北京理工大學
Beijing Institute of Technology

本科生实验报告

课程名称：高级数字信号处理

实验名称：MUSIC 算法求功率谱

任课教师：	田黎育			实验教师：	田黎育
实验日期：	2025 年 12 月 1 日			实验地点：	
实验类型：	☑ 原理验证	☐ 综合设计		☐ 自主创新	
学生姓名：	闫子易	班级：	13212302	学号：	1120230621
学　　院：	集成电路与电子学院			专　　业：	电子科学与技术
组　　号：		同组同学：			
成　　绩：					

集成电路与电子学院
SCHOOL OF INTEGRATED CIRCUITS AND ELECTRONICS

一、实验目的

① 2 个复正弦信号，32 点长度，16 阶(p=15)+复噪声，特征值排大小

② 要有算的过程，也就是 MUSIC 谱公式每一行乘的过程

二、实验原理

MUSIC 算法相当于在非参数法的基础上进行了一系列延伸，用到了子空间的数学思想。对于一般的复正弦信号+白噪声的模型，其表达式为

$$x[n]=\sum_{i=1}^{M}A_i\,e^{j(\omega_i n+\varphi_i)}+u[n]=c[n]+u[n]$$

其中，M 为复正弦信号的个数，c[n]为信号，u[n]为白噪声。

易知 $\mathbf{c}[\mathrm{n}]$ 的自相关函数为

$$R_c[m]=E[c[n]\cdot c^*[n+m]]=\sum_{i=1}^{M}|A_i|^2e^{j\omega_i i}$$

$\mathbf{x}[\mathrm{n}]$ 的自相关函数为

$$R_x[m]=R_c[m]+\rho_u\delta[m]$$

其中 ρ_u 为噪声功率。由维纳-辛钦定理可知，对序列的自相关函数作傅里叶变换即可得到序列的功率谱，由此可得 $\mathrm{x[n]}$ 的功率谱为

$$S_x(\omega)=\sum_{i=1}^{M}2\,\pi|A_i|^2\delta(\omega-\omega_i)+\rho_u$$

用信号向量表示相关矩阵，即

令 $\vec{e_i}=[1,a_i,a_i^2,\dots,a_i^P]^T$ ，为 $(\mathrm{P}+1)\times 1$ 的矩阵，其中 $a_i=e^{j\omega_i}$

则序列的自相关矩阵为

$$R_P=S_P+W_P=\sum_{i=1}^{M}|A_i|^2\vec{e_i}\cdot\vec{e_i}^H+\rho_u\vec{I}$$

其中， S_P 为 $\mathbf{M}$ 个复正弦信号 $(\mathrm{P}+1)$ 阶自相关矩阵的累加， $\vec{I}$ 为单位阵。

令 $E=[\overrightarrow{e_1},\overrightarrow{e_2},\dots,\overrightarrow{e_M}]=\begin{bmatrix}1 & 1 & \dots & 1\\ a_1 & a_2 & \dots & a_M\\ \vdots & \vdots & \ddots & \vdots\\ a_1^P & a_2^P & \dots & a_M^P\end{bmatrix}$

则 $S_P=E\cdot\begin{bmatrix}|A_1|^2 & & & \\ & |A_2|^2 & & \\ & & \ddots & \\ & & & |A_M|^2\end{bmatrix}\cdot E^H$

由矩阵相乘，秩不增加，可知 $\mathrm{P}\geq\mathrm{M}$ 时， S_P 不满秩。而由于噪声存在，故 R_P 仍为满秩。下面对 S_P 和 R_P 进行特征分解。

由特征值和特征向量相关概念可知, $S_P\cdot\vec{v}=\lambda\cdot\vec{v}$ ，故可得

$$S_P=\sum_{i=1}^{P+1}\lambda_i\,\overrightarrow{v_i}\cdot\overrightarrow{v_i}^H$$

其中 λ_i 为特征值， $\overline{v_i}$ 为特征向量。

由于 S_P 的秩为 M ，故特征值中有 M 个值不为 0，其余 $\mathrm{P}+1-\mathrm{M}$ 个为 0，且

$$\lambda_1\geq\lambda_2\geq\cdots\geq\lambda_{P+1}$$

故 S_P 可化简为

$$S_P=\sum_{i=1}^{M}\lambda_i\,\overrightarrow{v_i}\cdot\overrightarrow{v_i}^H$$

又因为 $R_P=S_P+W_P$ ，则

$$R_P=\sum_{i=1}^{M}\lambda_i\,\overrightarrow{v_i}\cdot\overrightarrow{v_i}^H+\sum_{i=1}^{P+1}\rho_u\,\overrightarrow{v_i}\cdot\overrightarrow{v_i}^H=\sum_{i=1}^{M}(\lambda_i+\rho_u)\overrightarrow{v_i}\cdot\overrightarrow{v_i}^H+\sum_{i=M+1}^{P+1}\rho_u\,\overrightarrow{v_i}\cdot\overrightarrow{v_i}^H$$

由此可将 R_P 的特征向量分为两个子空间：

$\overrightarrow{v_1}$ ， $\overrightarrow{v_2}$ ，…， $\overrightarrow{v_M}$ 张成信号子空间，特征值为 $(\lambda_i+\rho_u)$ ， $i=1,2,\dots,M$

$\overrightarrow{v_{M+1}}$ ， $\overrightarrow{v_{M+2}}$ ，…， $\overrightarrow{v_{P+1}}$ 张成噪声子空间，特征值为 ρ_u

由线性代数相关理论可知，信号向量与噪声子空间的向量正交，即

$$k=(M+1),(M+2),\dots,(P+1)\text{时},\overrightarrow{e_i}^H\cdot\overrightarrow{v_k}=0$$

$$\text{则}\sum_{k=M+1}^{P+1}\overrightarrow{e_i^H}\cdot\overrightarrow{v_k}=0$$

由此可知，当频率 ω_{i} 为信号的真实频率时，对应的 $\overline{e_i}$ 就满足

$$\sum_{k=M+1}^{P+1} \left|\overrightarrow{e_i}^H \cdot \overrightarrow{v_k}\right|^2 = 0$$

而其他频率处，由于噪声的影响，通常会使上式累积一定的值。

由此可构建 MISIC 谱

$$\hat{P}_x(\omega) = \frac{1}{\sum_{k=M+1}^{P+1} \left|\overrightarrow{e_i}^H \cdot \overrightarrow{v_k}\right|^2}$$

频率 ω_i 为信号的真实频率时，该谱会出现较高的峰，而其他频率处，该谱的值较小。由此可抑制噪声成分，在噪声环境中估计出多个信号频率。

三、运行代码及实验内容

按照实验要求生成复信号和复噪声，将其封装在 MUSIC_gen 函数里

```
%产生含噪的复正弦波
function [dat]=MUSIC_gen(f,A,var,num)

%f  :含若干频率的向量
%A  :不同频率分量的振幅
%var:噪声方差

m=length(f);    %复正弦信号的个数

signal=zeros(1,num);
noise = sqrt(var/2)*(randn(1,num) + 1j*randn(1,num));  % 复高斯噪声，方差为 var

n=1:num;
for k=1:m
    signal=signal+A(k)*exp(j*2*pi*f(k)*n);  %生成信号成分
end

dat=signal+noise;   %生成包含噪声的序列
```

然后再实现 MUSIC 算法，先用 MUSIC_macf 根据输入数据 x 和阶数 p 构造自相关矩阵(R_{xx})，对其做特征分解得到特征值和特征向量，然后按特征值大小排序以便把能量较大的信号子空间和能量较小的噪声子空间区分开；通过 f 的长度确定信号数(m)，取排序后最小的(p-m)个特征向量作为噪声子空间基向量，对每个噪声特

