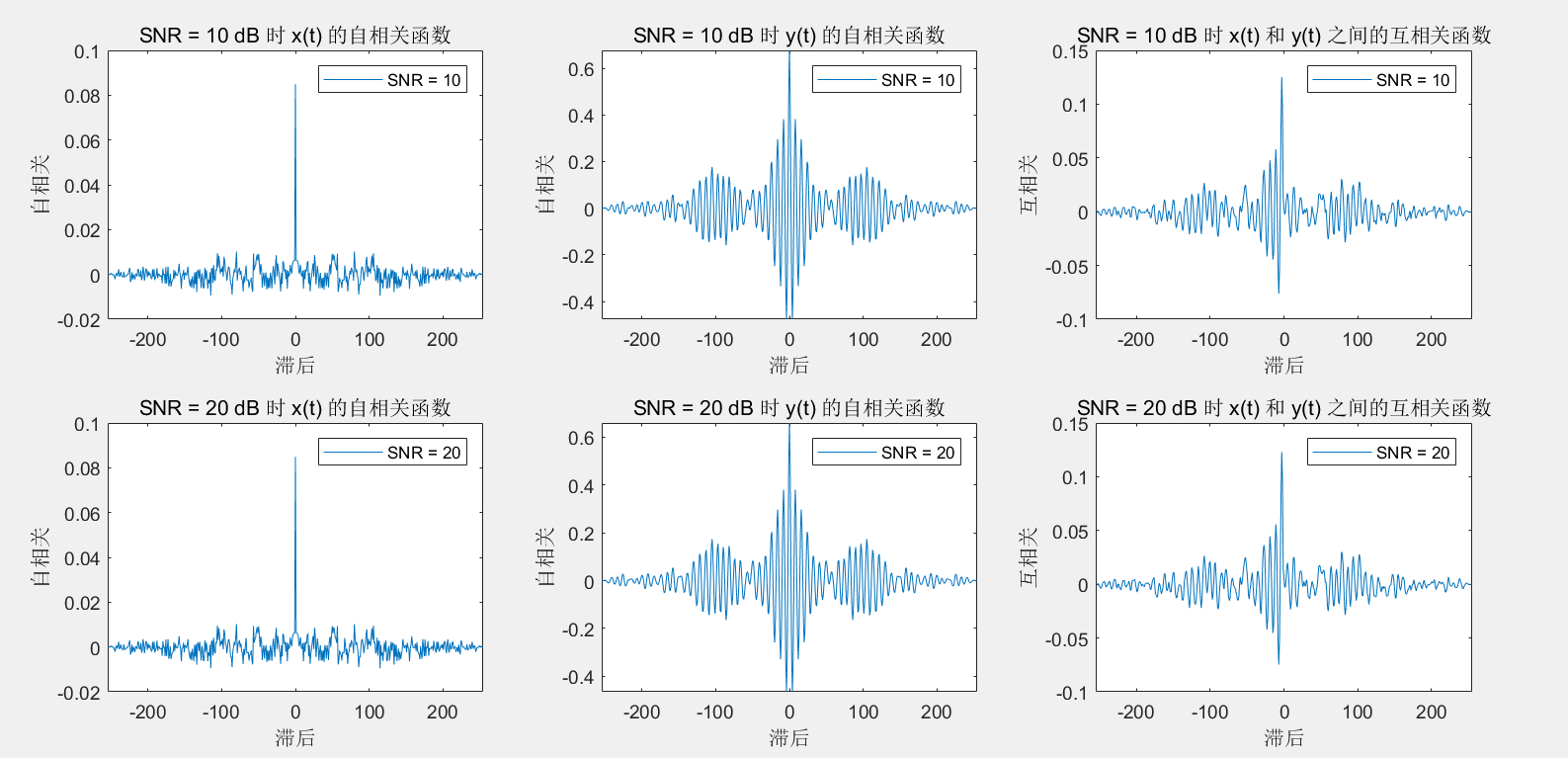
最终得到的matlab代码如下。

1. % 定义参数
2. N = 256; % 样本数
3. SNRs = [10, 20]; % 信噪比（以dB为单位）
4. % 定义传递函数 H(z)
5. z = tf('z', 1); % 定义 z 作为 z 域中的变量
6. H = 1 / ((z - (0.65 + 0.65i)) \* (z - (0.65 - 0.65i))); % 传递函数 H(z)
7. % 生成二阶白信号 x(t)
8. x = rand(N, 1) - 0.5; % 生成[-0.5, 0.5]均匀分布的随机数，然后减去均值0.5
9. % 初始化结果数组
10. autocorr\_x = cell(length(SNRs), 1);
11. autocorr\_y = cell(length(SNRs), 1);
12. crosscorr\_xy = cell(length(SNRs), 1);
13. % 遍历每个 SNR 值
14. for i = 1:length(SNRs)
15. % 计算给定 SNR 的噪声方差
16. snr = SNRs(i);
17. noise\_power = var(x) / (10^(snr/10));
18. n = sqrt(noise\_power) \* randn(N, 1); % 具有零均值的高斯白噪声
20. % 信号通过 H(z) 滤波
21. y = lsim(H, x, 0:(N-1)) + n; % 使用 lsim 进行时域仿真
23. % 计算 x(t) 的自相关函数
24. autocorr\_x{i} = my\_xcorr(x, x);
26. % 计算 y(t) 的自相关函数
27. autocorr\_y{i} = my\_xcorr(y, y);
29. % 计算 x(t) 和 y(t) 之间的互相关函数
30. crosscorr\_xy{i} = my\_xcorr(x, y);
31. end
32. % 绘制结果
33. figure;
34. for i = 1:length(SNRs)
35. lags = -(N-1):(N-1);
37. % 绘制 x(t) 的自相关函数
38. subplot(length(SNRs), 3, (i-1)\*3 + 1);
39. plot(lags, autocorr\_x{i});
40. title(['SNR = ', num2str(SNRs(i)), ' dB 时 x(t) 的自相关函数']);
