相位衍射相位恢复方法

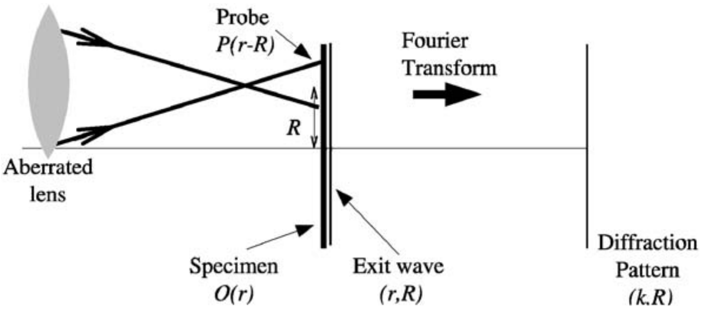
使用多重网格方法加速层叠相干衍射成像

Julian OU

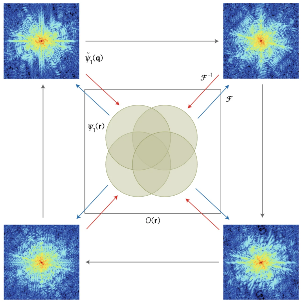
College of Physics, Sichuan University, Chengdu 610064, China

2021 年 5 月 20 日

目录



(a) 装置示意图



(b) 迭代示意图

图 1: 层叠相干衍射成像（Ptychography）

在照明区域的大小和位置已知的情况下，算法的内循环分为以下几个步骤。首先是给定整个对象区域的振幅和相位一初值 $O_0(\vec{r})$，以及照明函数一初值 $P_0(\vec{r})$

1. 用对象函数 $O_j(\vec{r})$ 和照明函数 $P_j(\vec{r}-\vec{R})$ 计算出该照明区域的出射波函数，

$$\Psi_j(\vec{r},\vec{R}) = O_j(\vec{r})P_j(\vec{r}-\vec{R}) \tag{1}$$

其中 $\vec{R}$ 为样品与照明区域之间的相对位移。

2. 对出射函数应用傅里叶变换，并用衍射图案的测量值替代计算值，

$$F_j(\vec{k}) = \mathcal{F}\{\Psi_j(\vec{r})\} \qquad F_j'(\vec{k}) = |F_{je}(\vec{k})| \cdot \frac{F_j(\vec{k})}{|F_j(\vec{k})|}$$

其中 $|F_{je}(\vec{k})|$ 是指照射第 j 个照明区域所产生的衍射图案的测量值。

3. 应用傅里叶逆变换并做适当的约束，得到更新后的出射波函数，

$$\Psi_j'(\vec{r}) = \mathcal{F}^{-1}\{F_j'(\vec{k})\} \qquad \Psi_j''(\vec{r}) = C[\Psi_j'(\vec{r})]$$

其中 C 函数指的是约束函数，也可以是无约束。

4. 用更新后的出射波函数拆分得到对象函数的更新函数，

$$O_{j+1}(\vec{r}) = O_j(\vec{r}) + \alpha\left(\frac{|P_j(\vec{r}-\vec{R})|^2}{[P_j(\vec{r}-\vec{R})]^2_{MAX}}\frac{P_j^*(\vec{r}-\vec{R})}{|P_j(\vec{r}-\vec{R})|^2+\beta}\right)[\Psi_j''(\vec{r}) - \Psi_j(\vec{r})] \tag{2}$$

或（ePIE）得到对象函数和照明函数的更新函数，

$$O_{j+1}(\vec{r}) = O_j(\vec{r}) + \alpha_O\frac{P_j^*(\vec{r}-\vec{R})}{|P_j(\vec{r}-\vec{R})|^2_{MAX}}[\Psi_j''(\vec{r}) - \Psi_j(\vec{r})]$$

$$P_{j+1}(\vec{r}) = P_j(\vec{r}) + \alpha_P\frac{O_j^*(\vec{r}+\vec{R})}{|O_j(\vec{r}+\vec{R})|^2_{MAX}}[\Psi_j''(\vec{r}) - \Psi_j(\vec{r})]$$

Subsection 1

更新函数的权重表示

考虑最简单的更新函数，由 (1) 可得：

$$O_{j+1}(\vec{r}) = \frac{\Psi_j''(\vec{r})}{P_j(\vec{r}-\vec{R})} = \frac{P_j^*(\vec{r}-\vec{R})}{|P_j(\vec{r}-\vec{R})|^2}\Psi_j''(r) \tag{3}$$

