碳化硅与硅晶圆片干涉测厚模型的研究

摘要

本研究针对碳化硅（SiC）和硅（Si）晶圆片外延层厚度测量问题，结合实验光谱数据，系统分析了外延层厚度反演方法及多光束干涉的影响，并提出相应的修正策略。

针对问题 1，针对问题一，建立了基于单光束干涉的厚度模型。通过波程差产生的干涉极大与极小关系推导出厚度 d 与波数 σ、外延层折射率 n_1 及入射角 θ_i 的定量关系式并采用 Sellmeier 色散关系描述折射率随波长的变化。

针对问题 2，基于问题一的厚度模型，设计了分步厚度反演算法。首先对数据进行预处理并寻峰，根据问题一中给出的反射率极大值与厚度的关系式，并假设折射率 n 不随波长变化，得到厚度的初步估计值，然后以该初值为起点，通过最小二乘拟合进一步优化厚度 d，并结合不确定度计算，评估厚度估计的稳定性，得到 $d = 9.1451 \pm 0.0828\,\mu\text{m}$ ，验证了厚度计算的可靠性。

针对问题 3，建立了 Airy 无限反射多光束干涉模型，用于分析多光束干涉的影响。通过快速傅里叶变换（FFT）分析实验数据，显著的 FFT 峰值对应稳定的光程差干涉条纹，是多光束干涉存在的直接证据。利用 Airy 模型结合非线性最小二乘拟合，得到硅晶圆片厚度 $d_{\,3} = 4.8115 \pm 0.0010\,\mu\text{m}$ 和 $d_{\,4} = 4.8125 \pm 0.0010\,\mu\text{m}$，不确定度通过协方差矩阵分析，并结合 Bootstrap 检验了厚度估计的稳定性。

最后针对附件 1 和附件 2 的碳化硅晶圆片数据，FFT 检测表明存在多光束干涉。为消除多光束干涉对厚度计算的影响，建立了 FFT 滤波修正模型，采用平滑处理与 FFT 滤波去除干涉成分后，再利用单层 TMM 模型进行拟合，得到修正后的厚度为 $d_{\,1} = 9.2297 \pm 0.063\,\mu\text{m}$ 和 $d_{\,2} = 9.2667 \pm 0.068\,\mu\text{m}$。结果表明，该方法能够有效消除多光束干涉的影响，实现高精度厚度测量。

综上所述，本研究建立了系统的外延层厚度反演方法，设计了稳健的厚度计算算法，并提出多光束干涉消除策略，为碳化硅与硅晶圆片的精确厚度测量提供了可靠方法，也为薄膜光学特性分析提供了参考。

1 问题背景重述

1.1 问题背景

碳化硅作为一种新兴的第三代半导体材料，以其优越的综合性能表现正在受到越来越多的关注。碳化硅外延层的厚度是外延材料的关键参数之一，对器件性能有重要影响。因此，制定一套科学、准确、可靠的碳化硅外延层厚度测试标准显得尤为重要。

红外干涉法是一种无损测量外延层厚度的方法，其工作原理为：外延层与衬底由于掺杂载流子浓度不同而具有不同的折射率。红外光入射到外延层后，一部分光从外延层表面反射，另一部分光从衬底表面反射，两束光在一定条件下形成干涉条纹（图 1）。通过分析干涉光谱的波长、外延层折射率及红外光入射角等参数，可确定外延层的厚度。通常，外延层的折射率并非恒定，它与掺杂浓度及光谱波长等因素相关。

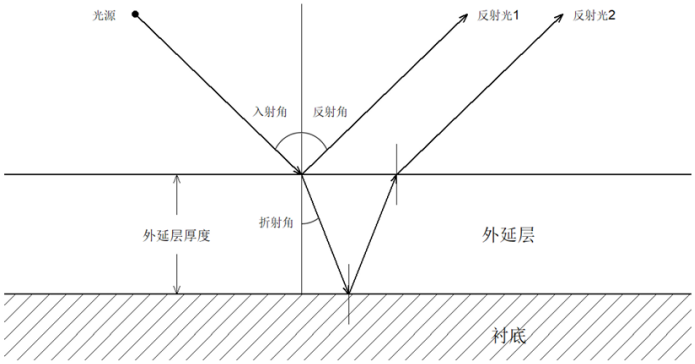


图 1: 外延层厚度测量原理示意图（单次反射干涉）

1.2 问题要求

问题一与问题二只考虑外延层与衬底界面仅发生一次反射与透射所产生的干涉条纹情形。问题一首先要求建立确定外延层厚度的数学模型。在此基础上，问题二要求设计相应的计算算法，对附件 1 和附件 2 提供的碳化硅晶圆片光谱实测数据进行分析与计算，并评价所得结果的可靠性。

进一步，问题三需要考虑光波在外延层与衬底界面产生多次反射和透射，从而形成多光束干涉的情况，需要推导多光束干涉产生的必要条件，并分析其对外延层厚度测量精度可能产生的影响。基于这些条件，应对附件 3 和附件 4 提供的硅晶圆片测试数据进行分析，判断是否存在多光束干涉，并建立相应的数学模型及计算算法以确定硅外延层厚度，同时给出计算结果。

此外，如果多光束干涉可能出现在碳化硅晶圆片的测试结果中，从而影响厚度计算

精度，应提出有效方法消除其影响，并提供修正后的厚度计算结果。整体要求强调模型建立、算法设计、实验数据处理以及结果可靠性分析的完整性和科学性。

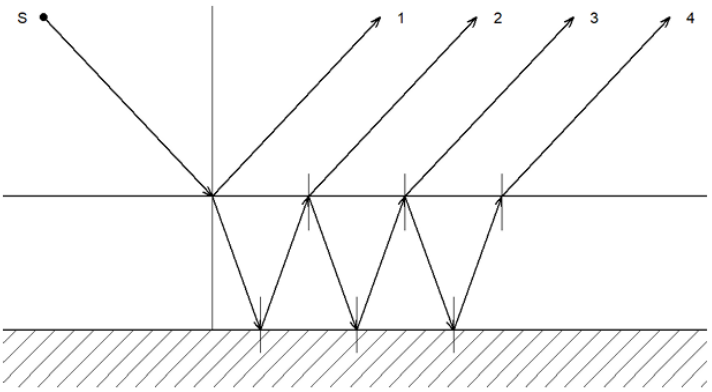


图 2: 多光束干涉示意图

2 问题分析

问题一：针对外延层厚度的确定，核心思想是利用干涉条纹形成的条件。光在外延层中传播时会产生光程差，不同波长下的相位差异导致了干涉现象。通过对比波数与干涉级数的线性关系，并结合色散公式对折射率的波长依赖性进行修正，从而建立厚度与干涉条纹之间的定量对应关系。该方法既能反映物理机理，又便于结合实际测量数据进行拟合与计算。

问题二：需要根据问题 1 建立的数学模型设计外延层厚度的反演算法。对于碳化硅晶圆片（附件 1 和附件 2），可以先结合实测光谱数据进行波峰提取，并假设外延层折射率为常量得到外延层厚度的估计值然后带入色散公式修正并最小二乘拟合，得到更精确的厚度值。为提高结果的可靠性，可进一步进行不确定度分析以评估实验噪声对拟合结果的影响。

问题三：需要首先推导无限次反射产生的反射率与光强公式，得出多光束干涉的必要条件。然后通过对附件 3 和附件 4 的硅晶圆片测试数据进行 FFT 分析，观察是否有明显的多光束频率成分，并采用无限反射 Airy 模型进行厚度反演。最后对于附件 1 和附件 2 的碳化硅晶圆片，同样通过 FFT 检测多光束成分，并通过 FFT 滤波以消除多光束干涉的影响，然后使用单层模型进行厚度拟合，以保证厚度估计的精度和稳健性。

3 模型假设

假设一：外延层材料为均匀各向同性介质，其折射率在同一波长下不随位置发生变化。

假设二：红外光为单色 s 偏振光，且光源具有足够的相干性，能够形成稳定的干涉条纹。

假设三：外延层与衬底界面为理想平面界面，界面粗糙度和缺陷对干涉结果的影响忽略不计。

假设四：外延层的折射率可以通过色散经验公式或实验数据进行准确描述。

假设五：空气的折射率可视作 1 处理。

假设六：衬底材质折射率远大于外延层材质折射率，即红外光在衬底界面发生全反射。

假设七：测量过程中环境因素（如温度、杂散光、外界振动等）对实验数据的影响可以忽略。

4 符号说明

符号	含义
n_0	空气折射率，视作 1
n_1	外延层折射率
n_2	衬底折射率
d	外延层厚度
λ	波长
σ	波数（cm^{-1}）
$\Delta\sigma$	相邻波峰之间的波数间隔
θ_i	入射角
θ_r	折射角
R	反射率（%）
m	干涉级数（1,2,3,…）
δ	光程差
$\Delta\varphi$	相位差
ϕ	半波损失相位差（$\pm\pi$）

5 问题一模型建立

在问题一中，我们仅考虑外延层与衬底界面发生一次反射和透射所产生的干涉条纹。此时，干涉的形成主要由两束光决定：一束来自外延层表面的反射光，另一束来自外延层与衬底界面的反射光。两束光之间存在光程差，当光程差满足一定条件时，便会产生干涉极大或极小。

本题的核心思路是：通过建立外延层厚度 d、入射角 θ、折射率 n 与波长 λ 之间的对应关系，利用光程差条件推导出干涉条纹的数学表达式。

5.1 光程差推导

在薄膜干涉问题中，影响两束主要反射光相位差的量有两部分：一是光在外延层往返传播产生的光程差，二是反射时可能出现的半波相移。下面给出一般性的数学推导。

设入射介质折射率为 n_0，外延层折射率为 $n_1(\lambda)$（受红外光谱波长影响），衬底折射率为 n_2。入射角为 θ，在外延层内的折射角由斯涅尔定律给出：

$$n_0 \sin\theta_i = n_1(\lambda) \sin\theta_r, \tag{1}$$

由于假设红外光在空气中的折射率为 1，因此可以写出

$$\cos\theta_r = \sqrt{1 - \left(\frac{1}{n_1(\lambda)} \sin\theta_i\right)^2}. \tag{2}$$