41. xlabel('滞后');
42. ylabel('自相关');
43. legend(['SNR = ', num2str(SNRs(i))]); % Add legend
44. % 绘制 y(t) 的自相关函数
45. subplot(length(SNRs), 3, (i-1)\*3 + 2);
46. plot(lags, autocorr\_y{i});
47. title(['SNR = ', num2str(SNRs(i)), ' dB 时 y(t) 的自相关函数']);
48. xlabel('滞后');
49. ylabel('自相关');
50. legend(['SNR = ', num2str(SNRs(i))]); % Add legend
51. % 绘制 x(t) 和 y(t) 之间的互相关函数
52. subplot(length(SNRs), 3, (i-1)\*3 + 3);
53. plot(lags, crosscorr\_xy{i});
54. title(['SNR = ', num2str(SNRs(i)), ' dB 时 x(t) 和 y(t) 之间的互相关函数']);
55. xlabel('滞后');
56. ylabel('互相关');
57. legend(['SNR = ', num2str(SNRs(i))]); % Add legend
58. end
59. % 保存图形
60. saveas(gcf, ['correlations\_SNR' num2str(SNRs(1)) '\_SNR' num2str(SNRs(2)) '.png']);
61. % 定义自相关和互相关函数的计算函数
62. function r = my\_xcorr(x, y)
63. N = length(x);
64. r = zeros(2\*N-1, 1);
65. for lag = -(N-1):(N-1)
66. idx = lag + N;
67. if lag < 0
68. r(idx) = sum(x(1:end+lag) .\* y(1-lag:end)) / N;
69. else
70. r(idx) = sum(x(1+lag:end) .\* y(1:end-lag)) / N;
71. end
72. end
73. end

MATLAB代码的主要思想是生成一个零均值的二阶白信号，通过一个线性时不变系统（LTI）滤波，然后在不同信噪比（SNR）条件下添加噪声，最后计算和绘制信号的自相关函数和互相关函数。接下来我将介绍我的核心思想。