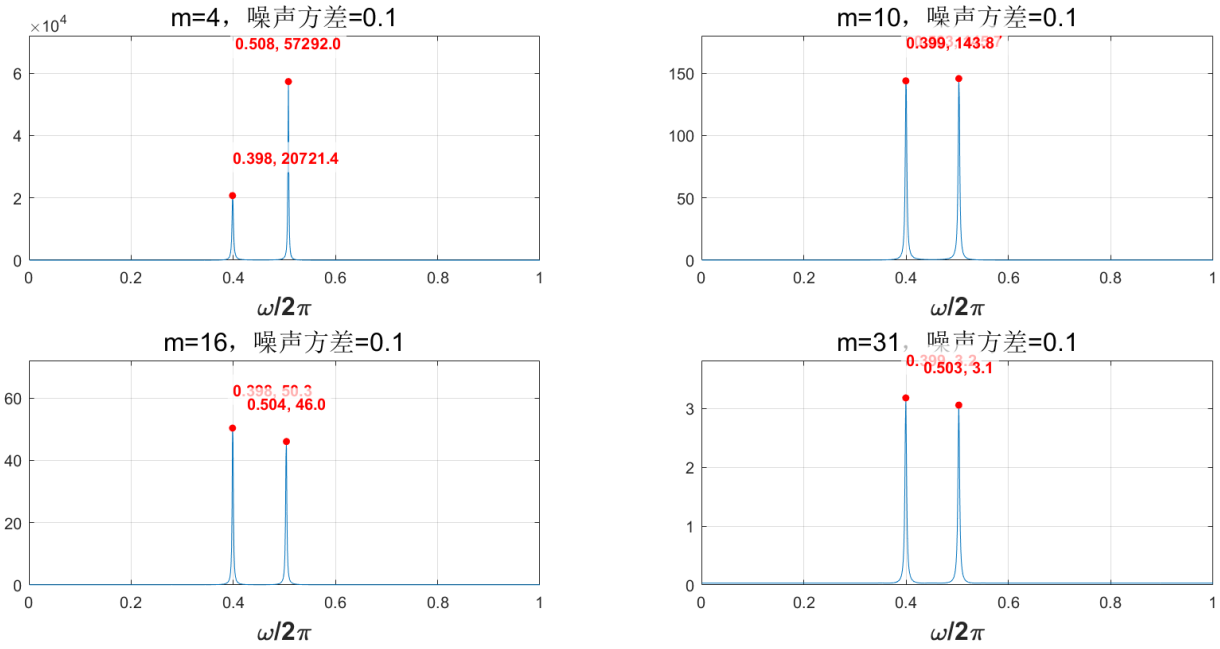
征向量做长度为 num 的 FFT，计算其频域幅值平方并累加得到 MUSIC 谱的分母项，最后对累加结果取倒数得到 MUSIC 谱估计（其中 p 要大于信号个数，num 控制频率分辨率）。

```
function [music] = MUSIC_pmusic(f, x, p, num)

    % 初始化谱向量
    temp = zeros(1, num);

    % Step 1: 构造自相关矩阵
    Rxx = MUSIC_macf(x, p);

    % Step 2: 特征分解
    [v, d] = eig(Rxx);

    % Step 3: 按特征值大小排序（升序）
    eigvals = diag(d);
    [~, idx] = sort(eigvals, 'ascend');

    v = v(:, idx);
    d = diag(eigvals(idx));

    % Step 4: 计算噪声子空间
    m = length(f);   % 信号个数
    for i = 1 : p - m
        V = fft(v(:, i), num);
        temp = temp + V .* conj(V);  % MUSIC 谱分母
    end

    % Step 5: 计算 MUSIC 谱
    music = 1 ./ temp;
end
```

首先，令序列长度 $N = 32$ ，信号频率为 $f = [0.4,0.5]$ ，信号幅值为 $A = [1,1,1]$ ，噪声方差为 $var = 0.1$ 。

我把阶数 m 从小到大情况下的 MUSIC 谱都画在了一张图上，方便比较观察。

N=32，信号个数=2

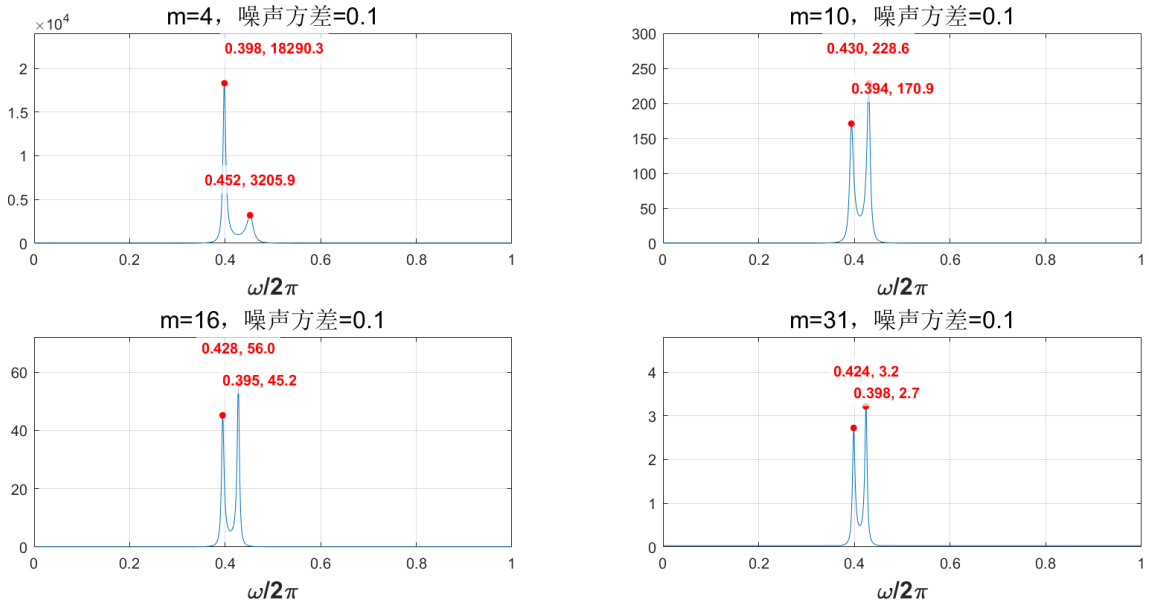


可以看到，在噪声方差为 0.1，N 为 32 点的小噪声环境下，MUSIC 算法在 m=4 时就已经能分辨出清晰的两个峰了，在 m 大于 10 时，两个峰已经非常尖锐且高度相当，可以轻松地分辨出频率 0.4 和 0.5。

（1）减少两个信号间的临近频率

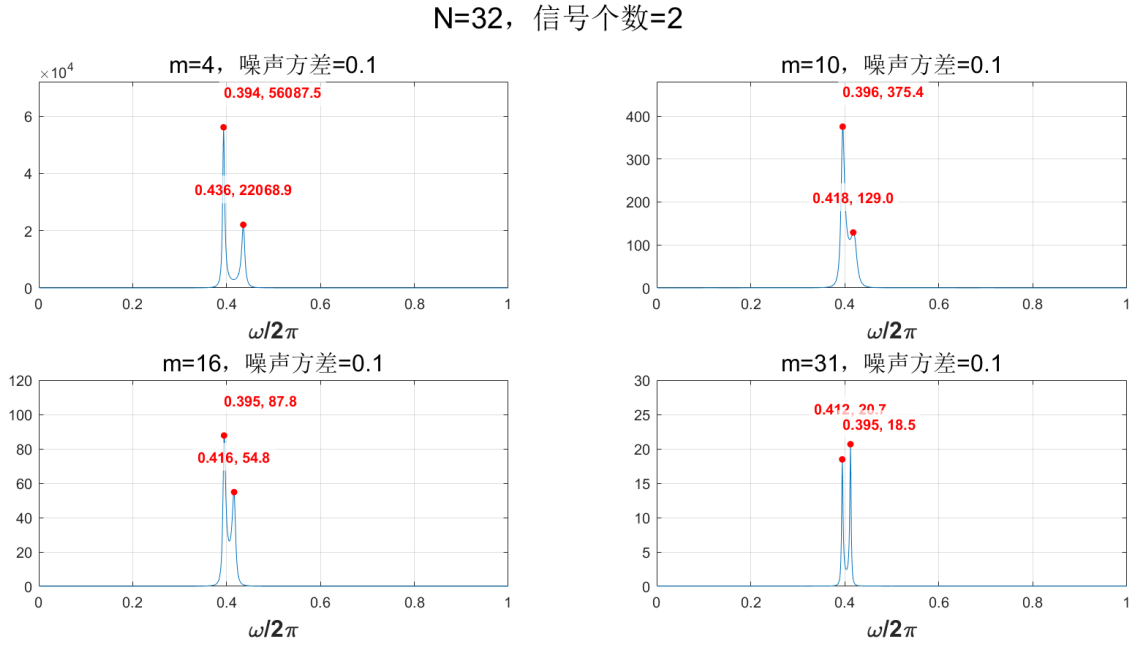
f1,f2 改为 0.4 和 0.42 我们接着将 f1,f2 改为 0.4 和 0.42，缩小频率差。