光在外延层中来回一次的光程差（optical path difference, OPD）为

$$\delta = \mathrm{OPD} = 2\, n_1(\lambda)\, d \cos\theta_r, \tag{3}$$

其中 d 为外延层厚度。

将光程差换算为相位差，得到（再加上由于反射引入的相位修正 ϕ）

$$\Delta\varphi \;=\; \frac{2\pi}{\lambda}\,\delta \;+\; \phi \;=\; \frac{4\pi n_1(\lambda) d \cos\theta_r}{\lambda} \;+\; \phi, \tag{4}$$

其中 ϕ 表示半波损失相位差项，常见取值为 0 或 π（或写作 $\pm\pi$，视具体反射界面而定）。

干涉极大（相长）或极小（相消）的条件可用相位差给出。一般地设第 m 个干涉阶

满足

$$\Delta\varphi = 2\pi m, \qquad m \in \mathbb{Z}, \tag{5}$$

代入上式并解出厚度 d 可得一般表达式

$$d \;=\; \frac{\big(m - \dfrac{\phi}{2\pi}\big)\,\lambda}{2\,n_1(\lambda)\cos\theta_r}. \tag{6}$$

由于实验数据以波数 σ（通常单位为 cm^{-1}）表示，利用 $\lambda = 1/\sigma$，上式等价地写为

$$d \;=\; \frac{m - \dfrac{\phi}{2\pi}}{2\,n_1(\sigma)\,\sigma\cos\theta_r}, \tag{7}$$

这里要求 d 与 σ 的单位一致（例如 d 以 cm 为单位时 σ 以 cm^{-1}）。

特殊情况说明：若无反射相位反转（$\phi = 0$），构成极大时满足 $2n_1(\sigma)d\cos\theta_r = m\lambda$；若存在一次半波反转（$\phi = \pi$），构成极大时满足 $2n_1(\sigma)d\cos\theta_r = (m + \frac{1}{2})\lambda$。

5.2 简化模型

然而在具体实现中，波峰的准确编号往往难以确定，且半波损失相位项 ϕ 的引入也会使运算复杂化。为了简化模型，可以采用一种粗略的近似方法：通过相邻干涉波峰之间的间距来估算外延层厚度。在此过程中，暂时将折射率 n_1 视为常数。根据文献报道，在红外波段范围内，碳化硅外延层的折射率可近似取 $n_1 \approx 2.6$ [1]。

根据干涉条件，相邻两条极大或极小条纹所对应的波数间隔 $\Delta\sigma$ 与厚度 d 近似满足：

$$d \approx \frac{1}{2n_1(\sigma)\cos\theta_r\,\Delta\sigma}. \tag{8}$$

这种方法不需要对每个波峰进行编号，也避免了相位修正项的引入，因而能够在实验数据中快速得到厚度的近似值，尽管精度有限，但可以为后续提供初始值猜测与合理性验证参考。

5.3 考虑色散模型

在对外延层折射率进行更为精确的建模时，需要引入色散模型。常见的色散经验公式包括 Cauchy 公式、Sellmeier 公式以及 Drude 模型等。下面对它们分别进行介绍与分

析。

Cauchy 公式

Cauchy 公式是一种经典的经验公式，其表达式为

$$n(\lambda) = A + \frac{B}{\lambda^2} + \frac{C}{\lambda^4}, \tag{9}$$

其中 A, B, C 为经验系数。Cauchy 公式在可见光波段具有较好的拟合效果，形式简洁，计算方便 [2]。然而，该公式在紫外和红外波段拟合精度较差，难以涵盖宽光谱范围。

Sellmeier 公式

Sellmeier 公式是一种物理基础较强的色散关系，通常写为

$$n^2(\lambda) = 1 + \sum_{j=1}^{m} \frac{B_j \lambda^2}{\lambda^2 - C_j}, \tag{10}$$

其中 B_j, C_j 为经验拟合常数。Sellmeier 公式能够在较宽波长范围内保持较好的精度，且与材料的物理吸收峰相关联，因此更适合用于描述半导体及晶体材料 [2]。

Drude 模型

Drude 模型从自由电子理论出发，常用于金属和高度掺杂半导体的色散关系。其表达式为

$$n^2(\omega) = \epsilon(\omega) = \epsilon_\infty - \frac{\omega_p^2}{\omega^2 + i\gamma\omega}, \tag{11}$$

其中 ϵ_∞ 表示高频介电常数，ω_p 为等离子体频率，γ 为阻尼因子 [2]。Drude 模型能够较好地描述自由载流子对折射率的影响，尤其适用于红外波段。但其对绝缘体和弱掺杂半导体的拟合效果有限。

模型选择

综合比较，Cauchy 公式简单但适用范围有限，Drude 模型更适合描述自由电子气体或金属行为，而 Sellmeier 公式兼具物理解释性与宽波段高精度的优点。因此，本模型采用 Sellmeier 色散公式作为对外延层折射率的描述基础。

5.4 Sellmeier 模型的引入与拟合思路

在前述简化模型中，我们假设折射率 n 在波数范围内为常数，并通过相邻干涉极值间距来近似估算厚度 d。然而，实际材料的折射率随波长（或波数）变化显著，若忽略这一色散效应，则模型在宽光谱范围内会产生系统性偏差。为提高厚度估计精度，本部分引入 Sellmeier 色散模型，其经验公式为

$$n^2(\lambda) = 1 + \frac{5.5394\,\lambda^2}{\lambda^2 - 0.026945}, \tag{12}$$

其中 λ 表示波长（单位：μm）。该公式来源于对碳化硅折射率实验数据的拟合 [1]。

具体建模思路

1. 数据准备：从实验数据中提取干涉峰值波数（$\sigma = 1/\lambda$），并分别对应两个不同入射角 θ_1 和 θ_2。

2. 初始估计：利用相邻峰间距的中位数，根据简化模型

$$d \approx \frac{1}{2n_1(\sigma)\cos\theta_r\,\Delta\sigma}.$$

得到 d 的粗略估计值作为初始猜测值，并以 Sellmeier 公式计算折射率初始值，进而估计外延层厚度的初始猜测。

3. 阶次编号：对于每个波峰，尝试枚举合理范围内的干涉阶次 m，保证不同入射角下的数据具有一致性。

4. 非线性拟合：构造残差函数

$$R = 2\,n_1(\sigma)\,d\cos\theta_r\,\sigma - m,$$

并采用最小二乘优化方法，在给定搜索区间内寻找最优厚度 d 与阶次编号 m，使整体残差最小。

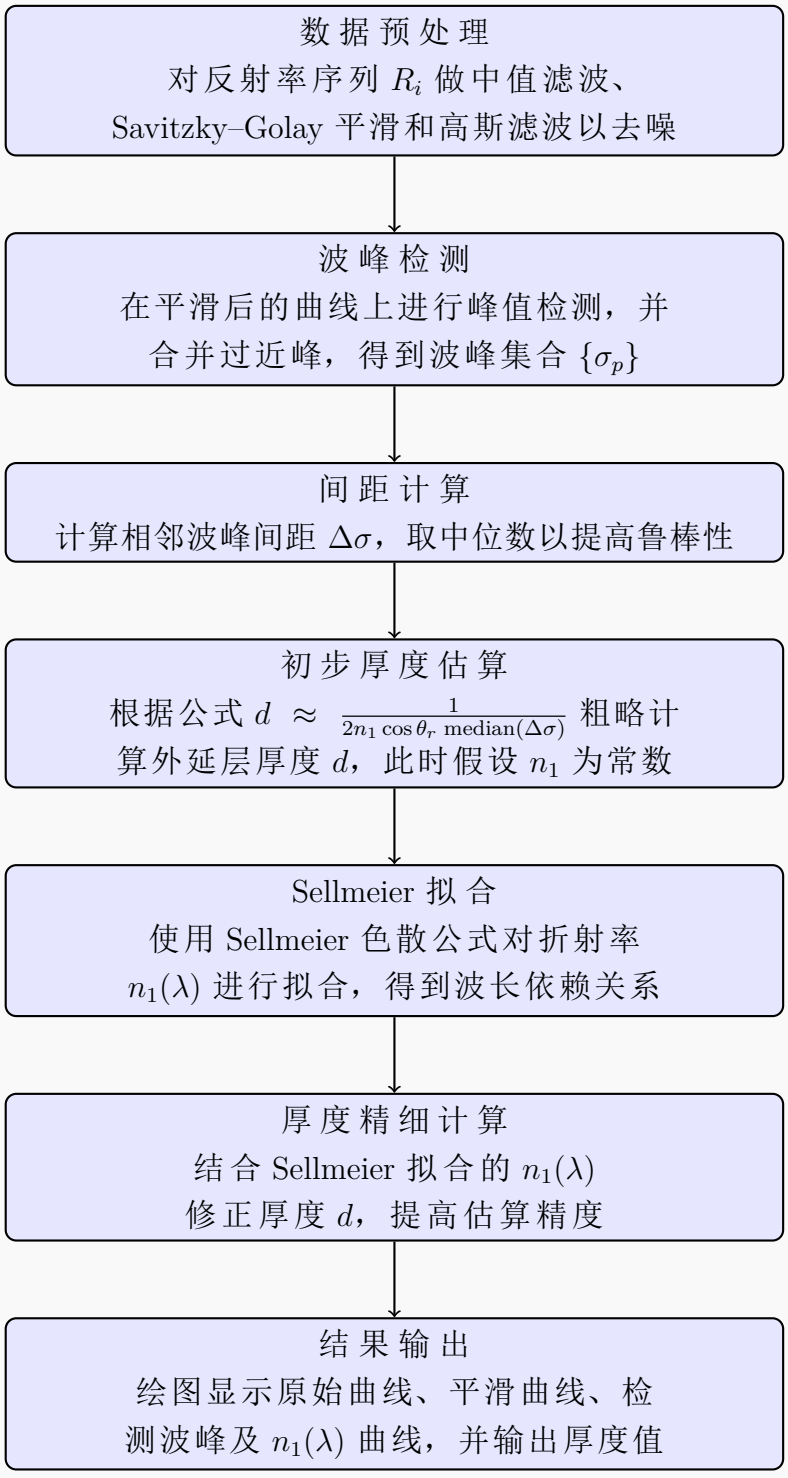
5. 拟合结果：通过最优参数计算出的 $2n_1(\sigma)d\cos\theta\sigma$ 与整数阶次 m 进行对比，若两者在散点图中接近对角线，则说明 Sellmeier 模型对实验数据的拟合较为理想。