1. **定义参数和传递函数**：
   * N = 256：样本数。
   * SNRs = [10, 20]：信噪比，以dB为单位。
   * 定义传递函数H(z)，这里使用了两个复数极点。
2. **生成二阶白信号**：
   * x = rand(N, 1) - 0.5：生成一个长度为N的均匀分布在[-0.5, 0.5]的随机数序列，并通过减去0.5来确保信号的均值为零。
3. **初始化结果数组**：
   * 初始化存储自相关和互相关结果的数组autocorr\_x、autocorr\_y和crosscorr\_xy。
4. **遍历每个SNR值**：
   * **计算噪声方差**：
     + noise\_power = var(x) / (10^(snr/10))：根据给定的SNR计算噪声功率。
     + n = sqrt(noise\_power) \* randn(N, 1)：生成具有零均值和计算出的噪声方差的高斯白噪声。
   * **信号通过LTI系统滤波**：
     + y = lsim(H, x, 0:(N-1)) + n：使用lsim函数进行时域仿真，将信号x通过传递函数H(z)滤波，并添加噪声n。
   * **计算自相关和互相关函数**：
     + autocorr\_x{i} = xcorr(x, 'biased')：计算输入信号x的自相关函数。
     + autocorr\_y{i} = xcorr(y, 'biased')：计算输出信号y的自相关函数。
     + crosscorr\_xy{i} = xcorr(x, y, 'biased')：计算输入信号x和输出信号y之间的互相关函数。
5. **绘制结果**：
   * 定义用于绘图的滞后向量lags。
   * 使用subplot函数将每个SNR值的结果绘制在不同的子图中。
   * 使用plot函数绘制自相关和互相关函数的曲线。
   * 为每个子图添加标题、坐标标签和图例。
6. **遍历每个SNR值**：
   * 自相关和互相关的计算通过遍历所有可能的滞后（lag）来实现。对于每个滞后，计算信号之间的乘积和，然后归一化得到相关值。这个函数在脚本的末尾定义。

生成的MATLAB图像如下图所示：



系统函数，根据在信号与系统中学到的知识可知，系统函数没有零点，而两个极点0.65+0.65i和0.65-0.65i都在虚轴右侧，所以这个系统应该是一个低通系统，因此可以观察到x(t)经过该系统后自相关函数被展宽。

**第2题：**

2、已知x(t)，要求利用互相关函数来估计不同信噪比下H(z)的系数，并计算MSE（均方误差）

算法推导：系统输入与输出的互相关函数Rxy(t)与系统冲激响应h(t)存在如下关系

h(t)实际是一个序列，对应H(z)的各个参数，要从Rxy(t)和Rx(t)计算出h(t)，其实就是一个反卷积的过程，这时就要利用到matlab中的toeplitz函数。

序列与序列之间的卷积运算可以等效成一个矩阵与另一个序列相乘，toeplitz函数就是用来求出这个对应的矩阵的，利用这个函数求出Rx(t)对应的矩阵，再用Rxy(t)除以这个矩阵，就能够恢复出h(t)

另外，本题中并不需要计算完整的自相关函数和互相关函数，系统冲激响应序列长度为5，自相关函数和互相关函数都只需要计算出t从0到4的5个值就够了，可以大大简化计算。

得到估计值后，再根据MSE的公式计算各个系数的均方误差。

（鉴于在上一题已经演示过直接估计法估计自相关函数和互相关函数，本题中直接调用matlab中的xcorr函数来简化代码）

代码如下

1. % 定义参数
2. N = 256; % 样本数
3. SNRs = [0, 10, 20, 30]; % 信噪比（以dB为单位）
4. num\_experiments = 100; % 独立实验次数
5. % 定义实际传递函数 H(z) 的系数
6. h\_true = [1, 0.75, 0.8, 0.5, 0.3];
7. % 初始化存储MSE的数组
8. MSE = zeros(length(SNRs), 1);
9. % 生成二阶白信号 x(t)
10. x = rand(N, 1) - 0.5; % 生成[-0.5, 0.5]均匀分布的随机数，然后减去均值0.5
11. for snr\_idx = 1:length(SNRs)
12. snr = SNRs(snr\_idx);
13. mse\_sum = 0;
14. for k = 1:num\_experiments
15. % 计算噪声方差
16. noise\_power = var(x) / (10^(snr/10));
17. n = sqrt(noise\_power) \* randn(N, 1); % 具有零均值的高斯白噪声
19. % 通过实际传递函数 H(z) 滤波
20. y = filter(h\_true, 1, x) + n;
21. % 使用互相关估计 H(z) 的系数
22. rxy = xcorr(x, y, 'biased');
23. rxx = xcorr(x, 'biased');
25. % 计算 Rxx 矩阵
26. Rxx = toeplitz(rxx(N:N+4));
28. % 估计 H(z) 的系数
29. h\_est = Rxx \ rxy(N:N+4);
31. % 计算每个系数的相对误差平方和
32. mse\_sum = mse\_sum + sum((abs(h\_true' - h\_est) .^ 2) ./ abs(h\_true') .^ 2);
33. end
34. % 计算平均MSE
35. MSE(snr\_idx) = mse\_sum / num\_experiments;
36. end
37. % 打印MSE结果
38. fprintf('SNR (dB) | MSE\n');
39. fprintf('-----------------\n');
40. for i = 1:length(SNRs)
41. fprintf('%6d | %.6f\n', SNRs(i), MSE(i));
42. end
43. % 绘制MSE随SNR变化的图
44. figure;
45. plot(SNRs, MSE, '-o');
46. title('MSE vs SNR');
47. xlabel('SNR (dB)');
48. ylabel('MSE');
49. grid on;

