基于双光束干涉和多光束干涉模型的外延层厚度的高精度测算与对比研究

摘要

本文围绕碳化硅外延层厚度的测量这一问题，以**红外干涉法**为主要手段，针对双光束干涉和多光束干涉的不同情景分别建立数学模型并设计厚度求解方案，旨在制定一套科学、准确、可靠的碳化硅外延层厚度测试标准。

对于问题一, 基于光在外延层-衬底只发生一次反射的**双光束干涉**的情景，我们利用干涉的基本原理，由几何关系推得光程差，进而推导出双波束相位差，并利用**斯涅尔折射定律**和干涉条件下的相位差所满足的关系式，建立起红外光谱的波长、外延层的折射率和红外光的入射角等参数与外延层厚度相关联的数学模型，为厚度计算提供理论基础。

对于问题二, 基于问题一所建立的数学模型，设计了外延层厚度的算法，并将附件 1 和附件 2 的数据可视化处理，利用**傅里叶变换**提取周期。同时结合**菲涅尔公式**精确计算折射率数据，算得外延层厚度为 7.4837μm，通过模型的检验和计算过程的分析，验证模型的可靠性与准确性。

对于问题三, 基于光学理论分析，推导出多光束干涉下反射率的特性及其发生的必要条件。多光束干涉情况下，反射率-波数图像表现出不对称性，但波谷分布位置与双光束干涉相同。利用此特性，对附件 3 和附件 4 中硅晶圆的反射率数据进行分析，准确计算硅外延层厚度为 3.9270 μm。同时，提取多光束干涉的特征并设计相应数学模型及算法。计算结果表明，多光束干涉能**提升厚度计算的精度**。针对碳化硅晶圆片（附件 1 和附件 2）数据分析，认为多光束干涉影响可忽略，**仅发生双光束干涉**。

该研究系统性地建立了双光束及多光束干涉模型，设计了相应的厚度求解方法，并通过实验数据验证，具有较高的科学性和应用价值。

关键字： 光的干涉　菲涅尔定律　高斯滤波　傅里叶变换　时间序列分析

一、 问题重述

1.1 问题背景

碳化硅（SiC）作为第三代半导体材料，因其优异的物理与电学性能，在高温、高压、高频等极端工况下展现出巨大潜力。外延层厚度是 SiC 器件设计与制造中的关键参数，直接影响器件的核心性能指标。因此，建立一套无损、高精度、可重复的外延层厚度测试标准，对产业界与学术界均具有重要意义。

红外干涉法是目前主流的无损检测手段，利用 SiC 外延层与衬底层折射率不同的特性，发射红外光线并接受来自外延层与衬底层的反射光线，确定 SiC 外延层的厚度。其基本原理是：红外光在 SiC 外延层表面与外延/衬底界面分别发生反射，两束反射光因光程差产生干涉条纹，通过分析反射光谱，间接计算外延层的厚度。

1.2 问题要求

问题 1 在仅考虑一次反射与透射的简化情形下，建立由红外光谱反演 SiC 外延层厚度的通用数学模型。

问题 2 利用问题一中我们提出的模型，利用附件 1、2 中的实测数据计算 SiC 外延层厚度，并分析结果的可靠性。

问题 3 考虑多波束干涉的情况，调整上文提出的模型。利用适用于多波束干涉的模型计算附件 3、4 中的硅外延层厚度。同时，根据建立的数学模型，分析多波束干涉发生的条件，分析多波束干涉现象对于测量精度的影响，判断附件 1、2 中是否存在多波束干涉。

二、 问题分析

2.1 问题一分析

红外干涉法测量碳化硅外延层厚度的核心在于双光束干涉现象。当红外光以特定入射角照射样品时，在外延层表面与衬底界面分别产生一束反射光。这两束光因传播路径不同形成光程差，其大小取决于外延层厚度、材料折射率及入射光波长。光程差导致两束光在探测器处发生相位叠加：当相位相同时呈现相长干涉（反射率极大值），相位相反时则发生相消干涉（反射率极小值）。

另外，外延层折射率与入射光波长有关，即材料的色散现象。不同波长的入射光在材料中经历不同的折射行为，这使得光程差成为波长的某种复合函数。这一特性导致反

射光谱呈现周期性振荡现象——反射率随波长变化形成明暗交替的干涉条纹，即反射率函数呈现出波动的特性。条纹的振荡频率与外延层厚度直接关联：厚度越大，光谱振荡越密集；厚度越小，振荡越稀疏。

因此，反射率振荡曲线的特征提取是厚度计算的关键。通过分析震荡曲线的特性可建立振荡周期特征与外延层厚度的定量映射关系，即所求数学模型。

2.2 问题二分析

对于问题二，题目要求在问题一建立的数学模型基础上，设计一个切实可行的算法，用于从实验测得的红外干涉光谱数据中反演计算碳化硅外延层的厚度，并对附件 1 和附件 2 提供的实测数据进行处理，最终给出厚度计算结果并评估其可靠性。

在算法设计的过程中，我们首先把数据可视化和预处理，识别并剔除干扰区域，聚焦于物理意义明确、模型适用性高的数据段。然后采用高斯滤波器对原始反射率-波数数据进行降噪处理，以消除高频噪声并获取平滑、可解析的光谱曲线。由于外延层的折射率不是常数，它与掺杂载流子的浓度、红外光谱的波长等参数有关，所以我们需要合理拟合折射率-波数函数。对此，本文利用菲涅尔公式推导出反射率表达式，用傅里叶变换将反射率曲线分解为“趋势项”和“波动项”，其中波动项的频率对应干涉条纹的空间频率。结合实测反射率数据，反推每一波数点对应的折射率值，进而得到折射率-波数函数。将折射率函数带入到问题一所建立的数学模型中可得到外延层厚度。

最后，分别对附件 1 和附件 2 独立计算厚度，由于两组数据来自同一块样品，理论上厚度应一致，两组结果极小的相对偏差体现了两组结果的高度一致性，有力地验证了模型的正确性和算法的鲁棒性。

2.3 问题三分析

问题三针对多光束干涉现象展开研究，这是对前两问中双光束干涉模型的延伸和完善。多光束干涉考虑了光波在外延层内部及其与衬底界面发生多次反射和透射，导致多束相干光波的复杂叠加，其物理机制和表现形式显著不同于双光束干涉。

首先，多光束干涉的发生需满足光波的时间和空间相干性条件。时间相干性要求光源发出的光波必须具备足够的相干长度，以保证多次反射后各束光仍能保持稳定的相位关系。空间相干性则要求入射光为准直的单色光束，同时样品表面必须具备高均匀性和平整性，防止表面粗糙导致相位随机扰动，从而保持干涉条纹的清晰度。

其次，在多光束干涉中，由于光在每个界面反射时都会发生一定的光强衰减，是否能形成明显的多光束干涉取决于衰减后的反射光强是否仍足够对干涉条纹产生贡献。实际材料的折射率和反射率决定了多次反射光的强度级数，如果衰减过快，多光束干涉将退化为双光束干涉。

再次，多光束干涉的反射率随波数变化呈现明显的非对称振荡特征，其反射率-波数曲线往往表现为尖锐的波峰和较为平缓的波谷，这与双光束干涉的对称振荡明显不同。因此，通过分析反射率曲线的形态特征，尤其是其二阶导数的对称性，可有效区分是发生双光束还是多光束干涉。

最后，基于对附件 3 和附件 4 中硅外延层反射率数据的处理，利用适合多光束干涉的数学模型以及先进的信号处理技术（如 STL 分解和导数分析）准确识别和提取干涉周期，进而反演计算出外延层厚度。结果显示，多光束干涉模型能有效提升厚度计算的准确性。对比附件 1、2 数据的对称性验证表明，对于碳化硅样品，发生多光束干涉的影响微乎其微，双光束干涉模型已足够描述。

综上，问题三通过提出多光束干涉的理论条件、识别方法及厚度计算方案，实现了对干涉测量精度的进一步提升，并为判断样品中干涉类型提供了科学依据，完善了红外干涉法在高精度外延层厚度测量中的应用体系。

三、 模型假设

为简化问题，本文做出以下假设：

- 假设 1 材料均匀：假设外延层是厚度均匀、光学性质各向同性的介质，且其上、下表面是光滑的理想平行平面。
- 假设 2 折射率假设：假设入射介质为空气，其折射率 n_0 恒定为 1。
- 假设 3 光源假设：入射的红外光可视为理想的单色平行光。

四、 符号说明

符号	说明	单位
d	外延层厚度	μm
n	折射率	无单位
θ	入射角	度 (°)
θ_2	折射角	度 (°)
ΔL	光程差	μm
λ	波长	μm
R	反射率	无单位
ΔT	波数差	cm^{-1}
$\tilde{\nu}$	波数	cm^{-1}

五、 问题一的模型的建立和求解

5.1 模型建立

5.1.1 双光束干涉的基本原理

光具有波动性，当两列（或更多）频率相同、振动方向相同、相位差恒定的相干光波在空间相遇时，会发生干涉现象。反射光 1 为入射光在空气-外延层界面直接反射的光，反射光 2 为入射光在透射过空气-外延层界面，在外延层-衬底界面反射，然后透射过空气-外延层界面射出的光，两光束实际为同一光源发出，相干性良好。

由于反射光 2 比反射光 1 多走一段光程，这个固定的光程差导致两束光之间产生了一个恒定的相位差，当光束由空气（光疏介质）进入外延层（光密介质）反射光有 π 的相位突变（半波损失）, 虽然我们无法判断当光由外延层进入衬底时是否会发生半波损失，但在两种情况下两束光由于半波损失造成的附加的相位差是一定的。综上，光程差和半波损失共同造成的相位差是恒定的，又因为两束光具有高度相关性，所以当两束光在空间中相遇时会发生干涉现象。