该方法的优点在于：一方面保留了原简化模型基于干涉条纹间距的直观物理解释，另一方面通过 Sellmeier 模型引入色散修正，从而在保证可实现性的同时提高厚度反演的精度。

6 问题二模型求解与可靠性分析

6.1 算法设计思路

外延层厚度计算算法



6.2 初步厚度估算结果

根据前述简化模型，利用附件 1 和附件 2 的波峰检测结果，可得到相邻波峰间距的中位数如下：

- 附件 1 波峰间距中位数：$219.122cm^{-1}$
- 附件 2 波峰间距中位数：$210.684cm^{-1}$

结合公式

$$d \approx \frac{1}{2n_1 \cos\theta_r \cdot \text{median}(\Delta\sigma)}$$

并取折射率 n_1 的经验值 $= 2.6$ [1]，可直接计算出外延层的初步厚度 $d = 8.969\mu m$

此外，两张寻峰结果的可视化图已保存，可用于验证波峰检测的准确性，如图所示。

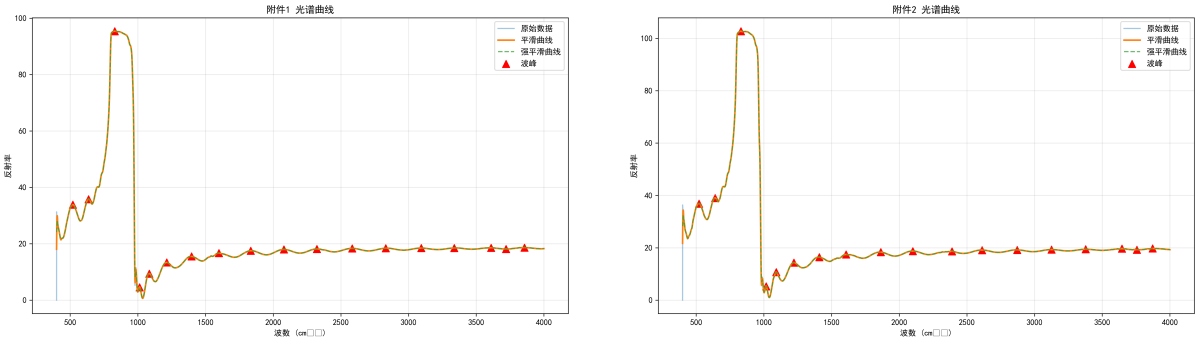


图 3: 附件 1 与附件 2 波峰检测结果示意图

6.3 考虑 Sellmeier 色散公式的拟合结果

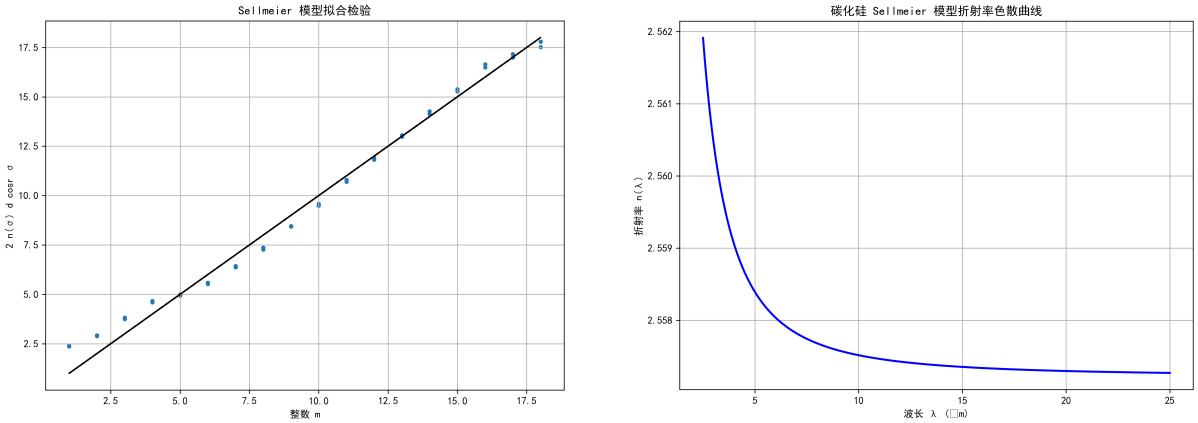
在引入 Sellmeier 色散模型后，我们将附件 1 与附件 2 的数据结合，对外延层厚度和折射率随波长的变化进行了拟合。拟合结果如下：

- 波峰编号 $m_0(10°) = 10$，$m_0(15°) = 10$
- 外延层厚度 $d = 9.1451\,\mu\text{m}$
- 拟合损失值 $\text{loss} = 1.2057 \times 10^1$

6.4 可靠性分析

6.4.1 模型与结果合理性分析

首先，从色散曲线中折射率-波长关系平滑且符合碳化硅材料的典型色散特性，说明 Sellmeier 模型能够合理反映材料实际的光学性质。其次，考虑色散效应后计算得到的厚度 $d = 9.1451\ \mu\text{m}$ 与简化模型估算值接近，其数量级符合外延层厚度的预期范围，体现了方法在量级上的合理性。再次，Sellmeier 模型的拟合误差（loss）为 12.057，表明拟合效果良好，误差处于可接受范围内。此外，拟合曲线中 $2\cos r \cdot \sigma$ 与干涉级数 m 的关系近似过原点的直线，进一步验证了波峰检测和厚度计算方法的一致性与自洽性。



(a) Sellmeier 模型拟合检验图　　(b) 折射率-波长色散曲线

图 4: Sellmeier 模型相关结果

6.4.2 不确定度分析

为了量化外延层厚度估算的不确定度，本研究依据简化模型和 Sellmeier 模型的计算方法进行分析。假设测量波峰位置的标准偏差为 $\Delta\sigma$，则厚度 d 的不确定度 Δd 可通过误差传播公式计算：

$$d = \frac{1}{2n\cos\theta_r\ \Delta\sigma} \quad \Rightarrow \quad \Delta d = \left|\frac{\partial d}{\partial \sigma}\right| \Delta\sigma = \frac{\Delta\sigma}{2n\cos\theta_r\ (\Delta\sigma)^2} = \frac{d}{\Delta\sigma} \cdot \Delta\sigma \tag{13}$$

在 Sellmeier 模型下，由于考虑了色散效应，厚度的不确定度可以类似处理：

$$d = \frac{m}{2n(\lambda)\cos\theta_r\ \sigma_m}, \quad \Delta d = \sqrt{\sum_m \left(\frac{\partial d}{\partial \sigma_m}\Delta\sigma_m\right)^2} \tag{14}$$

将实际测量数据代入，得到 Sellmeier 模型估算的外延层厚度为

$$d = 9.1451 \pm 0.0828\ \mu\text{m}$$

该不确定度结果表明，厚度估算具有较高精度，误差在可接受范围内，验证了所采用方法的可靠性与可重复性。

7 问题三模型建立与求解

在外延层厚度测量中，光波在外延层界面与衬底界面之间可能发生多次反射和透射。这种多光束反射-透射会产生多光束干涉效应，其干涉条件和干涉强度与材料折射率、外延层厚度及入射光波波长密切相关。

7.1 多光束干涉的理论推导

反射率推导

当光波入射到多层膜结构（如空气/外延层/衬底）时，会在各界面发生反射和透射，形成多光束干涉。假设红外光在空气-外延层表面以及外延层-衬底表面的折射角分别为 θ_r, θ_r'，则根据斯涅尔定律有

$$n_0 \sin\theta_i = n_1 \sin\theta_r = n_2 \sin\theta_r' \tag{15}$$

对于 s 偏振光，其在界面 $i \to j$ 的菲涅尔反射系数为

$$r_{ij}^{(s)} = \frac{n_i\cos\theta_i - n_j\cos\theta_j}{n_i\cos\theta_i + n_j\cos\theta_j} \tag{16}$$

在本问题中可得

$$r_1 = r_{01}^{(s)} = \frac{n_0\cos\theta_i - n_1\cos\theta_r}{n_0\cos\theta_i + n_1\cos\theta_r}, \quad r_2 = r_{12}^{(s)} = \frac{n_1\cos\theta_r - n_2\cos\theta_r'}{n_1\cos\theta_r + n_2\cos\theta_r'} \tag{17}$$

外延层厚度引起的相位延迟为

$$\delta = \frac{2\pi}{\lambda}\cdot 2n_1 d\cos\theta_r \tag{18}$$

由此可以推出 Airy 公式。各次反射光的叠加可以表示为几何级数：

$$E_r = r_1 + t_1 r_2 t_1 e^{i\delta} + t_1 r_2 r_1 r_2 t_1 e^{i2\delta} + \cdots,$$