N=32，信号个数=2



可以看到，在 f1,f2 为 0.4 和 0.42 时，m=4 的阶数已经略显吃力，能有两个峰，但一大一小不清晰而且对 0.42 的频率估计偏差很大。但当阶数达到 16 时，依然能看见清晰的两个峰，且估计的误差较小。

接着缩小频率间隔为 0.4，0.405



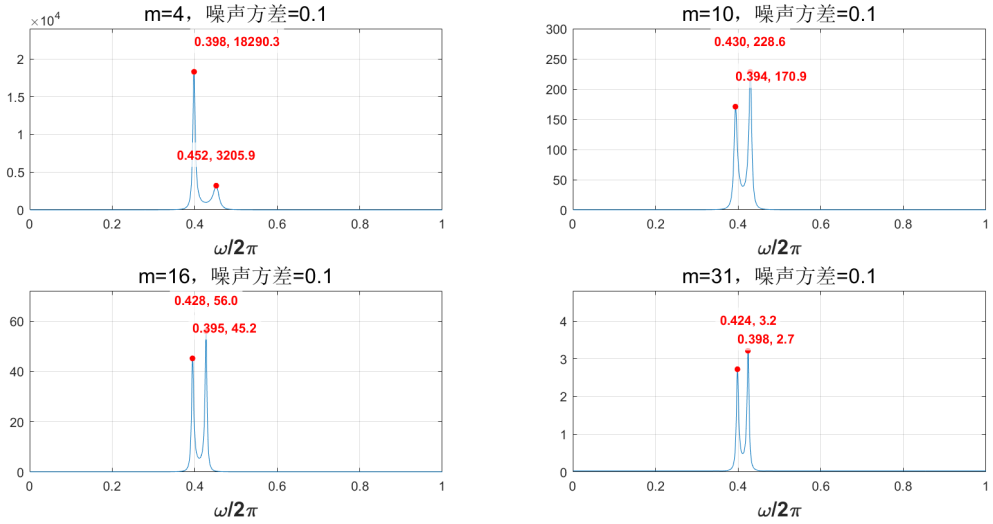
此时 m 为 10 以下已经无法分辨出两个频率的信号了，当 m=N/2=16 时，可以分辨出两个较高的谱峰，尽管预测结果略有偏差，说明 MUSIC 算法还是有很高的分辨率的。