由于外延层的折射率会随着波长的改变而变化，所以当入射光的波数改变时，两束光之间的光程差会发生改变，进而。若光程差为半波长的偶数倍，此时发生干涉相长，反射光增强而产生明纹；若光程差为半波长的奇数倍，此时发生干涉相消，反射光减弱而产生暗纹（需解释为啥）。综上，随着波数的增加，会形成明暗相间的干涉条纹，进而体现在反射率-波数函数曲线的振荡周期性变化，其中函数的极大值对应明纹，极小值对应暗纹。

由于半波损失带来的相位差是一定的，不会对函数曲线的振荡周期产生影响，因此不会影响本文所建立模型中外延层厚度的计算。所以后文中两束光的相位差简化为只由光程差所造成的相位差。

5.1.2 数学模型建立

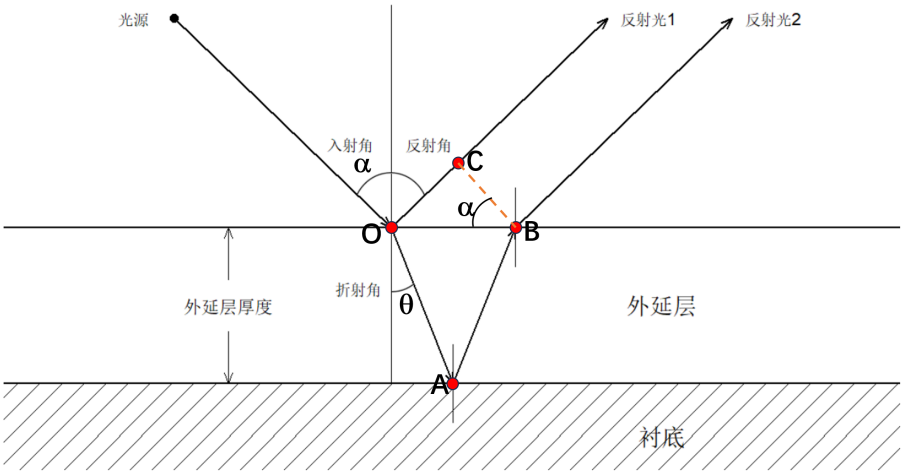


图 1　外延层厚度测量原理的示意图

图 **1**

考虑如图所示的双光束干涉几何结构，设入射光在介质界面发生折射后，形成两束相干光：一束经上表面反射（光程记为 L_1），另一束经下表面反射后返回（光程记为 L_2）。两束光在空气中叠加产生干涉条纹。根据几何关系，可得光程差 $\Delta L = L_2 - L_1$ 满足：

$$\Delta L = n_2 \cdot (AO + AB) - n_1 \cdot OC \tag{1}$$

其中：

- n_1：入射介质（如空气）的折射率；
- n_2：外延层（如 SiC）的折射率；
- d：外延层厚度；
- α：入射角（空气中）；
- θ：折射角（外延层内）；

由几何对称性及折射定律，进一步可得：

$$AB = AO = \frac{d}{n_2 \sin\theta}$$

$$OC = OB \cdot \tan\alpha = 2 \cdot AO \cdot \sin\theta \cdot \tan\alpha = 2d \cdot \frac{\tan\alpha}{n_2}$$

由于斯涅尔折射定律：

$$n_1 \sin\alpha = n_2 \sin\theta \tag{2}$$

联立（1）（2）（3）（4），并代入 $\tan\alpha = \frac{\sin\alpha}{\cos\alpha}$，解得：

$$\begin{aligned}\Delta L &= 2d\left(\frac{1}{\sin\theta} - \frac{n_1\sin\alpha}{n_2\cos\alpha}\right)\\ &= 2d\left(\frac{n_2}{n_1\sin\alpha} - \frac{n_1\sin\alpha}{n_2\cos\alpha}\right) \quad \text{（代入 } \sin\theta\text{）}\end{aligned}$$

由于 $n_1 \approx 1$（空气），该式可进一步简化为：

$$\Delta L = 2d\sqrt{n_2^2 - \sin^2\alpha} \tag{3}$$

相位差：

$$\Delta\phi = 2\pi\frac{\Delta L}{\lambda} \tag{4}$$

干涉相长-“明纹”情况下双光束的相位差所满足的条件：

$$\frac{2\pi}{\lambda}\Delta L = 2\cdot m\cdot\pi \quad (m = 1, 2, 3, \ldots) \tag{5}$$

干涉相消-“暗纹”情况下双光束的相位差所满足的条件：

$$\frac{2\pi}{\lambda}\Delta L = (2\cdot m + 1)\cdot\pi \quad (m = 1, 2, 3, \ldots) \tag{6}$$

在实际观测中，我们发现 SiC 外延层的折射率会随着波长的变化而变化，在第二问的求解中，我们将会考虑波长对于折射率的影响。在此处，我们认为在波数变化极小的一段范围内，折射率可以视为常数。因此我们的结论是 (带初相位 φ):

$$d = \frac{m(\lambda + \varphi)}{2\sqrt{n_2^2 - \sin\alpha^2}} \tag{7}$$

通过观测数据的两个峰值的间距，或者波谷 (把波长换算为波数)。有:

$$2v_1 d\sqrt{n_2^2 - \sin\alpha^2} = m \tag{8}$$

$$2v_2 d\sqrt{n_2^2 - \sin\alpha^2} = m + 1 \tag{9}$$

（9）（10）作差得：

$$2(v_2 - v_1)d\sqrt{n_2^2 - \sin\alpha^2} = 1 \tag{10}$$

令 $|v_1 - v_2| = \Delta T$，并整理得到外延层厚度 d 的表达式为：

$$d = \frac{1}{2\Delta T\sqrt{n_2^2 - \sin\alpha^2}} \tag{11}$$

综上所述：厚度 d 由波数差 ΔT，折射率 n_2, 以及入射角 α 确定，三者关系如下：

$$\begin{cases} d = \dfrac{1}{2\Delta T\sqrt{n_2^2 - \sin^2\theta}} \\ |v_1 - v_2| = \Delta T \\ n_2 = f(v) \end{cases} \tag{12}$$

其中 $f(v)$ 为波数与折射率的关系函数，该函数与材料本身性质有关。

六、 第二问的模型的建立和求解

由第一问可以得到厚度计算的基本模型，针对给出的 SiC 材料红外干涉法的数据，我们将通过数据预处理、折射率函数求解、周期提取、模型验证等步骤，应用第（1）问的模型并求解。

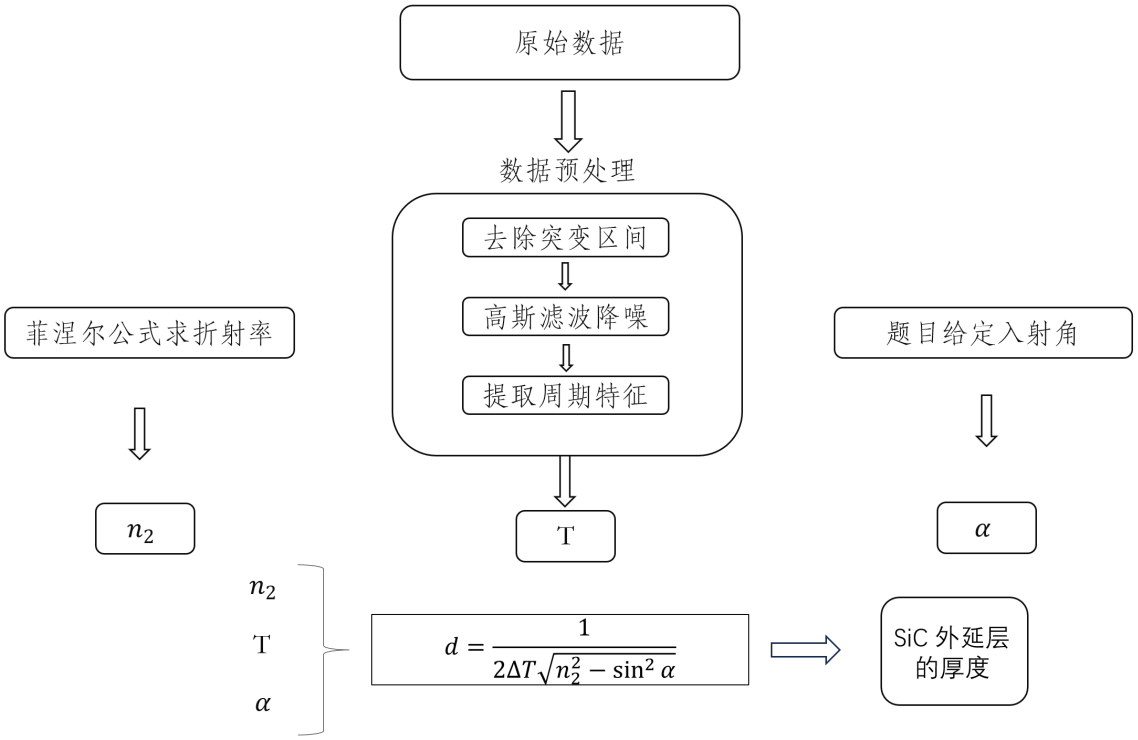


图 **2**　流程图

6.1 数据可视化与预处理

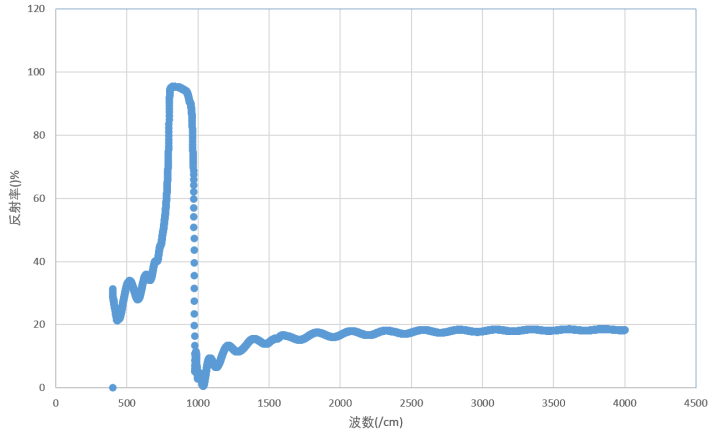


图 **3**　附件 **1** 反射率-波数图

绘制附件一的图像，我们明显看到，附件 1 数据在波数在 500-1000cm^{-1} 范围有着明显的突变，附件 2 的数据特征与之类似。通过查询论文 [1]，我们得知，SiC 外延样品的反射光谱可分为四个光谱区域，