代码的核心思想是通过模拟实验来估计信号传递函数 H(z)的系数，并在不同信噪比（SNR）条件下计算估计误差（均方误差，MSE）。代码具体包括以下几个步骤：

1. **定义参数**：

* N = 256：样本数。
* SNRs = [0, 10, 20, 30]：信噪比，以dB为单位。
* num\_experiments = 100：独立实验次数。

2. **定义实际传递函数 H(z)H(z)H(z) 的系数**：

* h\_true = [1, 0.75, 0.8, 0.5, 0.3]：实际传递函数的系数。

3. **生成二阶白信号**：

* x = rand(N, 1) - 0.5：生成长度为N的均匀分布在[-0.5, 0.5]的随机数序列，并通过减去0.5来确保信号的均值为零。

4. **遍历每个SNR值并进行独立实验**：

* **计算噪声方差**：
  + noise\_power = var(x) / (10^(snr/10))：根据给定的SNR计算噪声功率。
  + n = sqrt(noise\_power) \* randn(N, 1)：生成具有零均值和计算出的噪声方差的高斯白噪声。
* **通过实际传递函数滤波并添加噪声**：
  + y = filter(h\_true, 1, x) + n：将信号x通过传递函数滤波，并添加噪声n。
* **使用互相关估计传递函数的系数**：
  + rxy = xcorr(x, y, 'biased')：计算输入信号x和输出信号y之间的互相关函数。
  + rxx = xcorr(x, 'biased')：计算输入信号x的自相关函数。
  + Rxx = toeplitz(rxx(N:N+4))：构造自相关函数的Toeplitz矩阵。
  + h\_est = Rxx \ rxy(N:N+4)：使用最小二乘法估计传递函数的系数。
* **计算每个系数的相对误差平方和并累加**：
  + mse\_sum = mse\_sum + sum((abs(h\_true' - h\_est) .^ 2) ./ abs(h\_true') .^ 2)。

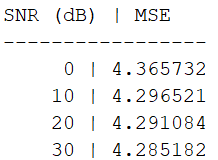
5. **计算平均MSE**：

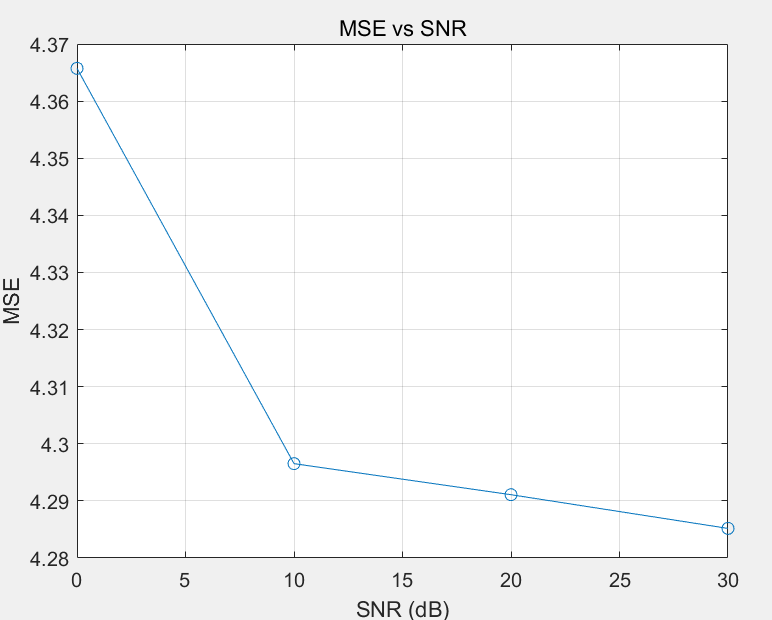
* MSE(snr\_idx) = mse\_sum / num\_experiments。