其中 t_1 为透射系数。对几何级数求和可得等效反射系数：

$$r_{\text{eff}} = \frac{r_1 + r_2 e^{i\delta}}{1 + r_1 r_2 e^{i\delta}}. \tag{19}$$

因此，总反射率为

$$R = |r_{\text{eff}}|^2 = \frac{r_1^2 + r_2^2 + 2r_1 r_2\cos\delta}{1 + (r_1 r_2)^2 + 2r_1 r_2\cos\delta}. \tag{20}$$

光强推导

若考虑透射光强，则 Airy 公式表明：

$$I = |E_r|^2 = \frac{I_0}{1 + F\sin^2(\delta/2)}, \tag{21}$$

其中 I_0 为入射光强，Finesse 系数定义为

$$F = \frac{4R}{(1-R)^2}. \tag{22}$$

7.2 多光束干涉的必要条件

根据前述推导，外延层中是否会出现显著的多光束干涉主要取决于以下几个条件：

1. **界面反射幅度条件**：若界面复合反射系数

$$Q = |r_1 r_2|$$

较小（如 $Q \ll 0.1$），则高阶反射迅速衰减，近似二束干涉即可；当 $Q \gtrsim 0.1 \sim 0.2$ 时，高阶项不可忽略，多光束效应显著。

2. **相干长度条件**：光源的相干长度 L_c 必须大于外延层往返光程差

$$\mathrm{OPD} = 2n_1 d \cos\theta_r,$$

否则不同次反射之间无法保持相干，条纹会被平均化。

3. **谱线宽度与 Finesse 条件**：由 Airy 公式可知，系统的谱线半高宽 $\Delta\delta$ 与复合反射率 $R \approx |Q|$ 相关，Finesse 定义为

$$F = \frac{4R}{(1-R)^2}.$$

当 F 较大时，理论谱峰非常尖锐。若实验中观测到的谱线宽度 $\Delta\sigma_{\mathrm{obs}}$ 与理论计算的 $\Delta\sigma_{\mathrm{theo}}$ 在同一数量级，则说明多光束干涉模型与实际一致。

7.3 多光束干涉对厚度计算精度的影响

由式 (20) 与式 (21) 可见，多光束干涉效应显著增强了干涉条纹的对比度，使得反射率与相位延迟 δ 之间的关系更加敏感。其主要影响体现在以下几个方面。

首先，当反射系数 r_1 与 r_2 较大时，干涉项在分母中产生增强效应，导致反射率曲线出现尖锐的极小值与极大值，从而提高了条纹定位的精确度，有助于厚度 d 的高精度估算。

其次，Airy 公式 (21) 显示干涉峰值位置由相位延迟 $\delta = \frac{4\pi n_1 d\cos\theta_r}{\lambda}$ 决定，因而在实际测量中，任何波长 λ、折射率 n_1 或入射角 θ_1 的微小误差，都会在 δ 中被放大，进而引入厚度 d 的系统偏差。

再次，多光束干涉具有极高的谱分辨率，但同时对光源相干性与样品表面质量敏感。若表面粗糙或存在吸收损耗，则等效反射系数 (19) 将偏离理想情况，使条纹对比度下降，厚度反演的可靠性降低。不过由于本文假设界面理想平整且无吸收损耗，此影响在当前模型中可忽略。

综上所述，多光束干涉一方面通过增强条纹清晰度和信噪比提高了厚度测量的分辨

率，另一方面也放大了实验条件和参数测量误差对外延层厚度计算精度的影响。因此，在利用干涉方法计算厚度时，需要综合考虑条纹分辨率、光源相干性以及实验噪声，以确保结果的可靠性。

7.4 多光束干涉的存在性判定算法设计

在前文中，我们基于 Airy 公式推导了多光束干涉的条件及其对反射率分布的影响。为了在实际实验数据中判断多光束干涉是否存在，并进一步提高外延层厚度计算的精度，我们提出了如下判定与计算算法。整体流程如图 5 所示，并在算法 1 中给出了伪代码描述。

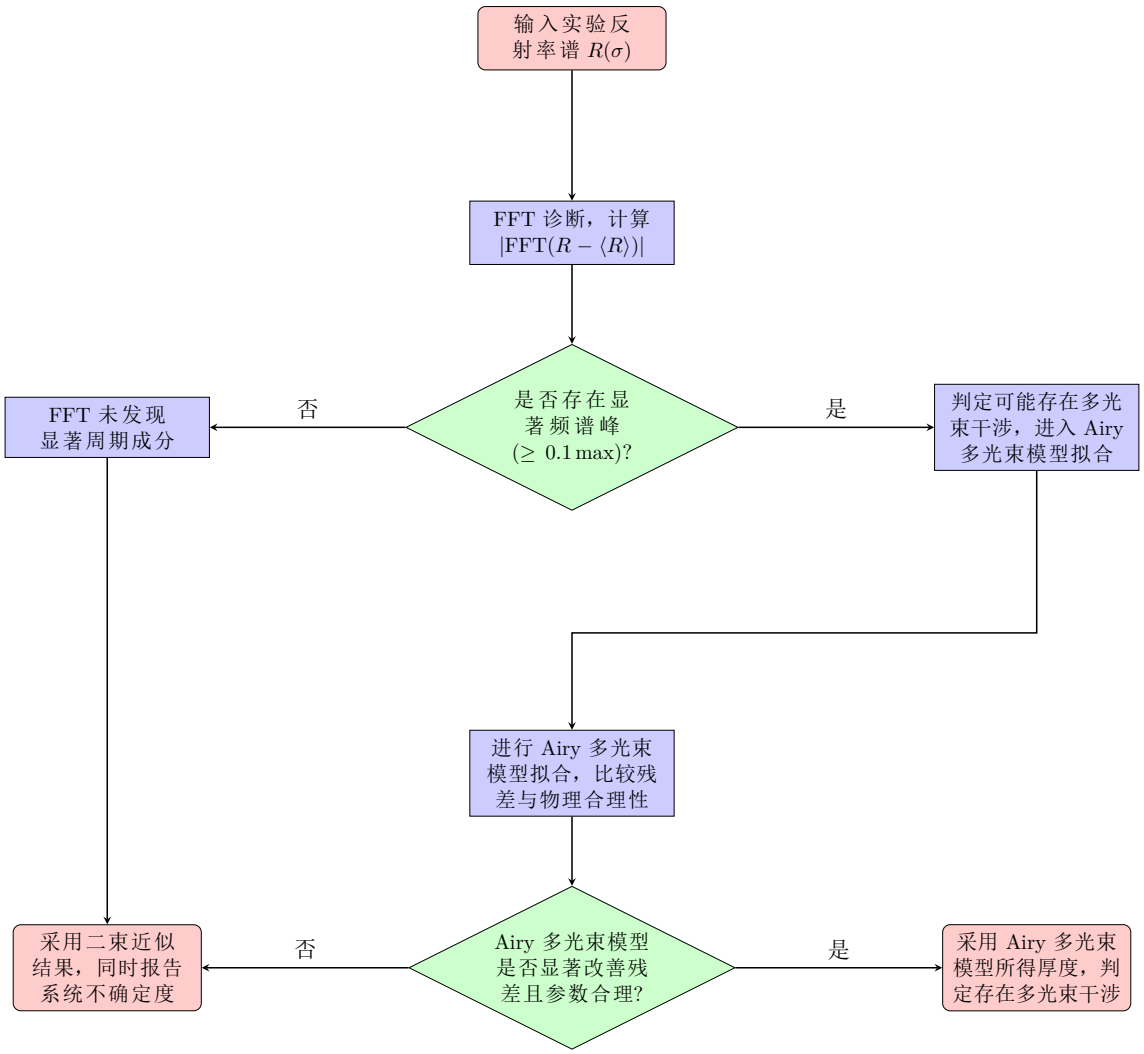


图 5: 多光束干涉存在性的判定流程图

Algorithm 1: 多光束干涉判定与厚度计算算法

Input: 实验反射率谱 $R(\sigma)$, 折射率初值 n_j, 厚度初值 d_j
Output: 外延层厚度估计 d, 干涉判定结果
对 $R(\sigma)$ 进行平滑与去均值处理;
计算其 FFT 幅值谱 $F(f) = |\mathrm{FFT}(R - \langle R \rangle)|$;
if *存在* $F(f)$ *峰值* $\geq 0.1 \times \max(F(f))$ **then**
    标记为“可能存在多光束干涉”;
构建单层 Airy 多光束模型;
利用最小二乘方法拟合 $R_{\mathrm{model}}(\sigma)$ 至 $R(\sigma)$;
计算残差平方和 χ^2;
if *Airy 多光束模型拟合显著降低* χ^2 *且参数合理* **then**
    输出 Airy 多光束模型厚度结果 d，判定“存在多光束干涉”;
else
    输出二束近似厚度结果 d，并增加系统不确定度项;

上述流程图与伪代码明确了多光束干涉判定与厚度计算的实现步骤。其核心方法可以分为以下几个环节：

首先，在数据预处理与诊断阶段，采用快速傅里叶变换（FFT）对实验反射率谱 $R(\sigma)$ 进行谱域分析，判断其中是否存在显著的谐波成分。显著的 FFT 峰值对应于稳定的光程差干涉条纹，其出现是多光束干涉存在的直接证据。

其次，在模型选择阶段，若判定存在多光束干涉，则引入基于 Airy 公式的完整干涉模型，考虑外延层界面与衬底界面产生的多次反射和透射过程。光学参数的色散特性通过 Sellmeier 经验公式建模，其中硅晶圆在红外光谱中的折射率可由经验公式表示，如文献 [3] 所示

$$n_{\mathrm{Si}}(\lambda) = \sqrt{1 + \frac{10.6684293\,\lambda^2}{\lambda^2 - (0.301516485)^2} + \frac{0.003043474\,\lambda^2}{\lambda^2 - (1.13475115)^2}}\,, \tag{23}$$