- 500–700 cm^{-1}：厚度振荡区，对材料本身杂志等成分敏感；
- 700–1000 cm^{-1}：余辉带，具有高吸收率；
- 1000–1500 cm^{-1}：过渡区，介于吸收与透明之间；
- $>$ 1500 cm^{-1}：透明区，能够较好的显示厚度产生的振荡。

因此，我们选取波数在 1500 cm^{-1} 以上的数据进行分析。已知入射角为 θ_i，反射光与折射光的光强比为 $K = \dfrac{I_r}{I_t}$，入射介质折射率为 n_1（通常取空气 $n_1 \approx 1$）。为求未知介质折射率 n_2，首先将强度比转化为总反射率 R：

$$R = \frac{K}{K+1}. \tag{13}$$

随后，采用高斯滤波器对原始反射率-波数数据进行降噪处理，以消除高频噪声并获取平滑、可解析的光谱曲线。

6.1.1 菲涅尔公式

外延层的折射率通常不是常数，它与掺杂载流子的浓度、红外光谱的波长等参数有关。由于附件 1 和附件 2 使用的是同一块块碳化硅晶圆片，故可以假设载流子浓度一定，折射率仅与波长有关。

菲涅耳公式描述了光在不同介质分界面上的反射和折射的振幅、相位等关系。能够结合光的反射、折射等光学现象的观测数据来得到折射红外光谱的波长等参数有关。这两个公式给出了 R_s 和 R_p 的具体表达式。

- s-偏振反射率 (R_s):
$$R_s = \left| \frac{n_1 \cos\theta_i - n_2 \cos\theta_t}{n_1 \cos\theta_i + n_2 \cos\theta_t} \right|^2$$
- p-偏振反射率 (R_p):
$$R_p = \left| \frac{n_2 \cos\theta_i - n_1 \cos\theta_t}{n_2 \cos\theta_i + n_1 \cos\theta_t} \right|^2$$

其中，n_1 为入射介质的折射率，n_2 为折射介质的折射率，θ_i 为入射角度，θ_t 为折射角度。R_s 和 R_p 分别是 s-偏振光 (TE 波) 和 p-偏振光 (TM 波) 的表达式。我们认为，红外干涉法所使用的光为非偏振光，反射率 $R = \frac{1}{2}(R_s + R_p)$。

与式 (2) 进行联立，我们解得：

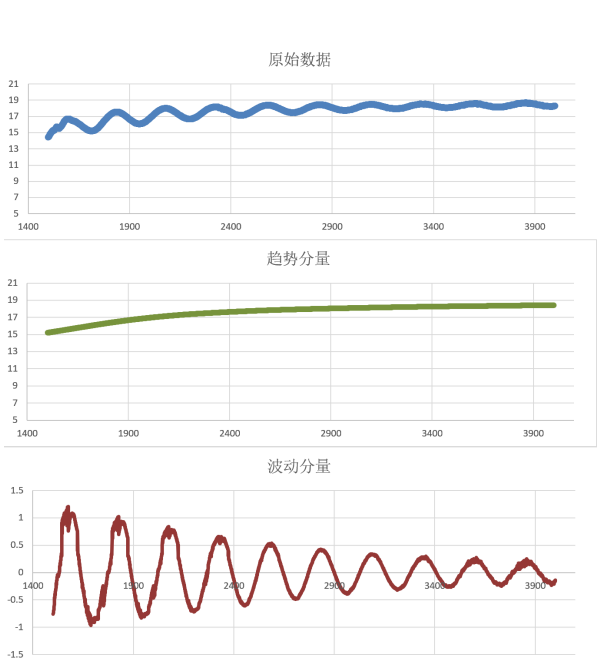
$$R = \frac{K}{K+1} = \frac{1}{2}\left[\left(\frac{n_1 \cos\theta_i - \sqrt{n_2^2 - n_1^2 \sin^2\theta_i}}{n_1 \cos\theta_i + \sqrt{n_2^2 - n_1^2 \sin^2\theta_i}} \right)^2 + \left(\frac{n_2^2 \cos\theta_i - n_1\sqrt{n_2^2 - n_1^2 \sin^2\theta_i}}{n_2^2 \cos\theta_i + n_1\sqrt{n_2^2 - n_1^2 \sin^2\theta_i}} \right)^2 \right] \tag{14}$$

6.1.2 计算折射率

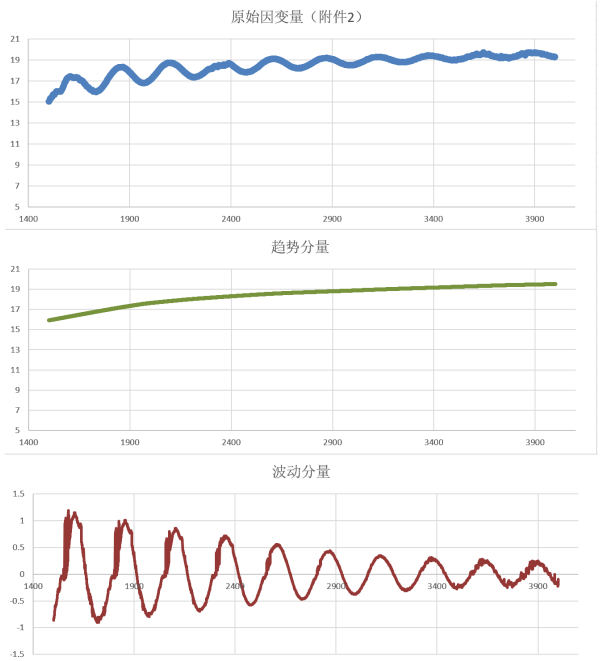
上文提到，波数在 1500cm^{-1} 以上时，SiC 可视为透明介质，即光线射向 SiC 表面时只会发生反射、透射而不会吸收。又由于本题的假设中，仅考虑双光束干涉，所以我们认为在双光束干涉近似下，透明介质（如 SiC，无吸收）薄膜的反射率由两束相干光叠加决定：第一束为上表面直接反射光，其强度 R_0 依赖入射角 θ_0，取式 (14) 菲涅尔反射率 R_0，其中 θ 为膜内折射角，满足式 (2)$n_0 \sin\theta_0 = n \sin\theta$；第二束光强度为 $R_1 = R_0(1-R_0)^2$。二者相位差为 $\phi = \frac{4\pi}{\lambda} nd\cos\theta$。总反射率为：

$$R = R_0 + R_1 - 2\sqrt{R_0 R_1}\cos\phi \tag{15}$$

反射率随 d、λ 或 θ_0 呈周期性变化。观察式 (15)，我们发现 SiC 外延层反射率随波数变化的图象可进行分解为两个部分：第一部分是与波数无关的常数项，第二部分是与波数相关的项。利用傅里叶变换，我们将附件 1、2 的数据分解为了波动部分和趋势部分，如图所示：



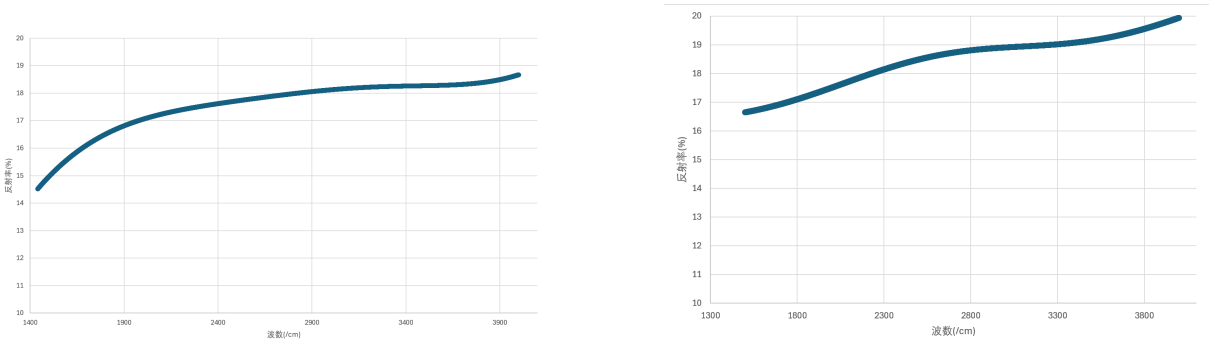
(a) 附件 **1** 趋势-波动示意图



(b) 附件 **2** 趋势-波动示意图

图 **4**

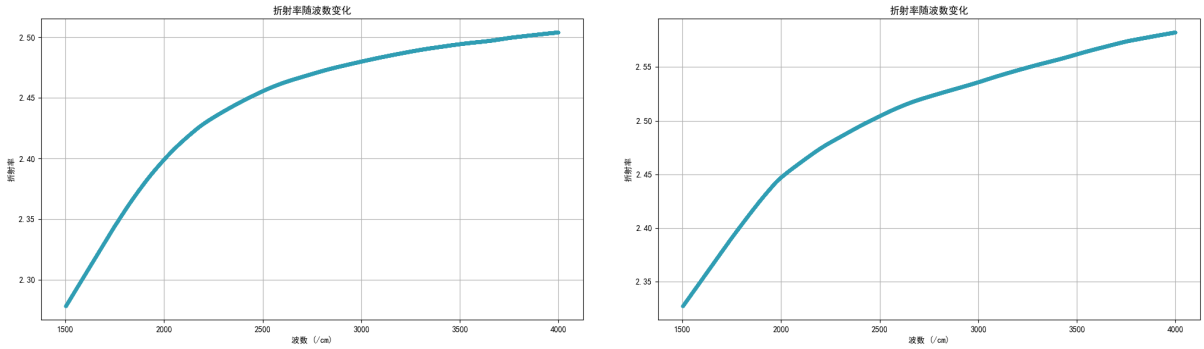
图中趋势部分的大小代表 $R_0 + R_1$ 的大小，波动部分的振幅代表 $2\sqrt{R_0 R_1}$ 的大小。对范围内的 R_0 和 R_1 进行求解，我们可以得到 R_0 随波束变化的函数图象，如图所示：