（2）噪声对 MUSIC 谱估计的影响

现在，我们再把两个信号的频率重新设为 0.4 和 0.42，一个适中的值，再探究噪声对 MUSIC 谱估计的影响。

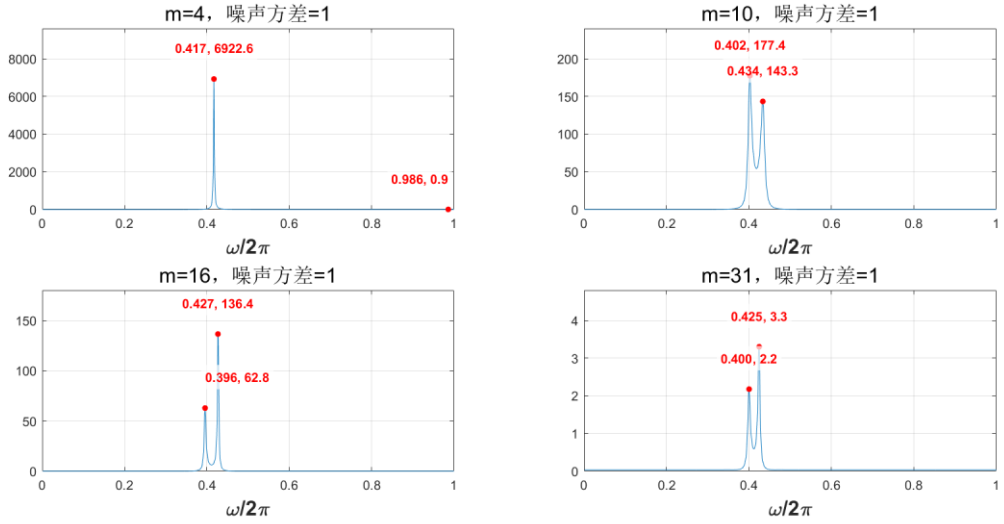
噪声方差为 0.1 时

N=32，信号个数=2



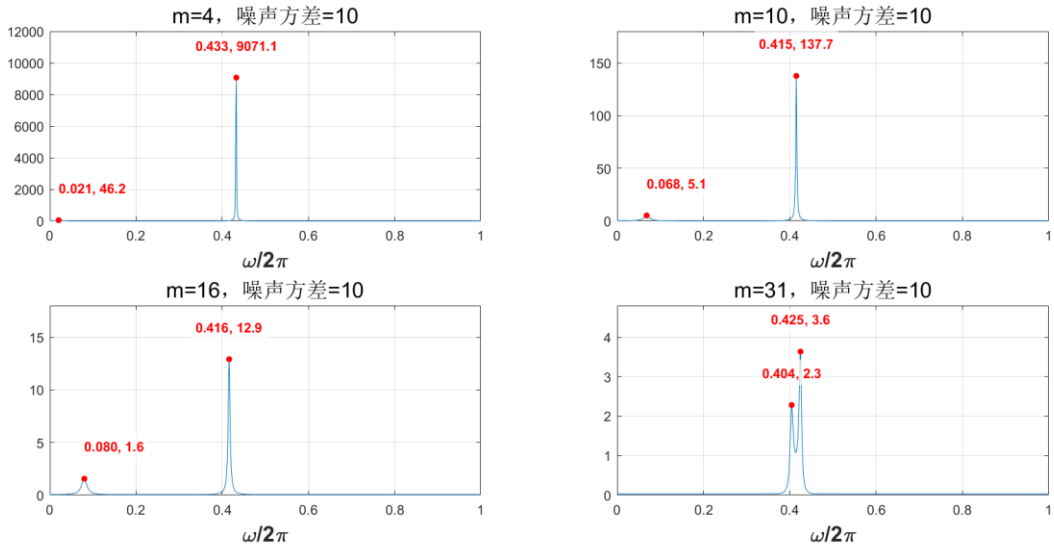
噪声方差为 1 时

N=32，信号个数=2



噪声方差为 10 时

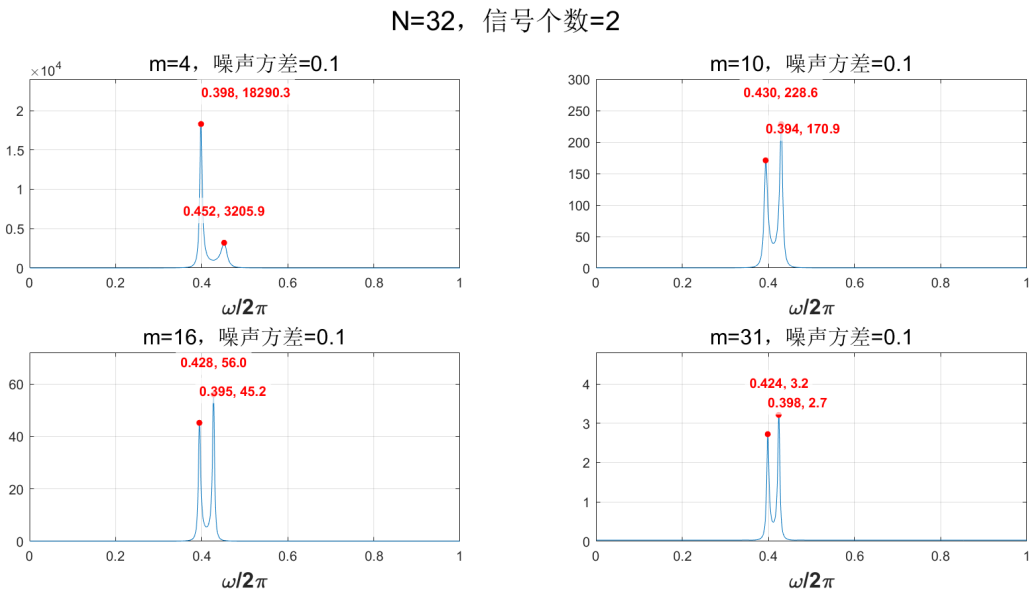
N=32，信号个数=2



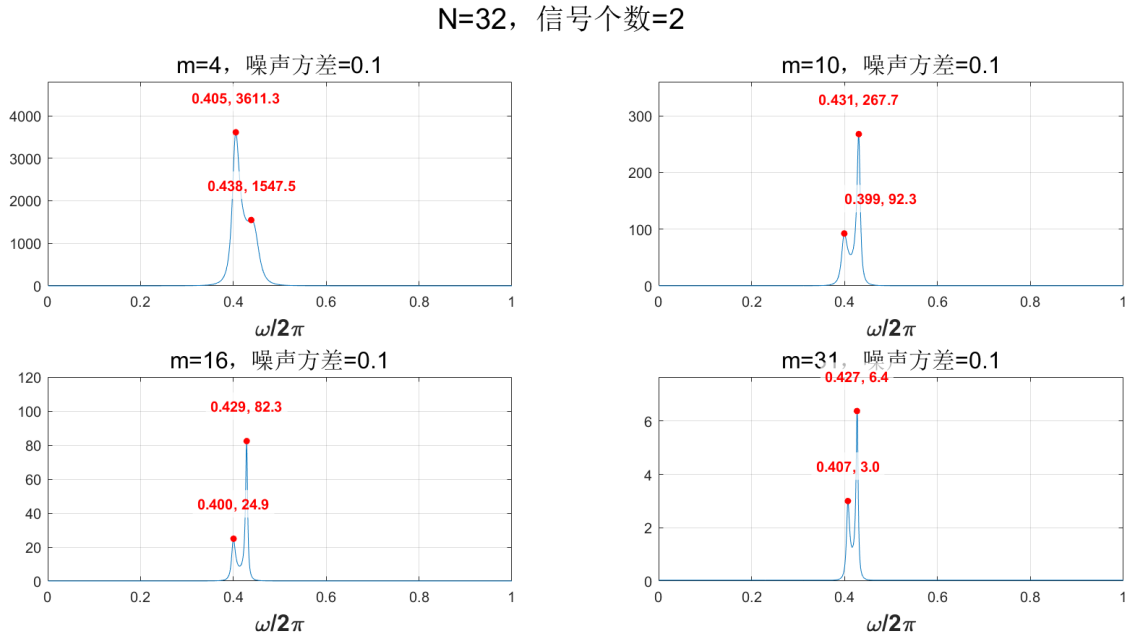
此时 m 在 16 以下时都无法分辨出两个峰（左边那个小峰明显是错误的），当 m 达到 31 时，才能分辨出两个信号的谱峰。可以说明噪声越大，MUSIC 谱估计效果越差，估计准确所需的阶数越高。

（3）两个信号的幅度比对 MUSIC 谱估计的影响

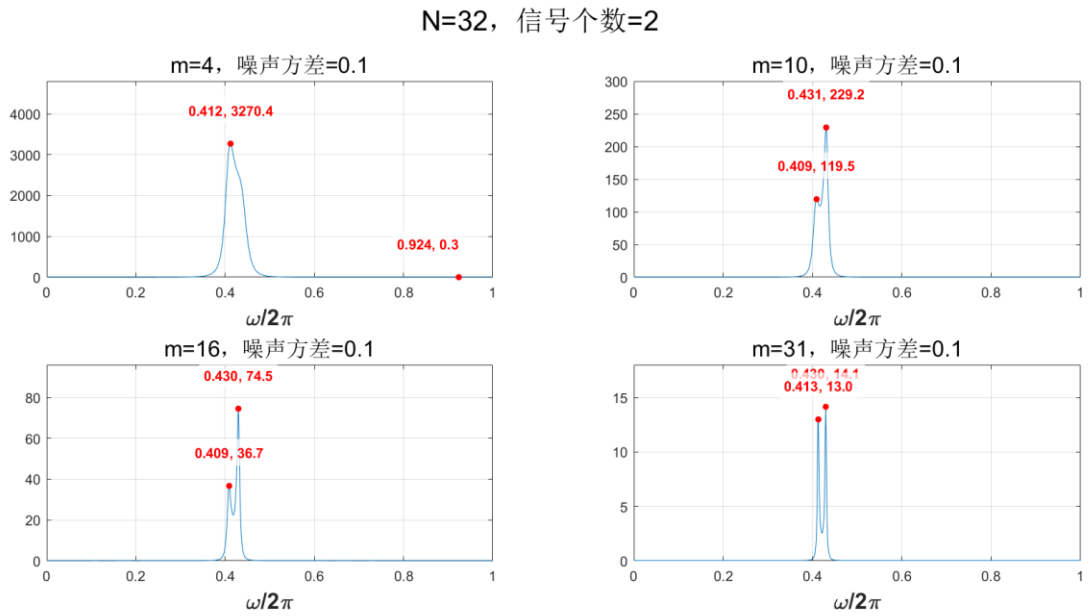
固定两个信号频率分别为 0.4 和 0.42，噪声方差为 0.1，先令 A1=A2=1