其中 λ 为波长（μm）。同时，硅衬底的折射率在近红外区可近似取常数 $n_2 = 3.42$[3]。

再次，在参数反演阶段，通过非线性最小二乘方法拟合实验反射率谱与理论反射率谱 $R_{\mathrm{model}}(\sigma)$，从而得到外延层厚度与复折射率等待估参数。

最后，在不确定度量化与结果评估阶段，利用协方差矩阵对拟合参数的不确定度进行估计，同时通过 Bootstrap 抽样等方法进一步评估拟合稳定性, 以保证厚度计算结果的可靠性。

综上所述，整个方法体系由 FFT 诊断、Airy 干涉模型、Sellmeier 色散模型以及不确定度分析共同组成，从而实现了对多光束干涉存在性的判定与对外延层厚度的精确计

算。

7.5 对硅晶圆数据的判定结果与厚度计算

根据前文提出的判定流程，对附件 3 与附件 4 的硅晶圆反射率谱进行了 FFT 诊断与单层 Airy 多光束模型拟合分析。附件 3（入射角 10°）的 FFT 峰值幅度为 303.15，附件 4（入射角 15°）的 FFT 峰值幅度为 341.93，均出现显著频谱峰值，表明多光束干涉效应显著存在。因此在厚度反演过程中，优先采用完整 Airy 公式下的多光束干涉模型结合 Sellmeier 色散关系进行拟合计算。

拟合结果表明：附件 3、4 条件下晶圆外延层分别厚度为

$$d_{\text{附件 3}} = (4.8115 \pm 0.0010)\,\mu\text{m}, \qquad d_{\text{附件 4}} = (4.8125 \pm 0.0010)\,\mu\text{m}.$$

两组结果在误差范围内相互一致，进一步验证了基于 FFT 判定与 Airy 多光束模型模型拟合的计算方法具有良好的稳定性与可靠性。图 6 与图 7 给出了 FFT 诊断图谱，直观展示了谱域谐波成分。

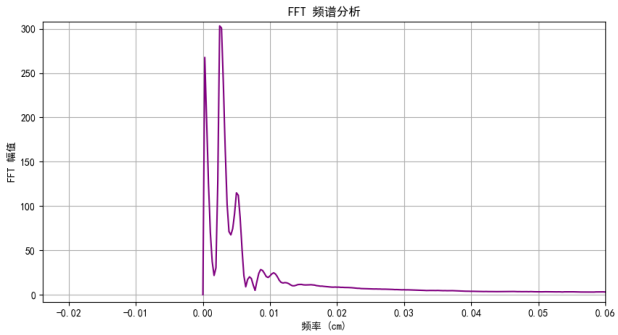


图 6: 附件 3（入射角 10°）FFT 诊断结果

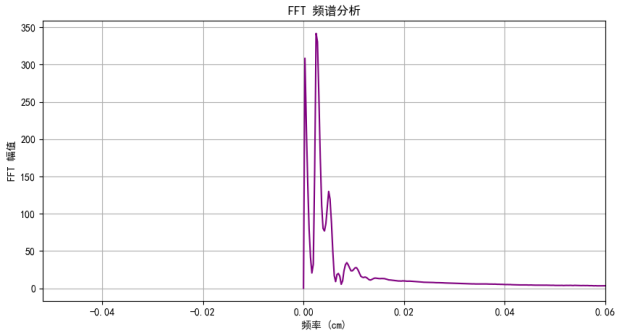
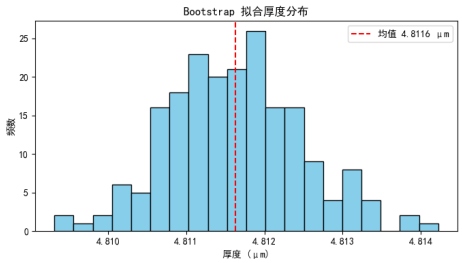


图 7: 附件 4（入射角 15°）FFT 诊断结果

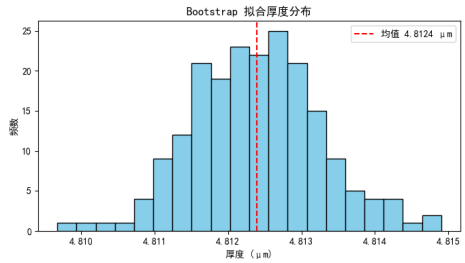
为了进一步验证厚度拟合的稳定性，本文采用 Bootstrap 抽样方法对实验数据进行重复拟合分析。在每次抽样中，以最优拟合曲线为基准，从残差向量中有放回抽样生成新的数据集，并重新进行拟合，统计得到厚度分布。结果表明，Bootstrap 的厚度估计具有高度稳定性，其统计结果如下：

$$d_{\text{附件 3}} = 4.8116\,\mu\text{m} \quad \pm \quad 0.0009\,\mu\text{m}, \qquad d_{\text{附件 4}} = 4.8124\,\mu\text{m} \quad \pm \quad 0.0009\,\mu\text{m}.$$

相应的厚度分布直方图如图 8 所示。可以看出，两组数据的厚度分布均呈近似正态分布，且标准差仅在 $10^{-3}\,\mu$m 量级，说明拟合结果对噪声扰动不敏感，厚度估计具有良好的稳健性。



(a) 附件 3 Bootstrap 结果



(b) 附件 4 Bootstrap 结果

图 8: Bootstrap 抽样厚度分布结果。

7.6 碳化硅晶圆片模型的修正与多光束干涉影响消除

题目进一步要求，如果认为多光束干涉同样出现在碳化硅晶圆片的测试结果（附件 1 和附件 2）中，从而影响到外延层厚度计算的精度，则需针对该现象建立修正模型，并给出消除其影响后的计算结果。这一要求的核心在于：首先判断碳化硅样品中是否存在显著的多次反射与透射引起的干涉效应，其次在理论建模与数值拟合中考虑多光束干涉的贡献，从而避免厚度估计的系统偏差。

7.6.1 碳化硅晶圆片多光束干涉存在性的判定

为了判定碳化硅外延层样品是否存在多光束干涉效应，本文沿用上文提及的快速傅里叶变换（FFT）方法对反射率谱进行了频谱分析，检查 FFT 频谱中是否存在显著峰值，对应于干涉条纹的空间频率，从而判定为多光束干涉成立。

对于附件 1（入射角 10°），FFT 频谱的峰值幅度为 367.81；对于附件 2（入射角

15°），FFT 峰值幅度为 399.65。两者均远大于经验阈值（0.01），表明实验反射率谱中存在显著的周期性条纹。结合图 9 和图 10 所示的 FFT 频谱，可以确认碳化硅晶圆片样品中确实存在多光束干涉现象。因此，在后续厚度与折射率反演中，应当对原有模型进行修正以消除多光束带来的影响。

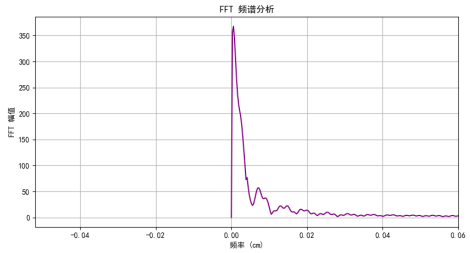


图 9: 碳化硅晶圆片（入射角 10°）的 FFT 频谱

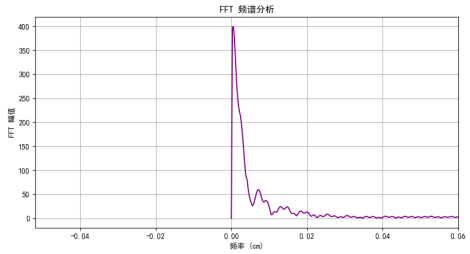


图 10: 碳化硅晶圆片（入射角 15°）的 FFT 频谱

7.6.2 多光束干涉修正模型的建立

在实际实验测量中，碳化硅外延层的反射率谱常常受到多光束干涉的影响。若直接使用单光束模型进行拟合，将导致厚度估计存在系统误差。为了消除干涉影响，我们引入了两种处理方法：平滑滤波和 FFT 滤波。

首先，使用 Savitzky-Golay 平滑滤波对实验数据进行去噪，以减弱高频干涉成分；其次，通过对反射率谱进行 FFT 分析，将频率高于阈值的成分置零，再进行反变换得到 FFT 滤波后的数据。然后，将处理后的数据分别代入单层薄膜传输矩阵模型 (TMM) 中进行厚度拟合，采用问题二中使用的 Sellmeier 色散经验公式

$$n^2(\lambda) = 1 + \frac{5.5394\,\lambda^2}{\lambda^2 - 0.026945},$$

并利用归一化卡方 χ^2 评估拟合质量。

实验结果及拟合厚度如下表所示（厚度 d 单位为 μm，χ^2 为归一化卡方）：

表 1: 多光束干涉修正后的厚度拟合结果

样品	数据处理方法	$d \pm \Delta d$ (μm)	χ^2
附件 2 (15°)	原始数据	8.6634 ± 0.0713	0.0371
	平滑消干涉	8.6635 ± 0.0713	0.0371
	FFT 滤波消干涉	9.2667 ± 0.0680	0.0345
附件 1 (10°)	原始数据	8.6363 ± 0.0668	0.0316
	平滑消干涉	8.6364 ± 0.0667	0.0315
	FFT 滤波消干涉	9.2297 ± 0.0635	0.0292

图 11 与图 12 分别展示了附件 2（入射角 15°）和附件 1（入射角 10°）的反射率谱对比，其中包含原始数据、平滑消干涉和 FFT 滤波消干涉后的结果。可以看出，FFT 滤波后的曲线去除了明显的干涉振荡，更平滑地反映了外延层的光学特性。