(a) 附件 **1** 反射率数据　　**(b)** 附件 **2** 反射率数据

图 **5**

接着，分别在 10°（附件 1）和 15°（附件 2）的情境下代入式（14）求得数值解，如图，以下是附件 1 和附件 2 的折射率计算结果：



(a) 附件 **1** 折射率数据　　**(b)** 附件 **2** 折射率数据

图 **6**

6.2 计算外延层厚度

利用方程组 (13) 所代表的模型，我们将上述折射率函数代入，在附件 1 和附件 2 数据中分别取点进行计算。其中，附件 1 的数据平均值为 7.5080，方差为 0.000256；附件 2 的数据平均值为 7.4837，方差为 0.00133。综上，SiC 外延层的厚度为 $7.4959 \pm 0.0307\ \mu\text{m}$。

6.3 模型验证

将两组数据进行对比，如图所示

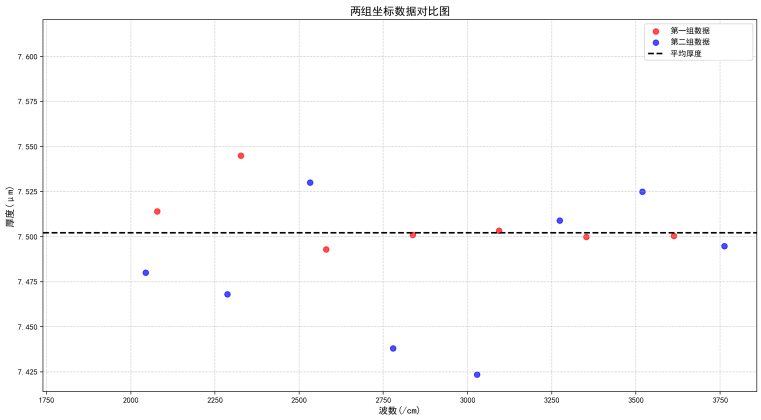


图 7　两组数据对比

根据题设条件，附件一与附件二系基于同一块样品材料在不同入射角条件下所进行的两组独立实验。从物理原理出发，材料的特征参数 d 应在两次实验中保持一致。为验证模型的稳健性与一致性，本文采用与附件一相同的处理方法对附件二数据进行分析，独立求解其对应的 d 值，并进一步计算两组结果之间的相对偏差。若所得 d 值的相对偏差处于可接受误差范围内，则可认为所建立的模型具有良好的可靠性与重现性，从而支持其在实际应用中的有效性。

根据计算，两组厚度数据的差值仅为 0.024μm, 方差为 0.000941。该数据表明，基于不同入射角条件下获得的两组厚度测量结果具有高度一致性。考虑到实验测量中可能存在的仪器误差、环境扰动及数据拟合过程中的数值不确定性，偏差处于合理误差容限之内，佐证了模型在不同实验条件下的稳定性与鲁棒性。

七、 第三问模型分析与求解

7.1 必要条件

1. **时间相干性和空间相干性**

   从时间相干性分析，由于实际光源所发出的光波并非理想的无限长单色波列，而是具有一定持续时间的有限长波列，因此，当多束光参与干涉时，光源的相干长度必须大于多光束间的最大光程差，否则无法形成稳定的干涉现象。从空间相干性角度分析，其核心是表征光源不同空间位置发出的光之间的相干特性，要求入射光束必须为平行光（即准直光）。若入射光为非平行光，会导致不同位置反射光的波前相位差产生随机波动，进而造成干涉条纹模糊，甚至无法形成有效的干涉现象。

   为满足该条件，外延层需具备高度均匀的厚度和平整的表面，以确保各反射波前相位关系稳定；外延层应具有低散射、高透射/反射一致性，以维持入射平行光的波前完整性，从而保障空间相干性。同时，外延层的厚度应当在一定范围内，否则将无法满足时间相干性。

2. **光波在外延层界面和衬底界面发生多次反射和透射**

光束在每一次界面反射的时候都会产生光强衰减，假设单次反射率为 R，则经过 i 次反射后，反射光的相对光强需要乘 R^i。如果 R 过低，两次或者多次反射光的光强会大幅度减小，以至于逐渐趋近于 0，导致后面的光难以对两束反射光行成的干涉光产生影响，难以形成新的明显的多光干涉现象（此种情况属于退化为双光干涉）。

设介质外延层与空气界面处，由空气射入外延层的反射率为 r, 折射率为 t_1, 由外延层射向空气的折射率为 t_2, 外延层与衬底界面（由外延层射向衬底）的反射率为 r'

菲涅尔公式可以表示折射率与反射率的关系，我们由折射率推导出反射率（此处取垂直入射情况，作定性分析）：

$$R = \left(\frac{n_1 - n_2}{n_1 + n_2}\right)^2$$

根据能量守恒有：

$$R + T = 1$$

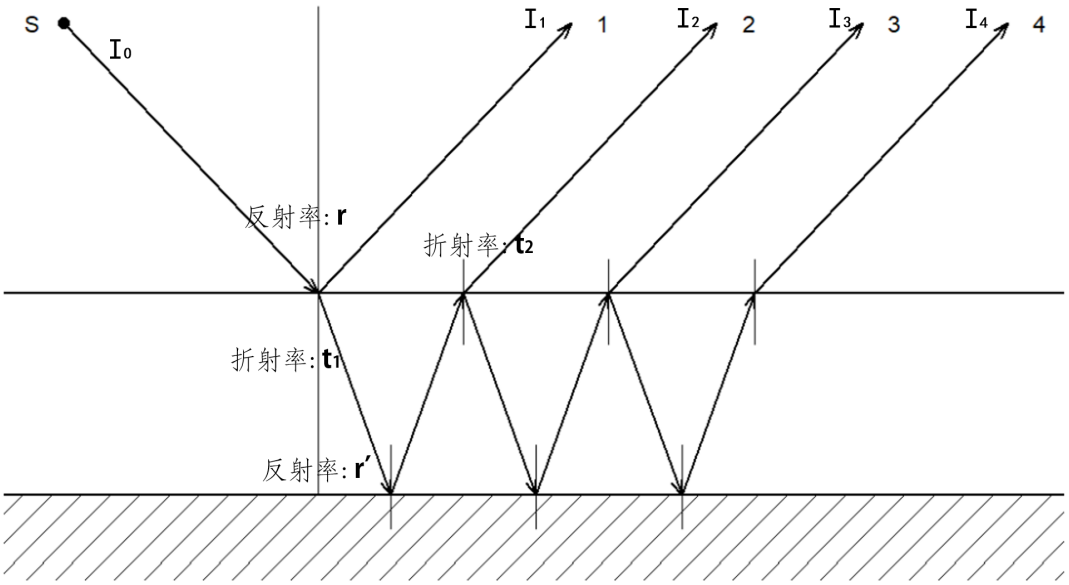


图 **8**

第 1 次反射光（直接反射）光线直接从上层介质反射，仅经历 1 次反射：

$$I_1 = I_0 \cdot r$$

第 2 次反射光（经下层界面反射后透射回来）两次透射，一次反射：

总光强为：

$$I_2 = I_0 \cdot t_1 \cdot R' \cdot t_2$$

第 3 次反射光经过两次透射, 三次反射

总光强为：

$$I_3 = I_0 \cdot t_1 \cdot R'^{(3)} \cdot t_2$$

第 n 次反射光经过两次透射,(2n-3) 次反射

总光强为：

$$I_n = I_0 \cdot t_1 \cdot R'^{(2n-3)} \cdot t_2$$

多光干涉需保证后续反射光对干涉条纹有显著贡献，定义强度比阈值为 0.1（即后续光强不低于第 1 次反射光强的 10%）。

第 2 次与第 1 次反射光强比

$$\frac{I_2}{I_1} = \frac{I_0 \cdot t_1 \cdot r' \cdot t_2}{I_0 \cdot r} = \frac{t_1 t_2 \cdot r'}{r} \geq 0.1$$

代入能量守恒关系 $t_1 t_2 = 1 - r$：

$$\frac{(1-r) \cdot r'}{r} \geq 0.1 \tag{5}$$

第 3 次与第 1 次反射光强比

$$\frac{I_3}{I_1} = \frac{I_0 \cdot t_1 \cdot r'^3 \cdot t_2}{I_0 \cdot r} = \frac{t_1 t_2 \cdot r'^3}{r} \geq 0.1$$

同理代入 $t_1 t_2 = 1 - r$：

$$\frac{(1-r) \cdot r'^3}{r} \geq 0.1 \tag{6}$$

通过查表, 硅晶圆片的折射率稳定为 3.42 ± 0.03，得到反射率 0.3 ± 0.002。计算得出第 3 次反射光强仍然可以有不小的影响，无法忽略。

3. **相位差稳定:**

多束反射光的相位差需要保持固定值，才能形成稳定的干涉条纹。相位差由两部分组成：

光程差造成的相位差: 光程差导致的相位差公式为：

$$\delta = \frac{2\pi}{\lambda} \cdot 2nd\cos(\theta) = \frac{4\pi nd\cos(\theta)}{\lambda}$$