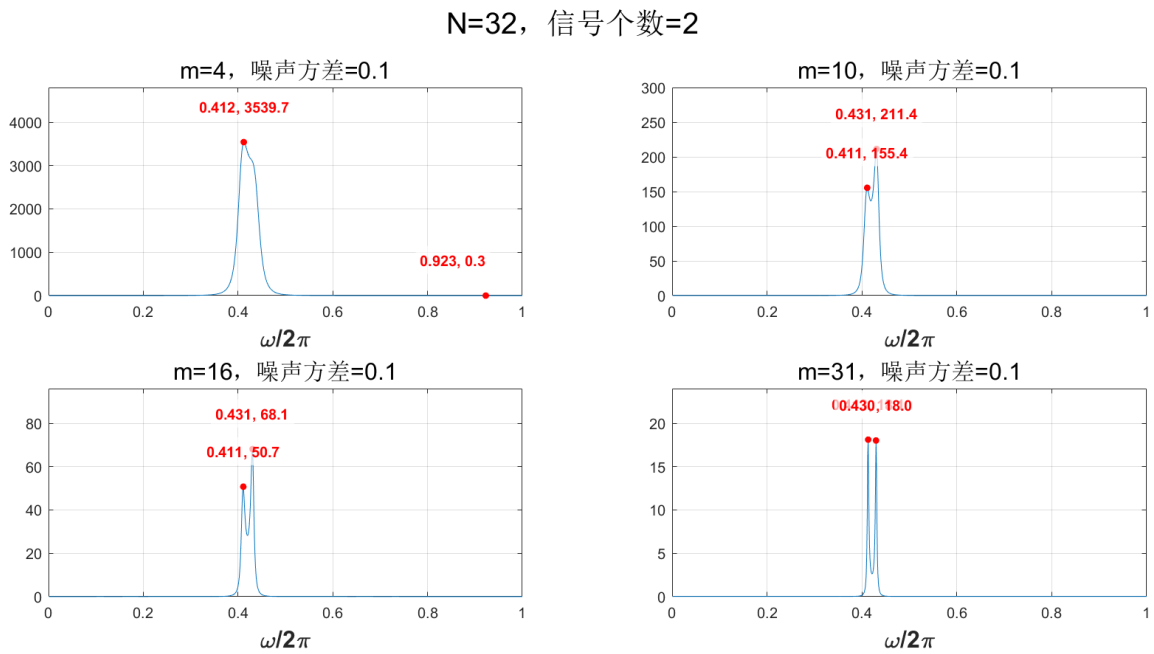
令 A1=1，A2=2



令 A1=1，A2=5



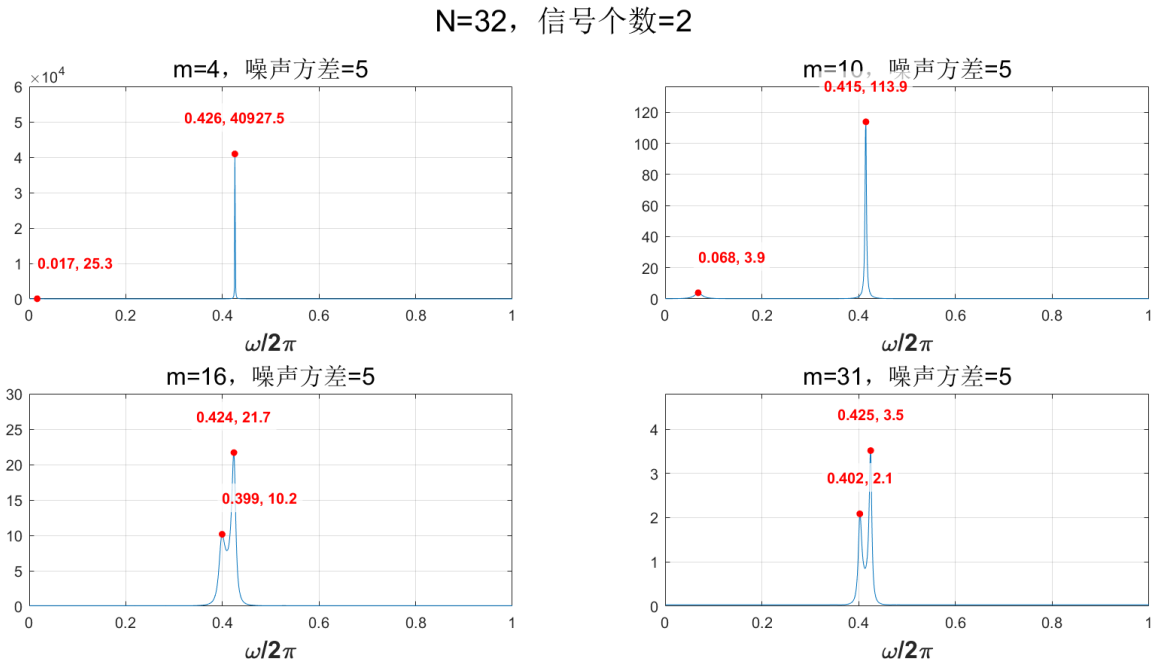
令 A1=1，A2=10



可以看到，随着第二个信号幅度的增加，在低阶时 MUSIC 算法已经无法区分开两个谱峰，但当 m 达到 16 时，即便两个信号的幅度比已经达到了 1：10，MUSIC 算法还是能区分出两条谱线，尽管预估的频率有些偏差，但这已经比上一次实验的 Levinson 递推法对“掩蔽效应”的的效果要好很多。

（4）阶数 m 对 MUSIC 谱效果的影响

固定两个信号频率分别为 0.4 和 0.42，噪声方差为 5，A1=A2=1，阶数 m 从 4 增加到 31。



可以明显看到，在同样的信号条件和噪声条件下，在阶数 m 较小时，MUSIC 算法无法分辨出两个谱峰，当 m 达到 16 时，两个谱峰逐渐出现，但还不明显，到 31 时，谱峰已经清晰尖锐，可以很好地估计两个信号的频率。

但为什么这里在 m 已经达到序列长度的大小时没有出现过拟合呢？（像前面实验的 AR 模型和 Levinson 算法那样）

原因就在 MUSIC 的子空间原理与正交性

协方差分解：对接收数据向量 $\mathbf{x}(n)$（阵列或时序模型化取决于你的实现）在平稳白噪声下有

$$\mathbf{R} = \mathbb{E}\{\mathbf{x}(n)\mathbf{x}^H(n)\} = \mathbf{A}\mathbf{S}\mathbf{A}^H + \sigma^2\mathbf{I},$$

其中 $\mathbf{A}$ 的列是各频率的导向向量（或者“方向向量”），$\mathbf{S}$ 是信号协方差，$\sigma^2\mathbf{I}$ 是白噪声协方差。

特征分解与子空间：令

$$\mathbf{R} = \mathbf{E}_s\mathbf{\Lambda}_s\mathbf{E}_s^H + \mathbf{E}_n\mathbf{\Lambda}_n\mathbf{E}_n^H,$$

其中 $\mathbf{E}_s$ 跨越信号子空间（维数为信号个数 K），$\mathbf{E}_n$ 跨越噪声子空间（维数为 $m-K$）。在理想白噪声下，$\mathbf{\Lambda}_n=\sigma^2\mathbf{I}$，且对真实频率 f_k 的导向向量 $\mathbf{a}(f_k)$ 满足正交性

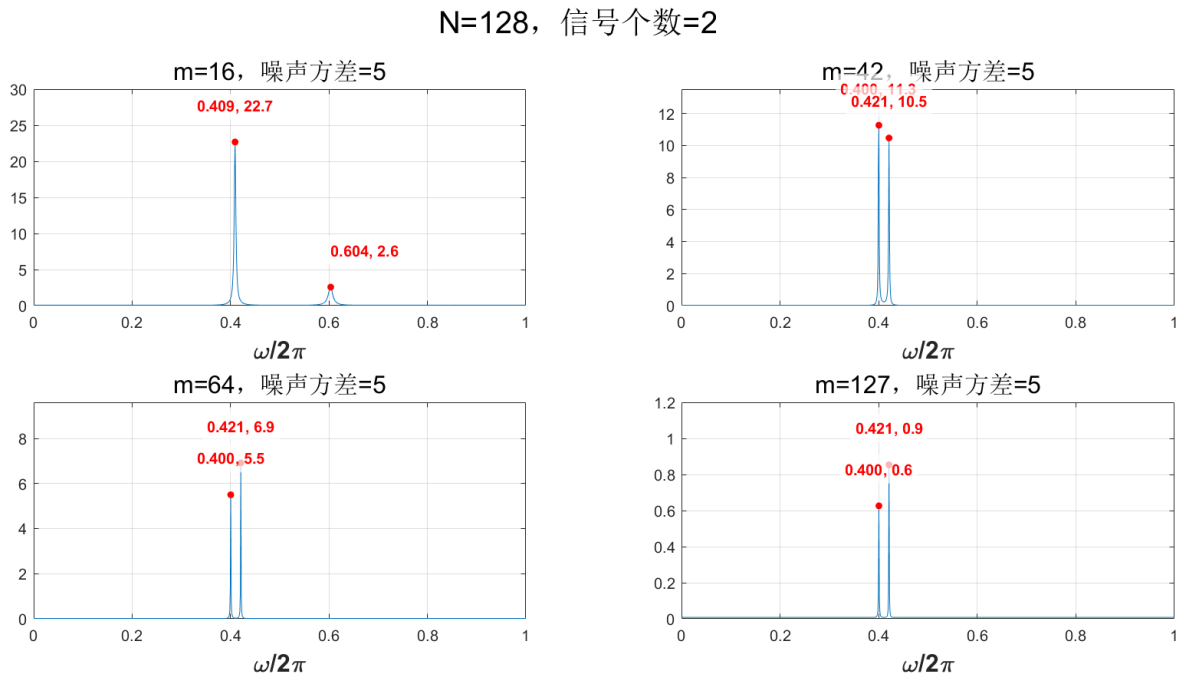
$$\mathbf{E}_n^H\mathbf{a}(f_k)=\mathbf{0}.$$

这条几何关系是 MUSIC 的根基，不依赖于“高阶多项式去拟合数据”。

当阶数（协方差矩阵的维度或子空间维度）增大到接近序列长度，所做的是提高在样本协方差上的特征分解精度与子空间分离的稳定性。只要$m>K$，噪声子空间仍然存在，正交关系仍可成立。增加m会让$\mathbf{E}_n\mathbf{E}_n^H$的估计更接近理想投影，使分母更接近零点，因而峰更“尖”。因为 MUSIC 的峰只在“导向向量落入信号子空间”时出现，受制于几何正交关系，所以它不会因增加参数自由度去“解释噪声”，不会产生一堆来源于噪声极点的伪峰。