需要注意的是，真实实验数据未经处理，且已知存在多光束干涉。如果直接使用单光束模型拟合原始数据，得到的厚度和折射率结果可能会产生偏差。FFT 滤波可以有效去除多光束干涉对测量的影响，因此与原始数据所得结果相比，FFT 滤波能够更准确地反映外延层的真实光学特性。因此，在最终结果中应优先选择 FFT 滤波后的拟合结果。

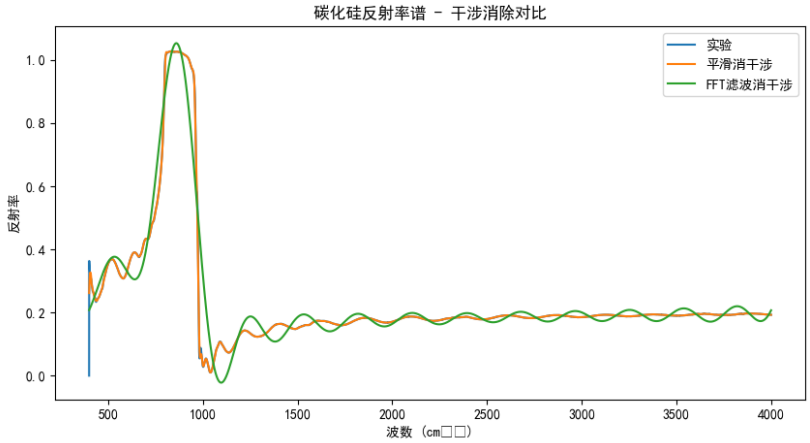


图 11: 附件 2（15°）反射率谱对比：原始数据、平滑消干涉、FFT 滤波消干涉。

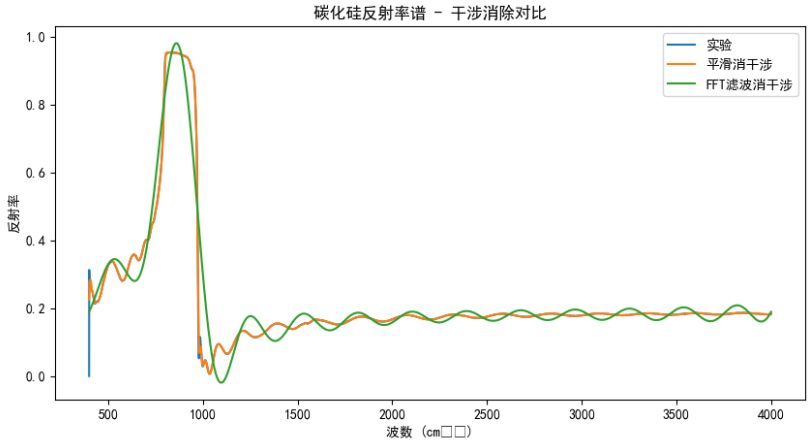


图 12: 附件 1（10°）反射率谱对比：原始数据、平滑消干涉、FFT 滤波消干涉。

综上所述，引入 FFT 滤波进行多光束干涉修正，可以有效改善厚度拟合结果，使测量更加准确可靠。最终修正后的模型计算出碳化硅晶圆外延层厚度为

$$d_{\text{附件 1}} = 9.2297 \pm 0.063\,\mu\text{m}, \qquad d_{\text{附件 2}} = 9.2667 \pm 0.068\,\mu\text{m}.$$

8 模型评价

在本研究中，针对附件 1、2 和附件 3 所涉及的厚度测量任务，分别建立了单光束干涉模型、无限反射 Airy 模型以及 FFT 滤波修正模型，对其适用性与计算精度进行评价如下：

单光束干涉模型（用于问题一、二，分析附件 1、2）

该模型假设反射仅来自单次界面反射，计算简单、解析性强，对噪声不敏感。适用于样品厚度较薄且多光束效应可忽略的情况。然而，当样品存在明显多光束干涉时，单光束模型容易导致厚度低估或拟合残差偏大，因此在附件 1、2 中适用性欠佳。

无限反射 Airy 模型（用于问题三，分析附件 3、4）

Airy 型模型考虑了多次界面反射叠加产生的干涉，能够较好拟合多光束干涉的周期性峰谷结构。其优点是可解释 FFT 频谱中谐波成分，厚度计算精度高，残差较小；缺点是对噪声较敏感，需要合理预处理数据，并且计算较为复杂。

FFT 滤波修正模型（用于问题三，分析附件 1、2）

在附件 1、2 中，通过对原始数据进行 FFT 滤波消除多光束干涉后，使用单层 TMM 模型进行拟合。该方法考虑单层外延层厚度及折射率的色散特性，能够得到稳定可靠的厚度反演结果。相比 Airy 模型，经过滤波后的 TMM 模型对噪声更稳健，计算结果符合实验物理预期。

综合评价

综上所述，对于厚度较薄且多光束可忽略的样品，单光束干涉模型适用；对于存在多光束干涉的碳化硅晶圆片，先通过 FFT 识别并滤除多光束影响，再采用单层 TMM 模型拟合，能够获得稳健且高精度的厚度测量结果。

参考文献

[1] M. Levinshtein, S. Rumyantsev, and M. Shur (Eds.), *Properties of Advanced Semiconductor Materials: GaN, AlN, InN, BN, SiC, SiGe*. John Wiley & Sons, 2001.

[2] D. E. Aspnes, "Optical properties of thin films," Thin Solid Films, vol. 89, no. 3, pp. 249–262, 1983.

[3] M. A. Green, "Self-consistent optical parameters of intrinsic silicon at 300 K including temperature coefficients," *Solar Energy Materials and Solar Cells*, vol. 92, no. 11, pp. 1305–1310, 2008. doi:10.1016/j.solmat.2008.06.009

A Python 代码附录

本研究的计算与数据分析过程依赖于以下四个 Python 脚本，分别用于数据预处理与寻峰、单光束模型拟合、多光束干涉分析，以及

A.1 peak_find.py：数据预处理与峰谷检测

用于对实验光谱数据进行平滑、滤波，并自动检测波峰和波谷。

```
import pandas as pd
import numpy as np
from scipy.signal import savgol_filter, find_peaks, medfilt
import matplotlib.pyplot as plt
from scipy.ndimage import gaussian_filter1d

# ---------------- 设置中文字体 ----------------
plt.rcParams['font.sans-serif'] = ['SimHei']      # 用黑体显示中文
plt.rcParams['axes.unicode_minus'] = False        # 正常显示负号

# ----------- 读入数据 -----------
file1 = "附件1.xlsx"
file2 = "附件2.xlsx"

data1 = pd.read_excel(file1)
data2 = pd.read_excel(file2)
data1.columns = ["波数", "反射率"]
data2.columns = ["波数", "反射率"]

# ----------- 预处理函数 -----------
def preprocess(df, col="反射率"):
    y = df[col].values

    # 1. 中值滤波，去掉尖刺
    y = medfilt(y, kernel_size=5)

    # 2. Savitzky-Golay 滤波（轻度）
    y_smooth = savgol_filter(y, window_length=31, polyorder=3)
```

```
    # 3. 二级平滑（用于计算导数，抑制平稳区伪峰）
    y_smooth_strong = gaussian_filter1d(y_smooth, sigma=5)

    df[col + "_平滑"] = y_smooth
    df[col + "_强平滑"] = y_smooth_strong
    return df

data1 = preprocess(data1)
data2 = preprocess(data2)


# ----------- 只找波峰，含去重 -----------
def detect_peaks(x, y_smooth_strong, prominence=0.005,
    distance=30, width=20, merge_threshold=50):
    """
    x: 波数
    y_smooth_strong: 强平滑曲线
    prominence: 峰的显著性
    distance: 相邻峰的最小间隔
    width: 峰的最小宽度
    merge_threshold: 如果两个峰间距小于该值，合并为一个（保留高的）
    """
    peaks, properties = find_peaks(
        y_smooth_strong,
        prominence=prominence,
        distance=distance,
        width=width
    )

    # ---- 二次筛选：合并太近的峰 ----
    final_peaks = []
    if len(peaks) > 0:
        final_peaks = [peaks[0]]
        for p in peaks[1:]:
            if x[p] - x[final_peaks[-1]] < merge_threshold:
                # 两个峰太近 -> 保留更高的那个
```

```
                if y_smooth_strong[p] > y_smooth_strong[final_peaks[-1]]:
                    final_peaks[-1] = p
            else:
                final_peaks.append(p)
    return np.array(final_peaks)


# ----------- 绘图并保存函数 -----------
def plot_save_peaks(df, filename, title="光谱曲线"):
    x = df["波数"].values
    y_raw = df["反射率"].values
    y_smooth = df["反射率_平滑"].values
    y_smooth_strong = df["反射率_强平滑"].values

    peaks = detect_peaks(x, y_smooth_strong)

    # 绘图
    plt.figure(figsize=(10,6))
    plt.plot(x, y_raw, alpha=0.4, label="原始数据")
    plt.plot(x, y_smooth, label="平滑曲线", linewidth=2)
    plt.plot(x, y_smooth_strong, "--", label="强平滑曲线", alpha=0.7)
    plt.scatter(x[peaks], y_smooth_strong[peaks],
        color="r", marker="^", s=80, label="波峰")
    plt.xlabel("波数 (cm⁻¹)")
    plt.ylabel("反射率")
    plt.title(title)
    plt.legend()
    plt.grid(alpha=0.3)
    plt.tight_layout()
    plt.savefig(filename, dpi=300)   # 保存图片
    plt.show()

    # 保存波峰数据
    peak_data = pd.DataFrame({
        "波数": x[peaks],
        "反射率": y_smooth_strong[peaks]
    })
```

```
    # 计算峰峰间距
    peak_spacing = np.diff(x[peaks])
    median_spacing = np.median(peak_spacing)
        if len(peak_spacing) > 0 else np.nan

    return peak_data, peak_spacing, median_spacing


# ----------- 分别保存两组数据 -----------
peak_data1, spacing1, median_spacing1 =
    plot_save_peaks(data1, "附件1_光谱波峰.png", title="附件1 光谱曲线")
peak_data2, spacing2, median_spacing2 =
    plot_save_peaks(data2, "附件2_光谱波峰.png", title="附件2 光谱曲线")

# ----------- 保存波峰数据到 Excel --------
with pd.ExcelWriter("波峰数据.xlsx") as writer:
    peak_data1.to_excel(writer, sheet_name="附件1波峰", index=False)
    peak_data2.to_excel(writer, sheet_name="附件2波峰", index=False)

# ----------- 输出峰间距中位数 --------
print("附件1峰峰间距中位数: ", median_spacing1)
print("附件2峰峰间距中位数: ", median_spacing2)
```