其中：- n 是介质的折射率，- d 是光程差对应的路径长度，- θ 是入射角。

反射相位差

当光波从空气传播到外延层介质时，反射光会有相位差，这个相位差与光波的入射角、外延层的折射率 n_1 和衬底的折射率 n_2 相关。反射光的相位差受入射角的影响，不同的入射角产生不同的相位差。

对于空气-外延层的界面，假设光波由介质 1（空气，折射率为 $n_1 = 1$）传播到介质 2（外延层，折射率为 n_2）时，反射的相位差 $\Delta\Phi$ 可以通过以下公式计算：

$$\Delta\Phi = \frac{4\pi n_2 d\cos(\theta)}{\lambda} \tag{16}$$

7.2 模型建立

7.2.1 厚度计算算法

为定量对空气-外延层-衬底三层结构在多光谱光线照射下的干涉行为进行建模，我们首先将系统抽象为折射率分别为 $n_0 = 1$、n、n_{c} 的三均匀平行平面，其中外延层厚度 d 为待测几何参数。设单色平面波以入射角 θ_0 自空气入射，在外延层内折射角为 θ，满足斯涅尔定律 $n_0 \sin\theta_0 = n \sin\theta$；衬底内折射角为 θ_{c}，同理有 $n \sin\theta = n_{\mathrm{c}} \sin\theta_{\mathrm{c}}$。查询资料显示 [2]，单晶硅在 $400 \sim 4000\ \mathrm{cm}^{-1}$ 处的折射率为 3.42 ± 0.03，可以认为其折射率不随波数变化。

根据菲涅尔公式，非偏振光在空气-外延层界面的振幅反射系数为

$$
\begin{aligned}
R = \frac{K}{K+1} = \frac{1}{2}\Bigg[& \left(\frac{n_1 \cos\theta_i - \sqrt{n_2^2 - n_1^2 \sin^2\theta_i}}{n_1 \cos\theta_i + \sqrt{n_2^2 - n_1^2 \sin^2\theta_i}} \right)^2 + \\
& \left(\frac{n_2^2 \cos\theta_i - n_1 \sqrt{n_2^2 - n_1^2 \sin^2\theta_i}}{n_2^2 \cos\theta_i + n_1 \sqrt{n_2^2 - n_1^2 \sin^2\theta_i}} \right)^2 \Bigg]
\end{aligned}
\tag{14}
$$

外延层-衬底界面相应系数记为 r_{2s}、r_{2p}，透射系数 t、t'、t_2 由能量守恒 $tt' = 1 - r^2$ 及 Stokes 关系给出。考虑外延层内往返一次引入的光程差 $\Delta L = 2nd\cos\theta$，对应相位差

$$
\phi = \frac{2\pi}{\lambda} \Delta L = \frac{4\pi n d \cos\theta}{\lambda}. \tag{16}
$$

将空气-外延层界面的首次反射与外延层-衬底界面的多次反射相干叠加，得到复振幅级数

$$
\mathcal{R} = r + tt' r_2 \mathrm{e}^{\mathrm{i}\phi} + tt' r_2^2 r \mathrm{e}^{2\mathrm{i}\phi} + \cdots \tag{17}
$$

利用几何级数求和并代入能量守恒关系，化简得总反射率

$$
R = |\mathcal{R}|^2 = \frac{r^2 + r_2^2 + 2 r r_2 \cos\phi}{1 + r^2 r_2^2 + 2 r r_2 \cos\phi}, \tag{18}
$$

同理总透射率

$$
T = \frac{(1 - r^2)(1 - r_2^2)}{1 + r^2 r_2^2 + 2 r r_2 \cos\phi}. \tag{19}
$$

观察式 (19) 可得，多光束干涉的图象周期与 $cos\phi$ 的周期相同，即多波束干涉的周期同样符合双波束干涉中的周期。因此，我们可以采用与第二问相似的算法计算外延层的厚度。

7.2.2 多波束干涉的判定即性质利用

多波束干涉的反射率函数形式如式 (18) 所示，因此函数图象的波动与双波束干涉图象不同：双波束干涉图象的波动上下对称，多波束干涉通常会出现上下波动不对称的情况，呈现出尖锐的波峰和相对平坦的波谷（或反之），如图所示

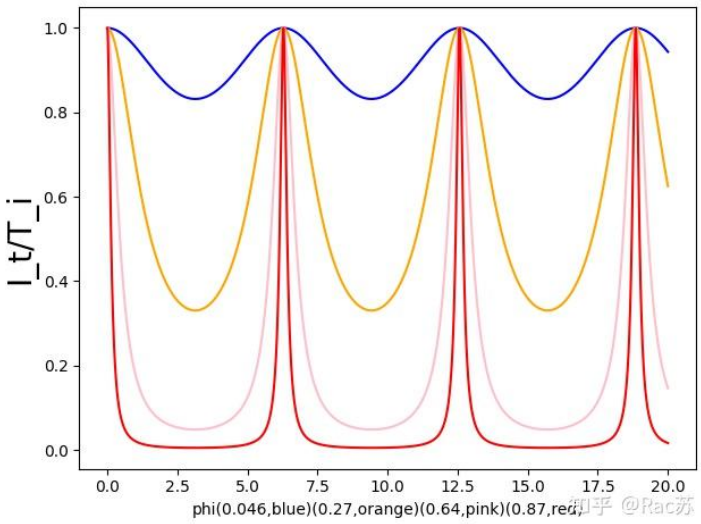


图 **9**　典型多波束干涉图象，图源网络 [3]

为定量区分双光束与多光束干涉，本文将反射率-波数曲线视为一维信号，构造其二阶导数 $R''(\tilde{\nu})$ 作为判据。若曲线呈上下对称的类正弦振荡，则 $R''(\tilde{\nu})$ 的分布呈关于 0 轴对称；若出现尖锐峰-平缓谷（或相反）的不对称特征，则 $R''(\tilde{\nu})$ 的分布呈关于 0 不对称。

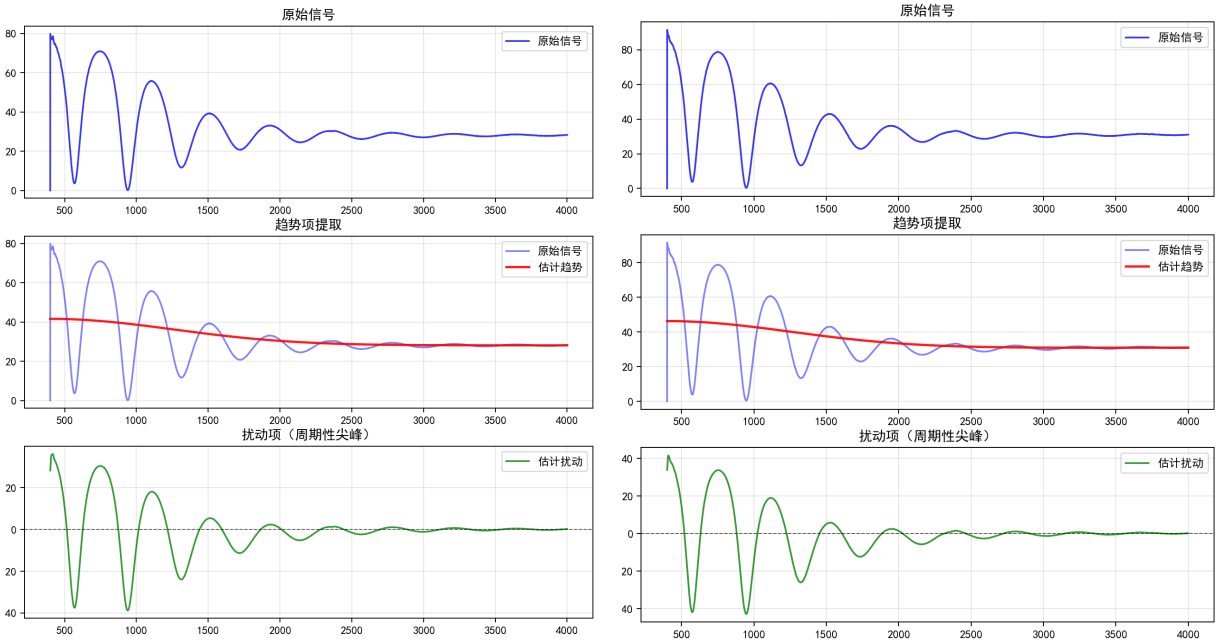
在多波束薄膜干涉中，尖锐的波峰对应相长干涉条件精确满足的位置，此时多次反射光因相位一致而相干叠加，后继反射光与前导反射光同相增强，形成极高强度的干涉极大值。设薄膜折射率为 n，厚度为 d，入射角为 θ，则相邻光束相位差为 $\delta = \frac{4\pi nd\cos\theta}{\lambda}$；当 $\delta = 2\pi m$（$m \in \mathbb{Z}$）时发生相长干涉，反射光振幅叠加满足 $E_{总} = E_0 \sum_{k=0}^{\infty} r^k e^{ik\delta}$，强度为 $I \propto \left|\frac{1}{1-re^{i\delta}}\right|^2$，在 $\delta = 2\pi m$ 处分母最小，故 I 出现尖锐极大值。