6. **打印和绘制MSE结果**：

* fprintf：打印每个SNR值的平均MSE。
* plot：绘制MSE随SNR变化的图。

运行代码得到结果如下：





可以观察到，随信噪比增大，估计得到的误差越来越小，推测可能是序列越往后在卷积中相乘次数就越少，导致误差增大。

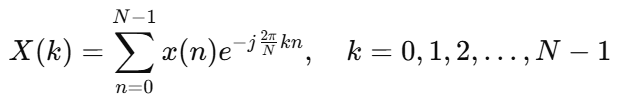
**第3题：**

3、H(z)与第1题相同，利用三种方法求x(t)未知时，不同信噪比下y(t)的功率谱密度。功率谱密度估计是信号处理和频谱分析中常用的方法之一，用于分析信号的频率成分和能量分布。下面详细介绍三种常见的功率谱密度估计方法：周期图法（Periodogram method）、自相关函数法（Bartlett方法）、以及现代功率谱估计方法（Burg方法）。

① **周期图法（Periodogram method）**

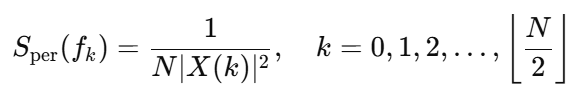
周期图法是最简单直接的功率谱密度估计方法之一，其基本思想是通过傅里叶变换直接计算信号的功率谱密度。具体步骤如下：

* **步骤**：
  1. **获取信号**：假设有一个长度为N的离散时间序列信号 {x(n)}，其中n=0,1,2,…,N−1。
  2. **计算周期图**：首先对信号进行零均值化（如果需要），然后计算其离散傅里叶变换（DFT）：



其中，X(k) 是信号的频域表示。

* 1. **计算功率谱密度**：通过信号的功率谱定义，可以得到频率 ​，其中 fs是信号的采样率。



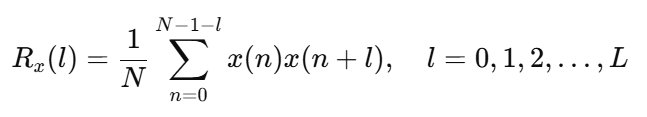
Sper​(fk​)表示在频率 fk处的功率谱密度估计值。

* **优缺点**：
  1. **优点**：简单直观，易于理解和实现。
  2. **缺点**：对信号长度和窗口选择敏感，较大的频率分辨率可能需要更长的信号长度。

② **自相关函数法（Bartlett方法）**

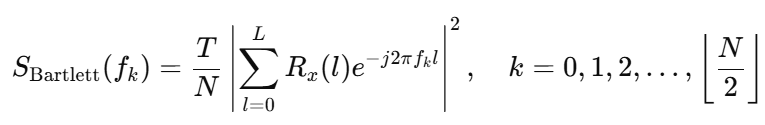
自相关函数法通过计算信号的自相关函数来间接估计功率谱密度，然后再通过周期图法进行估计。具体步骤如下：

* **步骤**：
  1. **获取信号**：假设有一个长度为N的离散时间序列信号 {x(n)}，其中n=0,1,2,…,N−1。
  2. **计算自相关函数**：信号的自相关函数 Rx​(l) 定义为：



其中 L是自相关函数的滞后阶数。

* 1. **计算周期图**：对自相关函数Rx(l) 进行傅里叶变换：



其中 TTT 是信号的总时长。

* **优缺点**：
  1. **优点**：通过自相关函数可以在一定程度上减小估计中的噪声。
  2. **缺点**：需要估计自相关函数并且需要选择适当的滞后阶数 LLL，处理复杂度较高。