（5）序列长度 N 对 MUSIC 谱效果的影响

固定两个信号频率分别为 0.4 和 0.42，噪声方差为 5，A1=A2=1，但序列长度 N 从 32 增加到现在的 128



可以看到，相同的信号、噪声下，增大序列长度 N 在同样的阶数[N/8, N/3, N/2, N-1]下所得到的谱峰明显更加尖锐清晰，并且预估的频率也更接近真实值，可以说明

序列越长，已知数据越多，能够用于估计的信息就越多，预估也就越精准。

（6）算的过程，也就是 MUSIC 谱公式每一行乘的过程

我把这一部分单独封装为了对输入参数进行检查，确保数据维度和变换点数合法，然后通过自相关矩阵和特征分解得到信号子空间和噪声子空间。接着在噪声子空间维度上逐步进行乘加运算：对每个噪声特征向量做 FFT 并计算模平方，再逐次累加到分母向量中，同时在命令行输出每一步的统计信息。最终分母向量的倒数就是 MUSIC 谱，代码将其与分母的范围和峰值一起打印出来，并把所有中间结果存入结构体以便后续可视化。可视化部分通过四个子图展示不同角度的过程：最终谱并标注两个最大峰值、关键频率点的累加曲线、累加过程的热力图以及最终分母的分布并标注两个最小值。为了避免标注重叠，在坐标上做了错位处理。

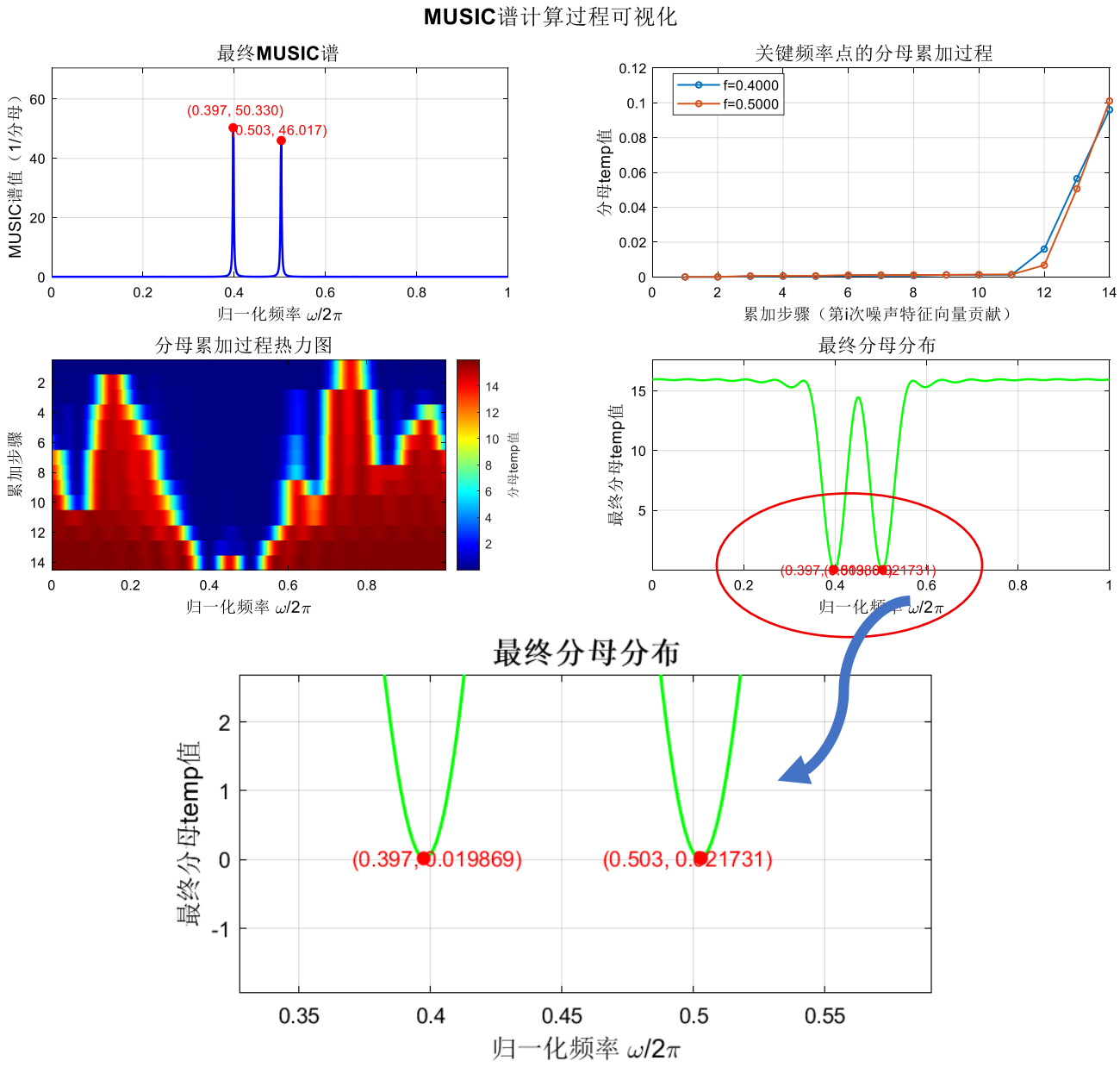
```
% ====================== 子图 1：最终 MUSIC 谱（标注 2 个极大值）
======================
    subplot(2, 2, 1);
    plot(freq, final_music, 'b-', 'LineWidth', 1.5);
    hold on;

    % 通用局部极大值检测（数学定义：某点大于左右相邻点）
    music_peaks_freq = [];
    music_peaks_val = [];
    for i = 2:length(final_music)-1
        if final_music(i) > final_music(i-1) && final_music(i) >
final_music(i+1)
            music_peaks_freq = [music_peaks_freq, freq(i)];
            music_peaks_val = [music_peaks_val, final_music(i)];
        end
    end

    % 选择幅值最大的 2 个极大值
    selected_peaks = [];
    if ~isempty(music_peaks_val)
        % 按幅值降序排序
        [sorted_vals, sorted_idx] = sort(music_peaks_val, 'descend');
        sorted_freqs = music_peaks_freq(sorted_idx);
```

```
        % 选前 2 个（不足 2 个则选所有）
        num_select = min(2, length(sorted_vals));
        selected_peaks = [sorted_freqs(1:num_select);
sorted_vals(1:num_select)];
    end

    % 错位标注坐标（避免重叠）
    vert_offset = [0.12, 0.08];  % 垂直偏移比例（上移）
    horz_offset = [0.005, -0.005]; % 水平偏移
    y_max = max(final_music);

    for k = 1:size(selected_peaks, 2)
        f_val = selected_peaks(1, k);
        music_val = selected_peaks(2, k);

        % 计算错位位置
        y_offset = music_val * vert_offset(k);
        f_val_offset = f_val + horz_offset(k);
        f_val_offset = max(min(f_val_offset, max(freq)), min(freq)); % 限制在频率
范围内

        % 绘制红点（填充）
        plot(f_val, music_val, 'ro', 'MarkerSize', 6, 'MarkerFaceColor', 'r');
        % 标注坐标（只显示数值）
        text(f_val_offset, music_val + y_offset, sprintf('(%.3f, %.3f)', f_val,
music_val), ...
             'FontSize', 10, 'Color', 'r', 'HorizontalAlignment', 'center');
    end

    xlabel('归一化频率 \omega/2\pi', 'FontSize', 12);
    ylabel('MUSIC 谱值（1/分母）', 'FontSize', 12);
    title('最终 MUSIC 谱', 'FontSize', 14, 'FontWeight', 'bold');
    grid on;
    ylim([0, 1.4*y_max]); % 扩大 y 轴范围，确保标注完全显示

    % ====================== 子图 2：关键频率点的累加过程（保持不变）
======================
    subplot(2, 2, 2);
    % 选择 3 个关键频率点（真实信号频率+中间频率）
    selected_freqs = [signal_freq];
    selected_freqs = unique(selected_freqs);  % 去重
    colors = lines(length(selected_freqs));
```

```
    for k = 1:length(selected_freqs)
        f_val = selected_freqs(k);
        [~, idx] = min(abs(freq - f_val));  % 找到频率对应的列索引
        accum_vals = temp_intermediate(:, idx);  % 矩阵第 idx 列=所有累加步骤的分母
值
        plot(1:noise_dim, accum_vals, 'o-', 'Color', colors(k, :), ...
             'LineWidth', 1.2, 'MarkerSize', 4, ...
             'DisplayName', sprintf('f=%.4f', f_val));
        hold on;
    end
    xlabel('累加步骤（第 i 次噪声特征向量贡献）', 'FontSize', 12);
    ylabel('分母 temp 值', 'FontSize', 12);
    title('关键频率点的分母累加过程', 'FontSize', 14, 'FontWeight', 'bold');
    grid on; legend('Location', 'best', 'FontSize', 10);

    % ====================== 子图 3：分母累加过程热力图（保持不变）
======================
    subplot(2, 2, 3);
    imagesc(freq, 1:noise_dim, temp_intermediate);  % 直接用累加矩阵绘图
    cbar = colorbar;  % 先创建 colorbar 对象（兼容所有版本）
    cbar.YLabel.String = '分母 temp 值';  % 再设置 Y 轴标签（替代只读的 Label 属性）
    xlabel('归一化频率 \omega/2\pi', 'FontSize', 12);
    ylabel('累加步骤', 'FontSize', 12);
    title('分母累加过程热力图', 'FontSize', 14, 'FontWeight', 'bold');
    colormap('jet');

    % ====================== 子图 4：最终分母分布（标注 2 个极小值）
======================
    subplot(2, 2, 4);
    plot(freq, final_denominator, 'g-', 'LineWidth', 1.5);
    hold on;

    % 通用局部极小值检测（数学定义：某点小于左右相邻点）
    den_minima_freq = [];
    den_minima_val = [];
    for i = 2:length(final_denominator)-1
        if final_denominator(i) < final_denominator(i-1) && final_denominator(i)
< final_denominator(i+1)
            den_minima_freq = [den_minima_freq, freq(i)];
            den_minima_val = [den_minima_val, final_denominator(i)];
        end
```