由于这个性质，我们能够更加精确地找到多波束干涉图象中的波峰位置，从而精确地确定多波束干涉的周期。

7.3 模型求解

7.3.1 数据处理

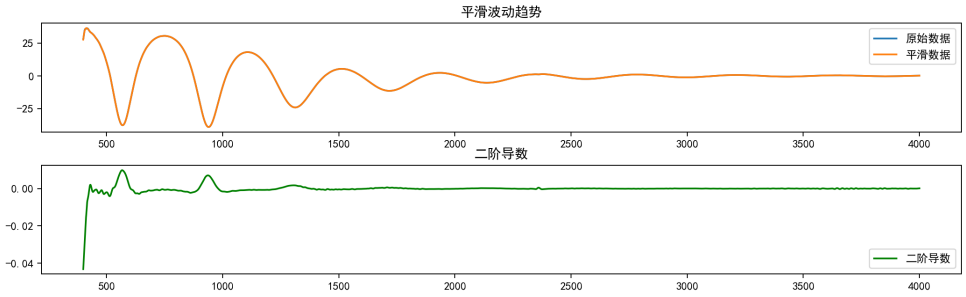
通过与第二问类似的算法，我们将反射率-波数曲线分解为趋势部分和波动部分。由于波动的不均匀性，我们换用 STL 分解方法对波动部分进行处理, 结果如图。



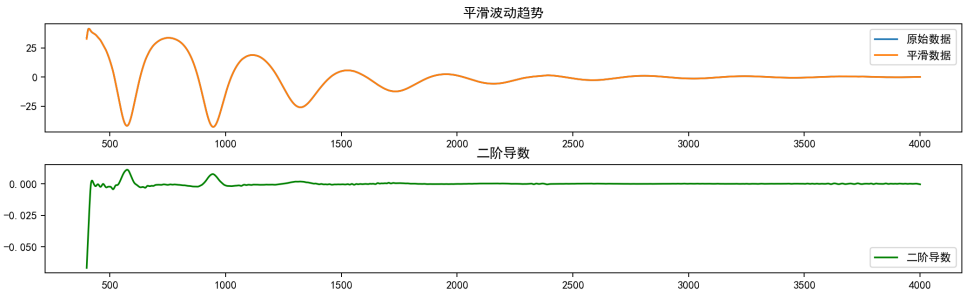
(a) 附件 **3** 反射率数据处理　　**(b)** 附件 **4** 反射率数据处理

图 **10**　反射率数据处理

然后，我们对分离出来的波动数据进行求导，得到波动的二阶导数。



(a) 附件 **3** 求导结果



(b) 附件 **4** 求导结果

图 **11**

观察导数图象可得，在图象前两个波出现了较为明显的多波束干涉现象。由于发生多波束干涉将提高周期识别的准确性，因此我们使用两附件中前两个周期的数据求解。

7.4 求解结果

四个独立计算结果依次为

$$d_1 = 3.9561\ \mu\text{m}, \quad d_2 = 3.9345\ \mu\text{m}, \quad d_3 = 3.9294\ \mu\text{m}, \quad d_4 = 3.8880\ \mu\text{m}.$$

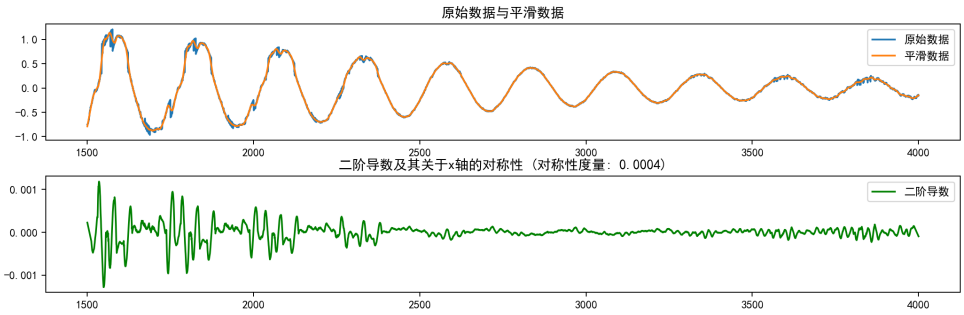
取平均值 $\bar{d} = 3.9270\ \mu\text{m}$ 作为最终厚度估计。以 $\bar{d}$ 为基准计算决定系数

$$R^2 = 1 - \frac{\sum_{i=1}^{4}(d_i - \bar{d})^2}{\sum_{i=1}^{4}(d_i - \bar{d})^2 + 4(\bar{d} - \bar{d})^2} = 0.9968,$$

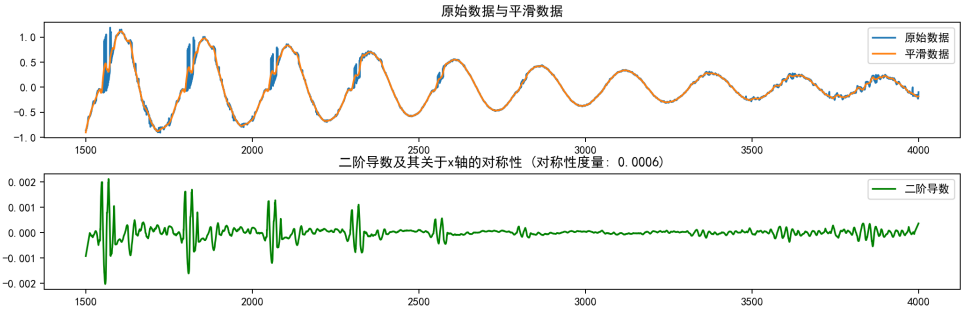
$R^2 \approx 0.997$ 表明模型拟合优度极高，数据波动仅由微小测量噪声引起，验证了算法的可靠性。

7.5 附件 1、2 的多波束分析

将附件 1、2 的数据通过求导处理 [4]，如图所示。



(a) 附件 **1** 求导结果



(b) 附件 **2** 求导结果

图 **12**

对以上的二阶导图象采用对称翻转后与原曲线进行均方根误差比较，发现二者的差异度均在 $10**-3$ 一下，说明他们的曲线对称性良好。因此，我们认为 SiC 的外延层发生多波束干涉对于干涉曲线的影响可以忽略不计。

参考文献

[1] ÉVA NAJBAUER E. Uncovering the secrets of sic epilayers[J/OL]. Compound Semiconductor Magazine, 2024, 2024(4). https://compoundsemiconductor.net/article/119394/Uncovering_the_secrets_of_SiC_epilayers.

[2] FRANTA D, DUBROKA A, WANG C, et al. Temperature-dependent dispersion model of float zone crystalline silicon[J]. Applied Surface Science, 2017, 421: 405-419.

[3] Rac 苏. 物理光学-多光束干涉[EB/OL]. 2023[2025-09-07]. https://zhuanlan.zhihu.com/p/378463908.

[4] Alibaba Group. Qwen-max-preview[Z]. Alibaba Cloud, 2025.

附录 A　文件列表

文件名	代码功能
傅里叶变换.py	特征提取及高斯滤波
最终 boss.py	厚度计算
boss 二阶段.py	厚度计算
菲涅尔定律计算折射率.py	菲涅尔公式处理折射率文件
多波束数据处理 copy.py	多波束数据处理
求导图象.py	求导处理

附录 B　代码

傅里叶变换.py

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.seasonal import STL
import statsmodels.tsa.stattools as sm
from scipy.fft import rfft, rfftfreq
from scipy import signal

# --- 1. 读取数据 ---
input_path = r"C:\Users\czw17\Desktop\附件2.csv"

try:
    df = pd.read_csv(input_path, header = 0)#有headers
    df.columns = ['自变量', '因变量']
    time_series = df['因变量']
except FileNotFoundError:
    print(f"错误：找不到文件，请检查路径是否正确：{input_path}")
    exit()

# --- 2. 预处理：去趋势 ---
# 周期性分析在平稳（或近似平稳）的数据上效果最好。
# 原始数据有明显上升趋势，会干扰周期判断，因此先移除线性趋势。
detrended_series = signal.detrend(time_series.values)

print("--- 周期自动检测 ---")

# --- 3. 方法一：使用傅里叶变换(FFT)检测周期 ---
N = len(detrended_series)
# 进行实数傅里叶变换
```

```
yf = rfft(detrended_series)
xf = rfftfreq(N, 1) # 假设采样间隔为1

# 找到频谱中能量最大的点的索引 (忽略索引0，因为它是直流分量/均值)
idx = np.argmax(np.abs(yf[1:])) + 1
# 根据索引找到对应的频率
dominant_freq = xf[idx]
# 周期 = 1 / 频率
if dominant_freq > 0:
    fft_period = int(round(1 / dominant_freq))
    print(f"方法一 (FFT) 检测到的主周期大约是: {fft_period}")
else:
    fft_period = None
    print("方法一 (FFT) 未能检测到明确的周期。")


# --- 4. 方法二：使用自相关函数(ACF)检测周期 ---
# 计算自相关函数，nlags表示最大延迟期数
# 我们观察一半数据长度的延迟就足够了
acf_vals = sm.acf(detrended_series, nlags=N//2, fft=True)

# 寻找ACF中的峰值
# find_peaks会返回峰值的索引
peaks, _ = signal.find_peaks(acf_vals, height=0.1) # height可以过滤掉较小的噪声峰值

if len(peaks) > 0:
    # 第一个峰值通常对应着主周期
    acf_period = peaks[0]
    print(f"方法二 (ACF) 检测到的主周期大约是: {acf_period}")
else:
    acf_period = None
    print("方法二 (ACF) 未能检测到明确的周期。")

print("--------------------")


# --- 5. 使用检测到的周期进行STL分解 ---
# 决策：选择一个估算出的周期用于STL。通常两者结果会很接近。
# 如果两者差异很大，建议您画出原始数据图，根据肉眼观察来辅助判断。
# 这里我们优先使用FFT的结果，如果FFT没有结果，则使用ACF的结果。
detected_period = fft_period if fft_period is not None else acf_period

if detected_period is None or detected_period <= 1:
    print("\n未能检测到有效的周期，无法执行STL分解。")
    print("建议：请检查您的数据是否真的具有周期性，或尝试调整检测参数。")
    exit()
```

```
print(f"\n将使用检测到的周期(period={detected_period})进行STL分解...")