③ **现代功率谱估计方法（Burg方法）**

现代功率谱估计方法采用自回归（AR）模型来描述信号，并基于Burg方法来估计其功率谱密度。具体步骤如下：

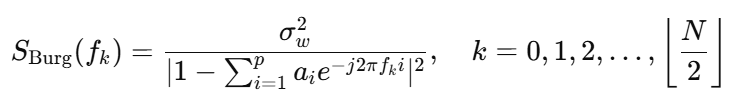
* **步骤**：
  1. **建立AR模型**：假设信号可以用 p 阶自回归（AR）模型表示：

文本, 信件

描述已自动生成

{w(n)}是白噪声序列， {ai​} 是模型系数。

* 1. **估计模型参数**：通过最小均方误差（MLE）或类似的方法估计AR模型的系数 {ai​}。
  2. **计算功率谱密度**：基于估计的AR模型系数，可以得到频域上的功率谱密度估计：



其中 ​ 是白噪声的方差。

* **优缺点**：
  1. **优点**：适用于非平稳信号，具有较高的频率分辨率。
  2. **缺点**：需要选择合适的AR模型阶数 ppp，并且计算量较大。

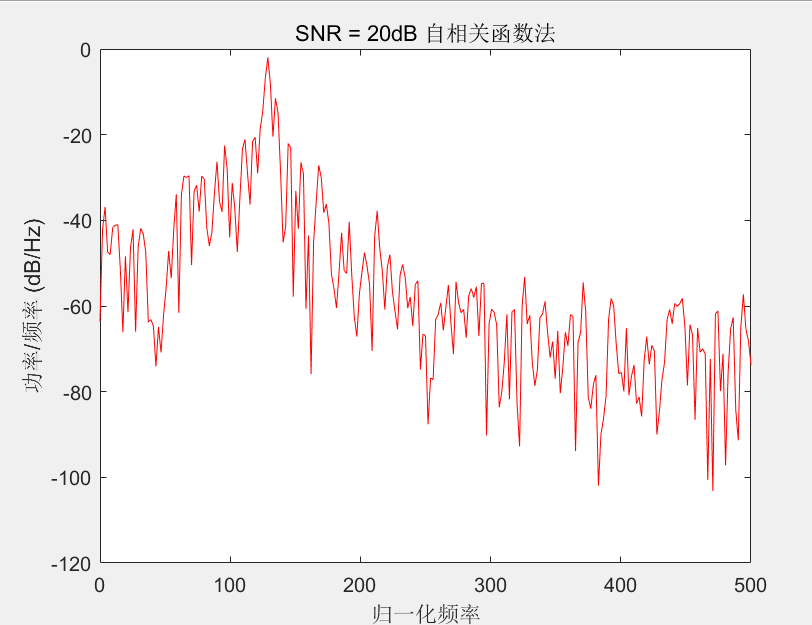
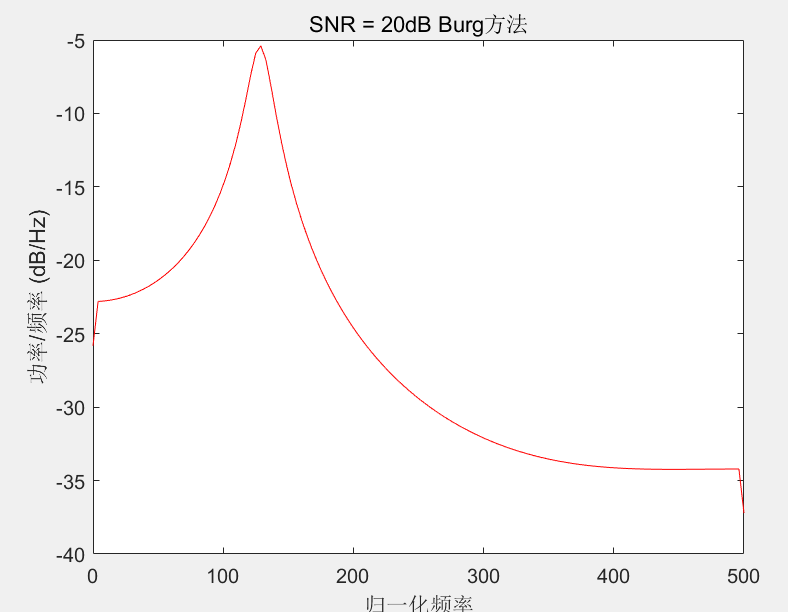
代码如下