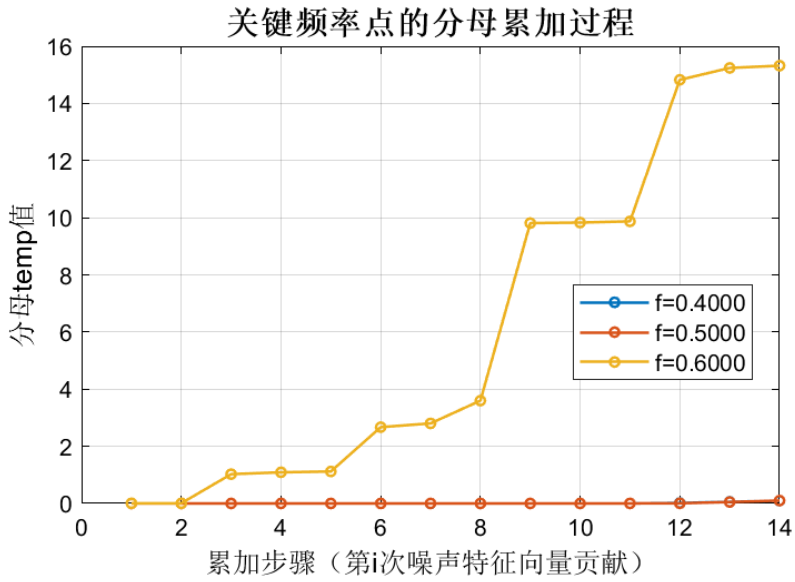
```
    end

    % 选择幅值最小的 2 个极小值
    selected_minima = [];
    if ~isempty(den_minima_val)
        % 按幅值升序排序（从小到大）
        [sorted_vals, sorted_idx] = sort(den_minima_val, 'ascend');
        sorted_freqs = den_minima_freq(sorted_idx);

        % 选前 2 个（不足 2 个则选所有）
        num_select = min(2, length(sorted_vals));
        selected_minima = [sorted_freqs(1:num_select);
sorted_vals(1:num_select)];
    end

    % 错位标注坐标（避免重叠）
    vert_offset = [0.15, 0.10];  % 垂直偏移比例（下移）
    horz_offset = [0.005, -0.005]; % 水平偏移
    y_min = min(final_denominator);
    y_max_den = max(final_denominator);

    for k = 1:size(selected_minima, 2)
        f_val = selected_minima(1, k);
        den_val = selected_minima(2, k);

        % 计算错位位置
        y_offset = den_val * vert_offset(k);
        f_val_offset = f_val + horz_offset(k);
        f_val_offset = max(min(f_val_offset, max(freq)), min(freq)); % 限制在频率
范围内

        % 绘制红点（填充）
        plot(f_val, den_val, 'ro', 'MarkerSize', 6, 'MarkerFaceColor', 'r');
        % 标注坐标（只显示数值）
        text(f_val_offset, den_val - y_offset, sprintf('(%.3f, %.6f)', f_val,
den_val), ...
            'FontSize', 10, 'Color', 'r', 'HorizontalAlignment', 'center');
    end
```

固定两个信号频率分别为0.4和0.5，噪声方差为0.1，A1=A2=1，序列长度N=32。

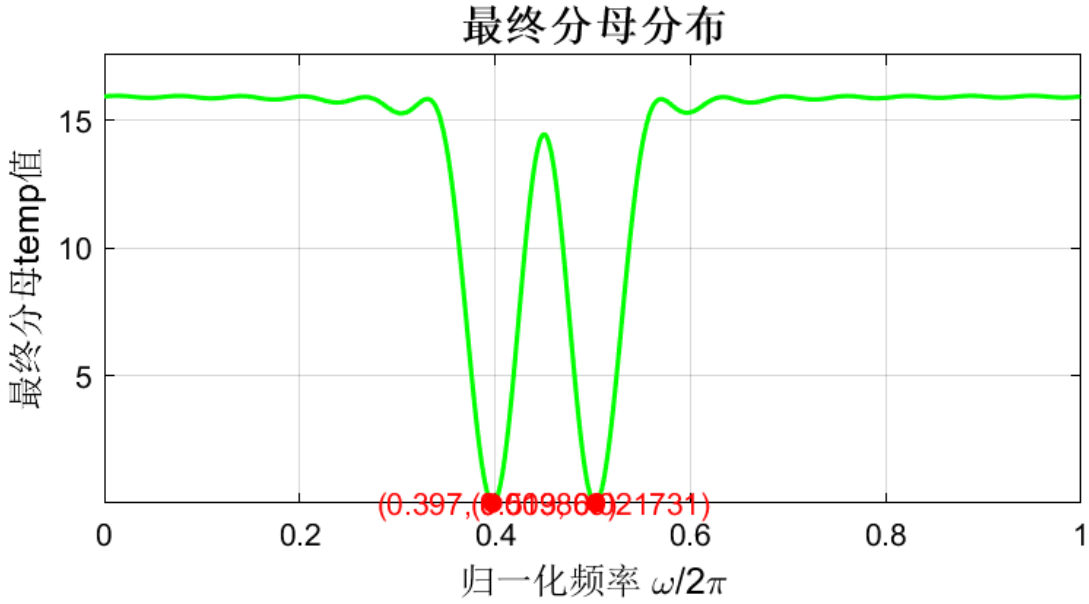


在“关键频率点的分母累加过程”这幅图里，可以清楚看到在归一化频率接近 0.4 和 0.5 的位置，分母的累加曲线随“乘加步数”（也就是噪声子空间的维数 p−K）增长得更慢、数值低。

当我们在“关键频率点的分母累加过程”这幅图里加一个不是信号频率的频率 selected_freqs = [signal_freq,0.6];，再生成这张图



可以看到，在非信号频率处，曲线升得更快、分母明显更大，而为信号频率的 0.4 和 0.5 在这时候几乎看不到有数值。所以在非信号频率处，每一步的投影能量更大，分母迅速抬升，谱值被强烈压制。