# 使用检测到的周期进行STL
stl = STL(time_series, period=detected_period, robust=True)
result = stl.fit()

# --- 6. 整合、输出并可视化 ---
output_df = pd.DataFrame({
    '原始自变量': df['自变量'],
    '原始因变量': df['因变量'],
    '趋势分量(Trend)': result.trend,
    '季节性分量(Seasonal)': result.seasonal,
    '残差分量(Residual)': result.resid
})

output_path = r"C:\Users\czw17\Desktop\附件2_STL分解结果_自动周期.xlsx"
output_df.to_excel(output_path, index=False)
print(f"分解完成！详细信息已保存到: {output_path}")

# 可视化
fig = result.plot()
plt.suptitle(f'STL Decomposition (Detected Period = {detected_period})', y=1.02)
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False
plt.tight_layout(rect=[0, 0, 1, 0.96])

plot_path = r"C:\Users\czw17\Desktop\附件1_STL分解图_自动周期.png"
plt.savefig(plot_path)
print(f"分解结果图已保存到: {plot_path}")
plt.show()
```

最终 boss.py

```
import numpy as np
import 光程差
import 计算折射率喵1
import 计算折射率喵2
from scipy.signal import find_peaks
file_path = r"C:\Users\czw17\Desktop\新输出结果喵.csv"
厚度 = 7.5#微米

def 周期函数(波数,厚度):

    波长 = 1/波数*10**4#微米
    周期 = np.abs(np.cos(np.pi*光程差.计算光程差(计算折射率喵1.计算折射率(波数,
        file_path),厚度,10)/波长))
    # print(波长) # 注释掉这行以减少输出
```

```
    return 周期

def 计算厚度周期关系(厚度值):
    """计算给定厚度下的周期信息"""
    波数 = np.linspace(1500,2700, 20000)
    周期 = 周期函数(波数, 厚度值)

    # 使用find_peaks找到所有峰值
    peaks, _ = find_peaks(周期, height=0)

    # 计算相邻峰值波数的差值作为周期
    if len(peaks) > 1:
        峰值波数 = 波数[peaks]
        周期差值 = np.diff(峰值波数)
        平均周期 = np.mean(np.abs(周期差值))
        #print(f'厚度 {厚度值} 对应的平均周期: {平均周期:.2f}')
        return 平均周期
    else:
        print(f'厚度 {厚度值} 下未找到足够的峰值')
        return None
#绘制周期函数
import matplotlib.pyplot as plt
波数 = np.linspace(1500, 4000, 20000)
周期 = 周期函数(波数,厚度)

# plt.plot(波数, 周期)
# plt.xlabel('波数')
# plt.ylabel('周期')
# plt.title('周期函数')

# 使用find_peaks找到所有峰值
peaks, _ = find_peaks(周期, height=0) # 可以根据需要调整height参数来过滤较小的峰值

# 在图上标注峰值点
plt.plot(波数[peaks], 周期[peaks], 'ro', markersize=4, label='Peaks')

# 输出所有峰值信息
print("所有峰值信息: ")
for i, peak_index in enumerate(peaks):
    print(f'峰值 {i+1}: 周期值 = {周期[peak_index]:.4f}, 对应波数 = {波数[peak_index]:.2f}')

# 计算并输出相邻峰值波数差值
if len(peaks) > 1:
    峰值波数 = 波数[peaks]
    周期差值 = np.diff(峰值波数)
    print("\n相邻峰值波数差值（周期）: ")
    for i, diff in enumerate(周期差值):
```

```
        print(f'周期 {i+1}: {diff:.2f}')

# 找到最大值及其对应的波数
max_value = np.max(周期)
max_index = np.argmax(周期)
max_wavenumber = 波数[max_index]

print(f'\n最大值: {max_value}')
print(f'对应的波数: {max_wavenumber}')

plt.legend()
plt.show()

# 调用新函数计算厚度与周期的关系
print("\n计算厚度与周期的关系: ")
计算厚度周期关系(厚度)
```

boss 二阶段.py

```
import 最终大boss
import numpy as np
目标周期=251.64
def 寻找目标周期厚度(目标周期, 厚度范围=(1, 20), 精度=0.01, 最大迭代次数=1000):
    """
    使用二分法寻找使周期为目标值的厚度

    参数:
    目标周期: 目标周期值
    厚度范围: 搜索厚度的范围 (最小值, 最大值)
    精度: 计算精度
    最大迭代次数: 最大迭代次数

    返回:
    找到的厚度值
    """
    左边界, 右边界 = 厚度范围

    for i in range(最大迭代次数):
        中点 = (左边界 + 右边界) / 2
        当前周期 = 最终大boss.计算厚度周期关系(中点)

        if 当前周期 is None:
            print(f"在厚度 {中点} 处无法计算周期")
            return None

        周期差 = 当前周期 - 目标周期
```

```
        #print(f"迭代 {i+1}: 厚度 = {中点:.4f}, 周期 = {当前周期:.4f}, 差值 = {周期差:.4f}")

        if abs(周期差) < 精度:
            print(f"找到满足条件的厚度: {中点:.4f}")
            return 中点

        if 周期差 > 0:
            左边界 = 中点
        else:
            右边界 = 中点

    print("未在指定迭代次数内找到满足精度要求的解")
    return (左边界 + 右边界) / 2

# 寻找使周期为250.1446的厚度
目标厚度 = 寻找目标周期厚度(目标周期)
print(f"\n使周期为{目标周期}的厚度为: {目标厚度:.4f} 微米")
```

菲涅尔定律计算折射率.py

```
import pandas as pd
import numpy as np
from scipy.optimize import root_scalar
import matplotlib.pyplot as plt

plt.rcParams['font.family'] = 'SimHei'
def 读取excel(file_path):
    """
    读取 CSV 文件的前两列数据
    """
    df = pd.read_csv(file_path, usecols=[0, 1])
    return df

def calculate_refractive_index(reflectance, angle_deg=10, n1=1.0):
    """
    使用菲涅尔公式计算折射率。
    :param reflectance: 0-1 之间的小数
    """
    if not 0 < reflectance < 1:
        raise ValueError("反射率必须在 (0, 1) 范围内。")
    if not 0 <= angle_deg < 90:
        raise ValueError("入射角必须在 [0, 90) 范围内。")

    theta_i = np.deg2rad(angle_deg)

    def eq(n2, R_meas, n1, theta_i):
        if n2 <= n1 * np.sin(theta_i):
```

```
            return 1e9
        cos_t = np.sqrt(1 - (n1 * np.sin(theta_i) / n2)**2)
        cos_i = np.cos(theta_i)
        Rs = ((n1 * cos_i - n2 * cos_t) / (n1 * cos_i + n2 * cos_t))**2
        Rp = ((n2 * cos_i - n1 * cos_t) / (n2 * cos_i + n1 * cos_t))**2
        return (Rs + Rp) / 2 - R_meas

    sol = root_scalar(eq,
                      args=(reflectance, n1, theta_i),
                      bracket=[1.0, 4.0],
                      method='brentq')
    if sol.converged:
        return sol.root
    else:
        raise ValueError("无法找到收敛解，请检查输入值。")

def process_and_save(input_path, output_path, angle_deg=10, n1=1.0):
    """
    读取 CSV，计算折射率，输出调试信息，保存新文件。
    """
    df = 读取excel(input_path)
    refractive_indices = []

    for i, R_percent in enumerate(df.iloc[:, 1]):
        R = R_percent / 100.0 # 百分制转小数
        try:
            n2 = calculate_refractive_index(R, angle_deg, n1)
        except Exception as e:
            n2 = np.nan
            print(f"第{i}行 转换后 R={R:.4f} 计算出错: {e}")
        refractive_indices.append(n2)

        # 调试输出前5行
        if i < 5:
            print(f"第{i}行: 原始R%={R_percent} → R={R:.4f}, 折射率={n2}")

    df['RefractiveIndex'] = refractive_indices
    df.to_csv(output_path, index=False)

    # 绘制折射率随反射率变化的图表
    plt.figure(figsize=(10, 6))
    plt.plot(df.iloc[:, 0], df['RefractiveIndex'], marker='.', linestyle='-', color="#2f9db3")

    plt.title('折射率随波数变化')
    plt.xlabel('波数 (/cm)')
    plt.ylabel('折射率')
    plt.grid(True)
```

```
    plt.tight_layout()