1. % 参数定义
2. N = 256; % 信号长度
3. SNR1 = 10; % 信噪比1（以dB为单位）
4. SNR2 = 20; % 信噪比2（以dB为单位）
5. Fs = 1000; % 采样频率
6. % 定义H(z)传递函数
7. b = [0, 0, 1]; % 传递函数分子系数
8. a = [1, -1.3, 0.845]; % 传递函数分母系数
9. H = filter(b, a, 1); % 生成传递函数
10. % 生成高斯白噪声n(t)
11. n = randn(1, N); % 高斯白噪声
12. % 生成输入信号x(t)
13. x = randn(1, N); % 高斯白噪声信号
14. % 将x(t)通过传递函数H(z)进行滤波，得到滤波信号x\_filt
15. x\_filt = filter(b, a, x); % 滤波后的信号
16. % 计算给定信噪比情况下的噪声方差
17. signal\_power = var(x\_filt); % 信号功率
18. noise\_pow1 = signal\_power / 10^(SNR1 / 10); % 噪声功率1
19. noise\_pow2 = signal\_power / 10^(SNR2 / 10); % 噪声功率2
20. % 生成含噪声的输出信号y(t)
21. y1 = x\_filt + sqrt(noise\_pow1) \* n; % SNR = 10dB时的含噪声信号
22. y2 = x\_filt + sqrt(noise\_pow2) \* n; % SNR = 20dB时的含噪声信号
23. %% 周期图法（Periodogram method）
24. [py1, f] = periodogram(y1, [], [], Fs); % 周期图法功率谱估计 SNR = 10dB
25. [py2, ~] = periodogram(y2, [], [], Fs); % 周期图法功率谱估计 SNR = 20dB
26. %% 自相关函数法（Bartlett方法）
27. Ry1 = xcorr(y1, 'biased'); % 自相关函数 SNR = 10dB
28. Ry2 = xcorr(y2, 'biased'); % 自相关函数 SNR = 20dB
29. [Py1, f\_bartlett] = periodogram(Ry1, [], [], Fs); % 自相关函数法功率谱估计 SNR = 10dB
30. [Py2, ~] = periodogram(Ry2, [], [], Fs); % 自相关函数法功率谱估计 SNR = 20dB
31. %% 现代功率谱估计方法（Burg方法）
32. order = 4; % AR模型阶数
33. [arPy1, f\_burg] = pburg(y1, order, [], Fs); % Burg方法功率谱估计 SNR = 10dB
34. [arPy2, ~] = pburg(y2, order, [], Fs); % Burg方法功率谱估计 SNR = 20dB
35. %% 转换为dB/Hz
36. py1\_dB = 10 \* log10(py1); % 周期图法 SNR = 10dB
37. py2\_dB = 10 \* log10(py2); % 周期图法 SNR = 20dB
38. Py1\_dB = 10 \* log10(Py1); % 自相关函数法 SNR = 10dB
39. Py2\_dB = 10 \* log10(Py2); % 自相关函数法 SNR = 20dB
40. arPy1\_dB = 10 \* log10(arPy1); % Burg方法 SNR = 10dB
41. arPy2\_dB = 10 \* log10(arPy2); % Burg方法 SNR = 20dB
42. %% SNR = 10dB 时的功率谱密度图
43. figure;
44. plot(f, py1\_dB, 'r'); % 绘制周期图法的功率谱密度
45. title('SNR = 10dB 周期图法'); % 标题
46. xlabel('归一化频率'); % x轴标签
47. ylabel('功率/频率 (dB/Hz)'); % y轴标签
48. figure;
49. plot(f\_bartlett, Py1\_dB, 'r'); % 绘制自相关函数法的功率谱密度
50. title('SNR = 10dB 自相关函数法'); % 标题
51. xlabel('归一化频率'); % x轴标签
52. ylabel('功率/频率 (dB/Hz)'); % y轴标签
53. figure;
54. plot(f\_burg, arPy1\_dB, 'r'); % 绘制Burg方法的功率谱密度
55. title('SNR = 10dB Burg方法'); % 标题
56. xlabel('归一化频率'); % x轴标签
57. ylabel('功率/频率 (dB/Hz)'); % y轴标签
58. %% SNR = 20dB 时的功率谱密度图
59. figure;
60. plot(f, py2\_dB, 'r'); % 绘制周期图法的功率谱密度
61. title('SNR = 20dB 周期图法'); % 标题
62. xlabel('归一化频率'); % x轴标签
63. ylabel('功率/频率 (dB/Hz)'); % y轴标签
64. figure;
65. plot(f\_bartlett, Py2\_dB, 'r'); % 绘制自相关函数法的功率谱密度
66. title('SNR = 20dB 自相关函数法'); % 标题
67. xlabel('归一化频率'); % x轴标签
68. ylabel('功率/频率 (dB/Hz)'); % y轴标签
69. figure;
70. plot(f\_burg, arPy2\_dB, 'r'); % 绘制Burg方法的功率谱密度
71. title('SNR = 20dB Burg方法'); % 标题
72. xlabel('归一化频率'); % x轴标签
73. ylabel('功率/频率 (dB/Hz)'); % y轴标签