A.2 peak_fit.py：单光束干涉模型与厚度拟合

基于单光束干涉理论建立厚度模型，并使用非线性最小二乘拟合附件 12 碳化硅晶圆片厚度。

```
# single_beam_peak_fit_sellmeier.py

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

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

# --- 用户设置 ---
sheet1 = "附件1波峰"    #  = 10°
sheet2 = "附件2波峰"    #  = 15°
peak_file = "波峰数据.xlsx"

theta1_deg = 10.0
theta2_deg = 15.0
theta1 = np.deg2rad(theta1_deg)
theta2 = np.deg2rad(theta2_deg)

m0_window = 200  # 枚举 m_ref ± 窗口
continuous_inits = [50e-4]  # 初始 d 值 (cm), 例如 50 μm

d_min = 1e-4  # 1 μm
d_max = 1e-2  # 100 μm

# --- helpers ---
def read_peak_sheet(filename, sheet):
    df = pd.read_excel(filename, sheet_name=sheet)
    if '波数' in df.columns:
        sigma = df['波数'].values.astype(float)
    else:
        sigma = df.iloc[:,0].values.astype(float)
    return np.sort(sigma)

def n_of_sigma_sellmeier(sigma):
    lam_cm = 1.0 / sigma             # 波长 (cm)
    lam_um = lam_cm * 1e4            # 转 μm
    lam2 = lam_um**2
    numerator = 5.5394 * lam2
    denominator = lam2 - 0.026945
    n2 = 1.0 + numerator / denominator
    return np.sqrt(n2)
```

```
def residuals_continuous(x, all_sigma, all_theta, all_m):
    d_cm = x[0]
    n_vals = n_of_sigma_sellmeier(all_sigma)
    pred = 2.0 * n_vals * d_cm * np.cos(all_theta) * all_sigma
    return pred - all_m

# --- main ---
if __name__ == "__main__":
    if not os.path.exists(peak_file):
        raise FileNotFoundError(f"未找到 {peak_file}")

    sigma1 = read_peak_sheet(peak_file, sheet1)
    sigma2 = read_peak_sheet(peak_file, sheet2)
    print(f"附件1峰数: {len(sigma1)}, 附件2峰数: {len(sigma2)}")

    def median_spacing(sig):
        ds = np.diff(sig)
        ds = ds[(ds>0)&(ds<np.percentile(ds,95))]
        return np.median(ds) if len(ds)>0 else np.nan

    med1, med2 = median_spacing(sigma1), median_spacing(sigma2)
    print("Δ 中位数 (10°):", med1, " (15°):", med2)

    nd1 = 1.0/(2*med1*np.cos(theta1)) if not np.isnan(med1) else None
    print("approx nd:", nd1)

    idx_ref1, idx_ref2 = len(sigma1)//2, len(sigma2)//2
    sig_ref1, sig_ref2 = sigma1[idx_ref1], sigma2[idx_ref2]
    n_guess = n_of_sigma_sellmeier(sig_ref1)
    d_guess = nd1 / n_guess if nd1 else 50e-4
    m_ref1 = int(round(2*n_guess*d_guess*np.cos(theta1)*sig_ref1))
    m_ref2 = int(round(2*n_guess*d_guess*np.cos(theta2)*sig_ref2))
    print("估算 m_ref:", m_ref1, m_ref2)

    best = {'loss': np.inf}

    all_sigma_template = np.concatenate([sigma1, sigma2])
```

```
all_theta_template = np.concatenate([np.full_like(sigma1, theta1),
    np.full_like(sigma2, theta2)])

for m0_1 in range(m_ref1-m0_window, m_ref1+m0_window+1):
    m_seq1 = m0_1 + np.arange(len(sigma1)) - idx_ref1
    if np.any(m_seq1<=0): continue
    for m0_2 in range(m_ref2-m0_window, m_ref2+m0_window+1):
        m_seq2 = m0_2 + np.arange(len(sigma2)) - idx_ref2
        if np.any(m_seq2<=0): continue
        all_m = np.concatenate([m_seq1, m_seq2])

        for d0 in continuous_inits:
            x0 = np.array([d0])
            lb, ub = [d_min], [d_max]
            try:
                res = least_squares(residuals_continuous,x0, bounds=(lb,ub),
                    args=(all_sigma_template, all_theta_template, all_m))
            except:
                continue

            loss = np.sum(res.fun**2)
            if loss < best['loss']:
                best.update({
                    'loss':loss, 'm0_1':m0_1,
                    'm0_2':m0_2, 'd_cm':res.x[0],
                    'sigma':all_sigma_template,
                    'theta':all_theta_template, 'm':all_m
                })

if best['loss']==np.inf:
    raise RuntimeError("未找到拟合解, 请放宽搜索条件")

# 计算厚度不确定度
res_best = least_squares(
    residuals_continuous,
    x0=[best['d_cm']],
    bounds=([d_min],[d_max]),
```

```
    args=(best['sigma'], best['theta'], best['m'])
)

J = res_best.jac
residuals = res_best.fun
N = len(residuals)
p = len(res_best.x)
cov = np.linalg.inv(J.T @ J) * np.sum(residuals**2) / (N - p)
d_uncertainty = np.sqrt(cov[0,0])

print(f"厚度 d = {best['d_cm']*1e4:.4f}
    ± {d_uncertainty*1e4:.4f} μm")


print("\n===== 拟合结果 (Sellmeier 模型) =====")
print(f"m0(10°)={best['m0_1']}, m0(15°)={best['m0_2']}")
print(f"厚度 d = {best['d_cm']*1e4:.4f} μm,
    loss = {best['loss']:.4e}")

pred_LHS = 2.0 * n_of_sigma_sellmeier(best['sigma'])
    * best['d_cm'] * np.cos(best['theta']) * best['sigma']
plt.figure(figsize=(8,6))
plt.scatter(best['m'], pred_LHS, s=10)
plt.plot(best['m'], best['m'], 'k--')
plt.xlabel('整数 m')
plt.ylabel('2 n( ) d cosr  ')
plt.title('Sellmeier 模型拟合检验')
plt.grid(True)
plt.tight_layout()
plt.savefig("fit_sellmeier_check.png", dpi=300)
plt.show()

# ===== 额外绘制 n-lambda 曲线 =====
# 波长范围: 2.5 μm ~ 25 μm
lam_um = np.linspace(2.5, 25, 500)     # 波长 (μm)
sigma = 1e4 / lam_um                   # 转换为波数 (cm^-1)
```

```
    n_vals = n_of_sigma_sellmeier(sigma)

    plt.figure(figsize=(8,6))
    plt.plot(lam_um, n_vals, 'b-', linewidth=2)
    plt.xlabel("波长  (μm)")
    plt.ylabel("折射率 n( )")
    plt.title("碳化硅 Sellmeier 模型折射率色散曲线")
    plt.grid(True)
    plt.tight_layout()
    plt.savefig("n_lambda.png", dpi=300)
    plt.show()
```

A.3 silicon.py：Airy 多光束干涉与 FFT 检测

建立 Airy 多光束干涉模型，通过 FFT 检测多光束干涉并拟合硅晶圆片厚度。

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.optimize import least_squares
from scipy.fft import fft, fftfreq

# ---------------- 设置中文字体 ----------------
plt.rcParams['font.sans-serif'] = ['SimHei']      # 用黑体显示中文
plt.rcParams['axes.unicode_minus'] = False        # 正常显示负号

# ------------------- 读取数据 -------------------
data = pd.read_excel("附件4.xlsx")
sigma = data.iloc[:, 0].values  # 波数 cm^-1
R_exp = data.iloc[:, 1].values / 100  # 百分比转0-1

# ------------------- 入射角 -------------------
theta_deg = 15
theta = np.deg2rad(theta_deg)

# ------------------- Sellmeier 硅晶圆经验公式 -------------------
```

```
def n_silicon(lambda_um):
    return np.sqrt(1 + 10.6684293*lambda_um**2/(lambda_um**2 - 0.301516485**2) +
                   0.003043474*lambda_um**2/(lambda_um**2 - 1.13475115**2))

# ------------------- 多光束判断 (FFT) -------------------
def check_multibeam(sigma, R):
    R_centered = R - np.mean(R)
    N = len(R_centered)
    fft_vals = np.abs(fft(R_centered))
    freqs = fftfreq(N, d=(sigma[1]-sigma[0]))
    fft_peak = np.max(fft_vals[1:N//2])
    print(f"FFT peak magnitude: {fft_peak}")
    return fft_peak > 0.01, freqs[:N//2], fft_vals[:N//2]

multi_beam, freqs, fft_vals = check_multibeam(sigma, R_exp)
if multi_beam:
    print("检测到多光束干涉，优先使用无限反射 Airy 模型拟合")
else:
    print("未检测到明显多光束干涉，可简化模型拟合")