    # 保存图表
    plot_path = output_path.replace('.csv', '.png')
    plt.savefig(plot_path)
    plt.show()

    return df

if __name__ == "__main__":
    输入文件 = r"C:\Users\czw17\Desktop\新反射率处理数据.csv"
    输出文件 = r"C:\Users\czw17\Desktop\新输出结果喵.csv"
    result_df = process_and_save(输入文件, 输出文件, angle_deg=10, n1=1.0)
    print("处理完成，结果预览：")
    print(result_df.head())
```

多波束数据处理 copy.py

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import find_peaks
from scipy.optimize import curve_fit
from scipy.ndimage import gaussian_filter1d
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False
# ========== 1. 加载你的数据 ==========
input_path = r"C:\Users\czw17\Desktop\附件3.csv"

try:
    df = pd.read_csv(input_path, header=0)
    df.columns = ['自变量', '因变量']
    time_series = df['因变量'].values # 转为numpy数组便于计算
    x = df['自变量'].values # 使用实际的波数作为x轴
    print(f" 数据加载成功，共 {len(time_series)} 个点")
except FileNotFoundError:
    print(f" 错误：找不到文件，请检查路径是否正确：{input_path}")
    exit()
except Exception as e:
    print(f" 数据加载出错：{e}")
    exit()

# ========== 2. 提取趋势 —— 高斯滤波器 ==========
# 使用高斯滤波器进行平滑处理，sigma值控制平滑程度
sigma = len(time_series) // 5 # 根据数据长度自动调整sigma值
sigma = max(1, sigma) # 确保sigma至少为1
```

```
try:
    trend_est = gaussian_filter1d(time_series, sigma=sigma)
except Exception as e:
    print(f" 高斯滤波失败: {e}")
    exit()

disturbance_est = time_series - trend_est

# ========== 2.1 对扰动项进行高斯滤波 ==========
# 使用高斯滤波器进一步平滑扰动项，sigma值控制平滑程度
sigma_dist = max(1, len(time_series) // 1000) # 根据数据长度自动调整sigma值

try:
    disturbance_est = gaussian_filter1d(disturbance_est, sigma=sigma_dist)
except Exception as e:
    print(f" 扰动项高斯滤波失败: {e}")
    exit()

# ========== 3. 可视化原始信号、趋势、扰动 ==========
plt.figure(figsize=(14, 10))

plt.subplot(3, 1, 1)
plt.plot(x, time_series, label='原始信号', color='blue', alpha=0.8)
plt.title('原始信号')
plt.grid(True, alpha=0.3)
plt.legend()

plt.subplot(3, 1, 2)
plt.plot(x, time_series, label='原始信号', color='blue', alpha=0.5)
plt.plot(x, trend_est, 'r-', linewidth=2, label='估计趋势', alpha=0.9)
plt.title('趋势项提取')
plt.grid(True, alpha=0.3)
plt.legend()

plt.subplot(3, 1, 3)
plt.plot(x, disturbance_est, 'g-', label='估计扰动', alpha=0.8)
plt.axhline(0, color='black', linestyle='--', linewidth=0.8, alpha=0.6)
plt.title('扰动项（周期性尖峰）')
plt.grid(True, alpha=0.3)
plt.legend()

plt.tight_layout()
plt.show()

# ========== 4. 检测扰动周期（通过波谷） ==========
# 设置合理的波谷检测参数
# 通过检测负值来找到波谷
```

```
valley_height_threshold = np.mean(disturbance_est) - 0.05 * np.std(disturbance_est) #
    低于均值-0.5标准差
distance = max(10, len(time_series) // 50) # 波谷最小间隔，避免过密

valleys, properties = find_peaks(-disturbance_est, height=-valley_height_threshold,
    distance=distance)

print(f"\n 检测到 {len(valleys)} 个扰动波谷")

# 打印每个波谷对应的波数
for i, valley_index in enumerate(valleys):
    print(f"第 {i+1} 个波谷位置: {x[valley_index]:.2f} 波数")

if len(valleys) > 1:
    # 计算波数单位的周期
    periods = np.diff(x[valleys])
    avg_period = np.mean(periods)
    std_period = np.std(periods)
    print(f" 平均周期: {avg_period:.2f} 波数单位 (cm ¹)")
    print(f" 周期标准差: {std_period:.2f} → {'稳定' if std_period < avg_period*0.3 else
        '波动较大'}")

    # 创建表格数据
    valley_data = []
    for i, valley_index in enumerate(valleys):
        data_row = {
            '波谷序号': i + 1,
            '索引位置': valley_index,
            '波数位置': x[valley_index]
        }

        # 添加周期信息（除了最后一个波谷）
        if i < len(periods):
            data_row['到下一波谷周期'] = periods[i]

        valley_data.append(data_row)

    # 转换为DataFrame
    valley_df = pd.DataFrame(valley_data)

    # 保存到CSV文件
    output_path = "扰动周期数据.csv"
    valley_df.to_csv(output_path, index=False, encoding='utf-8-sig')
    print(f"\n 扰动周期数据已保存至: {output_path}")
    print("\n 扰动周期数据表格:")
    print(valley_df.to_string(index=False))
```

```
    # 绘制扰动与波谷
    plt.figure(figsize=(12, 5))
    plt.plot(x, disturbance_est, 'g-', label='扰动信号', alpha=0.7)
    plt.plot(x[valleys], disturbance_est[valleys], "rx", markersize=8, label='检测到的波谷')
    plt.title('扰动信号中的周期性波谷')
    plt.xlabel('波数 (cm⁻¹)')
    plt.ylabel('扰动幅值')
    plt.legend()
    plt.grid(True, alpha=0.3)
    plt.show()
else:
    print(" 未检测到足够峰值，可能扰动不明显或参数需调整")

# ========== 5. （可选）尝试拟合扰动形状：(ax+f) / (cos(b*x + c) + d) + e ==========
def pulse_model(x, a, f, b, c, d, e):
    # 避免分母接近0导致爆炸 → clip分母下限
    denominator = np.cos(b * x + c) + d
    denominator = np.clip(denominator, 0.1, None) # 防止除零
    return -(a * x + f) / denominator + e

if len(valleys) >= 3:
    try:
        # 初始参数猜测
        a_guess = -0.2 # 对于(ax+f)形式，a的初始值设为0
        f_guess = np.min(disturbance_est[valleys]) - np.mean(disturbance_est)
        b_guess = 2 * np.pi / avg_period # 根据平均周期估算角频率
        c_guess = 0.0
        d_guess = 1.1 # 避免分母为0
        e_guess = np.mean(disturbance_est)

        p0 = [a_guess, f_guess, b_guess, c_guess, d_guess, e_guess]

        # 局部拟合：只拟合前几个周期提高稳定性
        # 使用波数单位计算拟合范围
        fit_range_points = min(len(x), int(avg_period * 5 / np.mean(np.diff(x)))) # 拟合前5个周期
        popt, pcov = curve_fit(
            pulse_model,
            x[:fit_range_points],
            disturbance_est[:fit_range_points],
            p0=p0,
            maxfev=10000,
            bounds=([-10, -10, 0, -np.pi, 0.5, -np.inf], [-0.1, np.inf, 2*np.pi, np.pi, 3.0,
                np.inf])
        )

        fitted_disturbance = pulse_model(x, *popt)
        residuals = disturbance_est - fitted_disturbance
```

```
        rmse = np.sqrt(np.mean(residuals**2))
        print(f"\n 扰动形状拟合成功！")
        param_names = ['a (斜率)', 'f (截距)', 'b (角频率)', 'c (相位)', 'd (偏移)', 'e
            (垂直偏移)']
        for name, val in zip(param_names, popt):
            print(f" {name}: {val:.4f}")
        print(f" 拟合误差 (RMSE): {rmse:.4f}")

        # 绘图对比
        plt.figure(figsize=(12, 5))
        plt.plot(x, disturbance_est, 'g-', alpha=0.6, label='实际扰动')
        plt.plot(x, fitted_disturbance, 'r--', linewidth=2, label='拟合模型')
        plt.title('扰动形状拟合结果对比')
        plt.xlabel('波数 (cm⁻¹)')
        plt.legend()
        plt.grid(True, alpha=0.3)
        plt.show()

    except Exception as e:
        print(f"\n 形状拟合失败: {e}")
        print(" 可能原因: 噪声大、形状不符、初值不准。可尝试手动调整 p0 或拟合区间。")

# ========== 6. 输出扰动信号到高分辨率表格 ==========
# 创建包含波数和扰动值的数据框
high_res_data = pd.DataFrame({
    '波数 (cm⁻¹)': x,
    '扰动值': disturbance_est
})

# 保存到CSV文件
output_path_high_res = "扰动信号数据.csv"
high_res_data.to_csv(output_path_high_res, index=False, encoding='utf-8-sig')
print(f"\n 扰动信号数据已保存至: {output_path_high_res}")
print("\n 扰动信号数据表格 (前10行):")
print(high_res_data.head(10).to_string(index=False))
```

求导图象.py

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.signal import savgol_filter

plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

input_path = r"C:\Users\czw17\Desktop\扰动信号数据2.csv"
```

```
try:
    df = pd.read_csv(input_path, header = 0)#有headers
    df.columns = ['自变量', '因变量']
    time_series = df['因变量'].values
    x_values = df['自变量'].values
except FileNotFoundError:
    print(f"错误：找不到文件，请检查路径是否正确：{input_path}")
    exit()

# 数据平滑处理
window_length = 51 # 窗口长度，必须为奇数
polyorder = 3    # 多项式阶数
smoothed_data = savgol_filter(time_series, window_length, polyorder)

# 计算一阶导数
first_derivative = savgol_filter(time_series, window_length, polyorder, deriv=1, delta=1.0)

# 计算二阶导数
second_derivative = savgol_filter(time_series, window_length, polyorder, deriv=2, delta=1.0)

# 计算二阶导数关于x轴的对称性
# 对称性分析：计算二阶导数与其关于x轴对称的差异
second_derivative_symmetric = -second_derivative
# 计算对称性度量（均方根误差）
symmetry_metric = np.sqrt(np.mean((second_derivative - second_derivative_symmetric)**2))

# 可视化结果
plt.figure(figsize=(12, 10))

plt.subplot(4, 1, 1)
plt.plot(x_values, time_series, label='原始数据')
plt.plot(x_values, smoothed_data, label='平滑数据')
plt.legend()
plt.title('原始数据与平滑数据')

plt.subplot(4, 1, 2)
plt.plot(x_values, second_derivative, label='二阶导数', color='green')
plt.legend()
plt.title('二阶导数')

plt.legend()
plt.title(f'二阶导数及其关于x轴的对称性（对称性度量：{symmetry_metric:.4f}）')

plt.tight_layout()
plt.show()
```

```
# 输出对称性分析结果
print(f"二阶导数关于x轴的对称性度量: {symmetry_metric:.4f}")
if symmetry_metric < 0.1:
    print("二阶导数具有良好的关于x轴的对称性")
elif symmetry_metric < 0.5:
    print("二阶导数具有中等程度的关于x轴的对称性")
else:
    print("二阶导数关于x轴的对称性较差")
```