代码实现了在不同信噪比（SNR）条件下，通过三种方法（周期图法、自相关函数法和Burg方法）估计系统输出信号 y(t) 的功率谱密度（PSD），并绘制了相应的图形。核心步骤解析如下：

1. **参数定义**:
   * 信号长度 N=256N。
   * 信噪比（SNR）为10dB和20dB。
   * 采样频率 Fs=1000Fs = 1000 Hz。
2. **定义传递函数 H(z)H(z)H(z)**:
   * 分子系数 b=[0,0,1]。
   * 分母系数 a=[1,−1.3,0.845]。
3. **生成高斯白噪声和输入信号**:
   * x(t) 和噪声 n(t) 都是高斯白噪声信号。
4. **信号滤波**:
   * 将 x(t) 通过传递函数 H(z) 滤波，得到滤波后的信号 x\_filtx
5. **噪声叠加**:
   * 计算给定信噪比情况下的噪声方差，并生成含噪声的输出信号 y(t)。
   * y1 是 SNR = 10dB 时的含噪声信号。
   * y2 是 SNR = 20dB 时的含噪声信号。
6. **功率谱密度估计**:
   * **周期图法（Periodogram method）**: 直接计算信号的功率谱密度。
   * **自相关函数法（Bartlett方法）**: 通过计算自相关函数，然后用周期图法估计功率谱密度。
   * **现代功率谱估计方法（Burg方法）**: 使用Burg方法估计自回归（AR）模型的功率谱密度。
7. **功率谱密度转换为dB/Hz**:
   * 将估计的功率谱密度转换为dB/Hz。
8. **绘图**:
   * 绘制不同SNR下、不同方法的功率谱密度图。

运行代码得到输出结果如下



图表, 折线图, 直方图

描述已自动生成图表, 直方图

描述已自动生成 图表, 折线图

描述已自动生成 图表, 折线图, 直方图

描述已自动生成

前面已经提到这个系统是一个低通系统，因此输出信号y(t)的功率谱会主要集中在低频部分，仿真结果也体现出了这一点，且Welch法得到的功率谱相比另外两种方法的得到的功率谱更加集中。

总结：在这次大作业中，我主要专注于LTI系统的分析以及随机信号的自相关和互相关的仿真模拟。这些实践让我深刻体会到，在信号处理领域，随机过程和概率论的知识具有广泛的应用。通过功率谱的估计和多种方法对自相关函数的估计，我不仅理论上理解了这些概念，还学会了如何在实际中应用它们。

这次作业让我认识到，理论推导与实际应用之间存在一定的差距。理论是理解问题的基础，但在实际工程中，将这些理论转化为有效的算法并实现代码才是关键。这需要我们不仅仅掌握理论知识，还要具备将其应用到实际问题中的工程能力。因此，理论与实践相结合的学习方式使我对信号处理有了更加全面和深入的理解。

在这个过程中，我学会了如何运用学到的理论知识来解决实际的工程问题，这不仅提升了我的技术能力，也增强了我的解决问题的信心。通过大作业，我不仅获得了新的知识，还培养了分析问题和解决问题的能力，这对我的学术和职业发展都是非常宝贵的经验。