# ------------------- Airy 无限反射模型 -------------------
def R_model(d_cm):
    lambda_um = 1e4 / sigma
    n = n_silicon(lambda_um)
    n_complex = n
    n_2 = 3.42
    theta_r = np.arcsin(np.sin(theta) / n)
    r12 = (1 - n)/(1 + n)
    r23 = (n - 1)/(n + 1)
    delta = 4 * np.pi * n_complex * d_cm * np.cos(theta_r) * sigma
    delta = np.mod(delta, 2*np.pi)
    R = np.abs((r12 + r23 * np.exp(1j*delta)) / (1 + r12*r23*np.exp(1j*delta)))**2
    return R

# ------------------- 拟合函数 -------------------
def residual(d_array):
    d = d_array[0]
```

```
    R_fit = R_model(d)
    return R_fit - R_exp

# ------------------- 初值和边界 -------------------
d0 = np.array([5e-4])  # 初始厚度 5 m -> 5e-4 cm
bounds = (1e-5, 1e-3)  # 厚度 0.1 m ~ 10 m

res = least_squares(residual, d0, bounds=bounds)
d_fit = res.x[0]

# ------------------- 不确定度计算 -------------------
residual_std = np.std(res.fun)
J = res.jac
cov = np.linalg.inv(J.T @ J) * residual_std**2
d_uncertainty = np.sqrt(np.diag(cov))[0]

print(f"拟合厚度 d = {d_fit*1e4:.4f} ± {d_uncertainty*1e4:.4f} m")

# ------------------- 绘图 -------------------
plt.figure(figsize=(10,5))
plt.plot(sigma, R_exp, label="实验")
plt.plot(sigma, R_model(d_fit), label=f"拟合 d={d_fit*1e4:.4f} m")
plt.xlabel("波数 (cm⁻¹)")
plt.ylabel("反射率")
plt.title("硅晶圆外延层反射率拟合")
plt.legend()
plt.show()

# ------------------- FFT 结果判定 -------------------
if multi_beam:
    print("FFT 分析结果：干涉条纹显著，多光束干涉成立。")
else:
    print("FFT 分析结果：干涉条纹不明显，可能为低阶干涉或噪声主导。")

# ------------------- 绘制 FFT 频谱 -------------------
plt.figure(figsize=(10,5))
plt.plot(freqs, fft_vals, color="purple")
```

```
plt.xlabel("频率 (cm)")
plt.ylabel("FFT 幅值")
plt.title("FFT 频谱分析")
plt.grid(True)
plt.show()

# ------------------- Bootstrap 抽样评估拟合稳定性 -------------------
n_bootstrap = 200   # 抽样次数
bootstrap_results = []

R_fit_best = R_model(d_fit)

np.random.seed(42)
for _ in range(n_bootstrap):
    # 在拟合残差上随机加噪声
    noise = np.random.choice(res.fun, size=len(res.fun), replace=True)
    R_bootstrap = R_fit_best + noise

    def residual_bootstrap(d_array):
        d = d_array[0]
        R_fit = R_model(d)
        return R_fit - R_bootstrap

    res_bs = least_squares(residual_bootstrap, d0, bounds=bounds)
    bootstrap_results.append(res_bs.x[0])

bootstrap_results = np.array(bootstrap_results)
d_mean = np.mean(bootstrap_results)
d_std = np.std(bootstrap_results)

print(f"Bootstrap 结果: 平均厚度 d = {d_mean*1e4:.4f} m,
    标准差 = {d_std*1e4:.4f} m")

# 直方图展示
plt.figure(figsize=(8,4))
plt.hist(bootstrap_results*1e4, bins=20, color="skyblue", edgecolor="black")
plt.axvline(d_mean*1e4, color="red", linestyle="--",
```

```
    label=f"均值 {d_mean*1e4:.4f} m")
plt.xlabel("厚度 (m)")
plt.ylabel("频数")
plt.title("Bootstrap 拟合厚度分布")
plt.legend()
plt.show()
```

A.4 sic_fixed.py：FFT 滤波消除多光束干扰并拟合厚度

通过平滑处理与 FFT 滤波去除多光束干涉信号，使用单层 TMM 模型拟合附件 12 碳化硅晶圆片厚度。

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.optimize import least_squares
from scipy.signal import savgol_filter
from scipy.fft import fft, ifft, fftfreq

# ---------------- 设置中文字体 ----------------
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

# ------------------- 读取数据 -------------------
data = pd.read_excel("附件1.xlsx")
sigma = data.iloc[:, 0].values
R_exp = data.iloc[:, 1].values / 100.0

theta_deg = 10
theta = np.deg2rad(theta_deg)

# ------------------- Sellmeier 色散公式 -------------------
def n_of_sigma_sellmeier(sigma):
    lam_cm = 1.0 / sigma
    lam_um = lam_cm * 1e4
    lam2 = lam_um**2
```

```
    numerator = 5.5394 * lam2
    denominator = lam2 - 0.026945
    n2 = 1.0 + numerator / denominator
    return np.sqrt(n2)

# ------------------- FFT 检测 -------------------
def check_multibeam(sigma, R):
    R_centered = R - np.mean(R)
    N = len(R_centered)
    fft_vals = np.abs(fft(R_centered))
    freqs = fftfreq(N, d=(sigma[1]-sigma[0]))
    fft_peak = np.max(fft_vals[1:N//2])
    return fft_peak, freqs[:N//2], fft_vals[:N//2]

fft_peak, freqs, fft_vals = check_multibeam(sigma, R_exp)
print(f"FFT peak magnitude: {fft_peak:.2f}")
if fft_peak > 0.01:
    print("检测到多光束干涉，需消除干涉影响")
else:
    print("未检测到明显多光束干涉")

# ------------------- 去干涉 -------------------
R_smooth = savgol_filter(R_exp, window_length=51, polyorder=3)

R_centered = R_exp - np.mean(R_exp)
N = len(R_centered)
fft_vals_full = fft(R_centered)
freqs_full = fftfreq(N, d=(sigma[1]-sigma[0]))
fft_filtered = fft_vals_full.copy()

# 改进阈值：保留低频，避免M型问题
freq_cut = 0.0035
fft_filtered[np.abs(freqs_full) > freq_cut] = 0
R_fft_filtered = np.real(ifft(fft_filtered)) + np.mean(R_exp)

# ------------------- TMM 模型 -------------------
def tmm_model(d_cm):
```

```
    k0 = 2*np.pi*sigma
    n_air = 1.0
    n_film = n_of_sigma_sellmeier(sigma)
    n_sub = 2.6
    theta_film = np.arcsin(n_air*np.sin(theta)/n_film)
    theta_sub = np.arcsin(n_air*np.sin(theta)/n_sub)

    def r_s(n1, n2, th1, th2):
        return (n1*np.cos(th1) - n2*np.cos(th2)) / (n1*np.cos(th1) + n2*np.cos(th2))

    r01 = r_s(n_air, n_film, theta, theta_film)
    r12 = r_s(n_film, n_sub, theta_film, theta_sub)
    delta = k0 * n_film * d_cm * np.cos(theta_film)
    r_total = (r01 + r12*np.exp(2j*delta)) / (1 + r01*r12*np.exp(2j*delta))
    return np.abs(r_total)**2

# ------------------- 拟合函数 -------------------
def fit_thickness(R_input):
    def residual(d_array):
        d = d_array[0]
        return tmm_model(d) - R_input

    d0 = np.array([9.1451e-4])
    bounds = (1e-5, 1e-3)
    res = least_squares(residual, d0, bounds=bounds)
    d_fit = res.x[0]
    residual_std = np.std(res.fun)
    J = res.jac
    cov = np.linalg.inv(J.T @ J) * residual_std**2
    d_uncertainty = np.sqrt(np.diag(cov))[0]
    chi2 = np.sum(res.fun**2) / len(R_input)  # 归一化卡方
    return d_fit, d_uncertainty, chi2

# ------------------- 三种情况拟合 -------------------
d_raw, d_raw_unc, chi2_raw = fit_thickness(R_exp)
d_smooth, d_smooth_unc, chi2_smooth = fit_thickness(R_smooth)
d_fft, d_fft_unc, chi2_fft = fit_thickness(R_fft_filtered)
```

```
print("\n厚度拟合结果 (Sellmeier 色散公式, s偏振, ²归一化):")
print(f"原始数据: d = {d_raw*1e4:.4f} ±
    {d_raw_unc*1e4:.4f} m, ² = {chi2_raw:.4f}")
print(f"平滑消干涉: d = {d_smooth*1e4:.4f} ±
    {d_smooth_unc*1e4:.4f} m, ² = {chi2_smooth:.4f}")
print(f"FFT滤波消干涉: d = {d_fft*1e4:.4f} ±
    {d_fft_unc*1e4:.4f} m, ² = {chi2_fft:.4f}")

# ------------------- 反射率谱对比 -------------------
plt.figure(figsize=(10,5))
plt.plot(sigma, R_exp, label="实验")
plt.plot(sigma, R_smooth, label="平滑消干涉")
plt.plot(sigma, R_fft_filtered, label="FFT滤波消干涉")
plt.xlabel("波数 (cm ¹)")
plt.ylabel("反射率")
plt.title("碳化硅反射率谱 - 干涉消除对比")
plt.legend()
plt.show()

# ------------------- FFT 频谱 -------------------
plt.figure(figsize=(10,5))
plt.plot(freqs_full[:N//2], np.abs(fft_vals_full)[:N//2], color="purple")
plt.xlabel("频率 (cm)")
plt.ylabel("FFT 幅值")
plt.title("FFT 频谱分析 (碳化硅)")
plt.grid(True)
plt.show()
```