========== 展示 m=16 时的 MUSIC 计算过程 ==========

==========

【MUSIC 计算过程】m=16, num=1024（开始计算）

==========

【第 1/14 次乘加】

乘法结果（$|V|^2$）：最小值=0.000000，最大值=14.054709，均值=1.000000

累加后分母：最小值=0.000000，最大值=14.054709，均值=1.000000

【第 2/14 次乘加】
乘法结果（|V|² ）：最小值=0.000000，最大值=14.531583，均值=1.000000
累加后分母：最小值=0.000001，最大值=14.552541，均值=2.000000

【第 3/14 次乘加】
乘法结果（|V|² ）：最小值=0.000000，最大值=9.453978，均值=1.000000
累加后分母：最小值=0.000115，最大值=15.352988，均值=3.000000

【第 4/14 次乘加】
乘法结果（|V|² ）：最小值=0.000000，最大值=8.210245，均值=1.000000
累加后分母：最小值=0.000171，最大值=15.358283，均值=4.000000

【第 5/14 次乘加】
乘法结果（|V|² ）：最小值=0.000000，最大值=8.297816，均值=1.000000
累加后分母：最小值=0.000296，最大值=15.488722，均值=5.000000

【第 6/14 次乘加】
乘法结果（|V|² ）：最小值=0.000000，最大值=7.192916，均值=1.000000
累加后分母：最小值=0.000298，最大值=15.493987，均值=6.000000

【第 7/14 次乘加】
乘法结果（|V|² ）：最小值=0.000000，最大值=9.708619，均值=1.000000
累加后分母：最小值=0.000345，最大值=15.508900，均值=7.000000

【第 8/14 次乘加】
乘法结果（|V|² ）：最小值=0.000000，最大值=13.948240，均值=1.000000
累加后分母：最小值=0.000395，最大值=15.515473，均值=8.000000

【第 9/14 次乘加】
乘法结果（|V|² ）：最小值=0.000000，最大值=8.626811，均值=1.000000
累加后分母：最小值=0.001028，最大值=15.550222，均值=9.000000

【第 10/14 次乘加】
乘法结果（|V|² ）：最小值=0.000000，最大值=8.204788，均值=1.000000
累加后分母：最小值=0.001065，最大值=15.691427，均值=10.000000

【第 11/14 次乘加】
乘法结果（|V|² ）：最小值=0.000000，最大值=13.241211，均值=1.000000
累加后分母：最小值=0.001074，最大值=15.722069，均值=11.000000

【第 12/14 次乘加】

乘法结果（|V|² ）：最小值=0.000001，最大值=7.363118，均值=1.000000
累加后分母：最小值=0.003854，最大值=15.873388，均值=12.000000

【第 13/14 次乘加】
乘法结果（|V|² ）：最小值=0.000000，最大值=8.776509，均值=1.000000
累加后分母：最小值=0.015697，最大值=15.974625，均值=13.000000

【第 14/14 次乘加】
乘法结果（|V|² ）：最小值=0.000000，最大值=13.544326，均值=1.000000
累加后分母：最小值=0.019869，最大值=15.975119，均值=14.000000

==========
【最终结果】
分母范围：[0.019869, 15.975119]，均值=14.000000
MUSIC 谱范围：[0.062597, 50.330366]，峰值=50.330366
==========

图中“最终分母分布”呈现出在归一化频率约 0.397 和 0.503 处明显的局部极小值（图中标注点为$0.397,\ 0.00986$），对应的分母值接近零，因此其倒数即 MUSIC 谱在这些位置出现尖锐峰值，说明系统成功分辨出两路窄带信号。原理上分母由噪声子空间特征向量在频率点处的投影能量累加得到，代码中实现为对每个噪声特征向量做 FFT 并累加，即

$$\text{temp}(\omega) = \sum_{k=M+1}^{P+1} \left|\overrightarrow{e_i}^H \cdot \overrightarrow{v_k}\right|^2$$

而 MUSIC 谱为$P(\omega) = 1/\text{temp}(\omega)$。当$\omega$等于真实信号频率时，导向向量落在信号子空间，与噪声子空间正交，导致各噪声特征向量的投影接近零，从而$\text{temp}(\omega)$出现极小值，$P(\omega)$出现极大值。实验中两处极小值清晰且相互分离，说明所选自相关阶数p、样本长度N提供了足够的分辨率；但分母并非严格为零，这是有限样本、噪声方差和频率网格离散化造成的偏差（噪声使投影不完全为零，频率不在格点上会产生泄露）。

MUSIC 中“每一行乘”的运算就是对噪声特征向量在频率轴上的离散傅里叶变换（DFT）取模平方，再累加作为谱的分母，因此本质上等价于对特征向量做 DFT 采样并计算能量密度。

MUSIC 的阵列/自相关模型中，**导向矢量（或频率向量）**取形式

$$\mathbf{a}(\omega) = \begin{bmatrix} 1 \\ e^{j\omega} \\ \vdots \\ e^{j\omega(p-1)} \end{bmatrix},$$

噪声子空间的第i个特征向量记为$\mathbf{v}_i = [v_{i,0}, v_{i,1}, \dots, v_{i,p-1}]^T$。MUSIC 的分母项对单个噪声特征向量为内积模平方：

$$|\mathbf{a}(\omega)^H \mathbf{v}_i|^2 = \left| \sum_{n=0}^{p-1} v_{i,n}\, e^{-j\omega n} \right|^2 .$$

右侧正是**对序列$v_{i,n}$在频率ω处的离散时间傅里叶变换（DTFT）采样**。若用离散点数N做 DFT/FFT 采样，则对应为

$$V_i[k] = \sum_{n=0}^{p-1} v_{i,n}\, e^{-j2\pi kn/N},$$

于是$|\mathbf{a}(\omega_k)^H \mathbf{v}_i|^2 = |V_i[k]|^2$。MUSIC 的分母是对所有噪声特征向量累加：

$$\mathrm{den}(\omega_k) = \sum_{i \in \mathrm{noise}} |\, \mathbf{a}(\omega_k)^H \mathbf{v}_i|^2 = \sum_i |\, V_i[k]|^2,$$

最终谱为$P(\omega_k) = 1/\mathrm{den}(\omega_k)$。因此**每次对一列特征向量做 FFT、取模平方并累加**，正是代码中看到的“每一行乘再累加”的操作，其数学含义即 DFT 能量叠加。

实验收获和体会

一开始我对“子空间正交性”只是停留在记结论的层面，直到亲手写代码生成含噪复正弦信号，构造自相关矩阵、做特征分解，再一步步实现“每一行乘”的累加过程，才把$|a(\omega)^H v_i|^2$这个公式和 FFT、模平方累加的代码操作对应起来——看着第 14 次乘加后分母最小值落到 0.019869，对应谱峰值达到 50.33，才理解了“信号频率处分母趋近于零、谱峰尖锐”的物理意义。

实验中参数调整的过程，比如一开始用 N=32、噪声方差 0.1，信号频率 0.4 和 0.5 时，m=4 就有清晰双峰，但当频率差缩小到 0.02（0.4 和 0.42），m=4 的结果就出现了明显偏差，直到 m=16 才重新得到准确估计，这让我直观感受到阶数 m 对频率分辨率的关键作用；而当噪声方差从 0.1 增大到 10 时，哪怕 m=10 都完全分辨不出两个信号，直到 m=31 才看到清晰谱峰，这也让我明白了算法对抗噪声的能力不是无限的，噪声越大，需要的阶数越高，这为后续实际应用中调参提供了明确依据。

当两个信号幅度比达到 1:10 时，我原本以为弱信号会被强信号“掩盖”，但 m=16 时依然能分辨出两条谱线，这比之前实验里的 Levinson 递推法表现好太多，也让我真正理解了子空间方法的核心优势——它不是靠拟合数据来逼近信号，而是通过正交性从噪声中“分离”出信号，这也解释了为什么它高阶不会像 AR 模型那样出现伪峰。另外，序列长度 N 从 32 增加到 128 时，同样噪声方差 5、m=16 的条件下，谱峰变得更尖锐，频率估计误差也更小，这让我深刻体会到“更多数据意味着更多信息”，在信号处理中，数据量和算法本身同样重要。