显然上式对于 (3) 式对于在照明强度较弱的情况下条件较差，应该更多的保留之前对于对象函数的估计，于是可以在更新时加上一定的权重：

$$\begin{aligned} O_{j+1}(\vec{r}) &= (1-w_j)O_j(\vec{r}) + w_j\frac{P_j^*(\vec{r}-\vec{R})}{|P_j(\vec{r}-\vec{R})|^2}\Psi_j''(r) \\ &= O_j(\vec{r}) + w_j\frac{P_j^*(\vec{r}-\vec{R})}{|P_j(\vec{r}-\vec{R})|^2}[\Psi_j''(\vec{r}) - \Psi_j(\vec{r})] \end{aligned}$$

$$O_{j+1}(\vec{r}) = O_j(\vec{r}) + w_j \frac{P_j^*(\vec{r}-\vec{R})}{|P_j(\vec{r}-\vec{R})|^2}[\Psi_j''(\vec{r}) - \Psi_j(\vec{r})]$$

其中 w_j 由

$$w_j = \alpha \frac{|P_j(\vec{r}-\vec{R})|^2}{|P_j(\vec{r}-\vec{R})|_{MAX}^2}$$

给出，表示第 j 个照明区域的更新权重，于是可以得到对象函数的更新函数：

$$O_{j+1}(\vec{r}) = O_j(\vec{r}) + \alpha \frac{P_j^*(\vec{r}-\vec{R})}{|P_j(\vec{r}-\vec{R})|_{MAX}^2}[\Psi_j''(\vec{r}) - \Psi_j(\vec{r})]$$

Subsection 2

更新函数的梯度表示

考虑以下的误差度量：

$$E_j^O = \frac{1}{2}\sum_{\vec{r}} |O_j(\vec{r})P_j(\vec{r}-\vec{R}) - \Psi_j''(\vec{r})|^2$$

PIE 算法的目标是找到一个修正的对象函数 $O_j(\vec{r})$，来降低该误差，使出射波 $O_j(\vec{r})P_j(\vec{r}-\vec{R})$ 更接近于算法中更新的出射波 $\Psi_j''(\vec{r})$，该误差相对于对象函数的梯度为：

$$\nabla E_j^O = \frac{\partial E_j^O}{\partial O_j(\vec{r})} = P_j^*(\vec{r}-\vec{R})[O_j(\vec{r})P_j(\vec{r}-\vec{R}) - \Psi_j''(\vec{r})]$$

由于误差是在该梯度方向上增加的，因此可以在负梯度方向上将对象函数移动一个小步长 γ 来减小误差：

$$O_{j+1}(\vec{r}) = O_j(\vec{r}) - \gamma\nabla E_j^O = O_j(\vec{r}) + \gamma P_j^*(\vec{r}-\vec{R})[\Psi_j''(\vec{r}) - \Psi_j(\vec{r})]$$

$$O_{j+1}(\vec{r}) = O_j(\vec{r}) + \gamma P_j^*(\vec{r} - \vec{R})[\Psi_j''(\vec{r}) - \Psi_j(\vec{r})]$$

其中 γ 设置为

$$\gamma = \frac{\alpha}{|P_j(\vec{r} - \vec{R})|_{MAX}^2}$$

当 $\alpha \leq 1$ 时，步长是稳定的，且对于凸问题收敛性是保证的，但是梯度下降方法收敛缓慢，且收敛缓慢，易停滞在局部极小值处，而 PIE 更新函数的每次应用使对象函数朝着不同方向的梯度移动，因此可以很好的避免这个问题。

$$O_{j+1}(\vec{r}) = O_j(\vec{r}) + \alpha \frac{P_j^*(\vec{r} - \vec{R})}{|P_j(\vec{r} - \vec{R})|_{MAX}^2}[\Psi_j''(\vec{r}) - \Psi_j(\vec{r})]$$

目录

Subsection 1

投影最优化问题

投影最优化问题

层叠相干衍射成像可以描述为：

$$\vec{d}_k = |\mathcal{F}(\hat{Q}_k \vec{z})|^2 + \epsilon_k, \qquad k = 1, 2, \cdots, N$$

其中 $\vec{z}$ 是重建对象，$\vec{d}_k$ 表示第 k 个扫描位置对应的衍射图案测量值，$\hat{Q}_k$ 为照明矩阵（探针），即表示第 k 次扫描的位置。考虑强度的高斯误差度量：

$$\Phi_{\mathcal{JG}}(\vec{z}) = \frac{1}{2} \sum_{k}^{N} \left\| |\mathcal{F}(\hat{Q}_k \vec{z})|^2 - \vec{d}_k \right\|_2^2$$

定义测量约束集，以及其投影算子分别为：

$$\mathcal{M}_k(\vec{z}) = \left\{ \vec{z} : |\mathcal{F}(\hat{Q}_k \vec{z})|^2 = \vec{d}_k \right\}, \qquad \mathcal{P}\mathcal{M}_k(\vec{z}) = \mathcal{F}^{-1}\left[\sqrt{\vec{d}_k} \odot \exp\left(i\theta \mathcal{F}(\hat{Q}_k \vec{z}) \right) \right]$$

则 PIE 对应的投影最优化问题可以表示为:

$$\min_{z} \Phi(\vec{z}), \qquad \Phi(\vec{z}) = \frac{1}{2} \sum_{k}^{N} \left\| \mathcal{P} \mathcal{M}_k(\vec{z}) - \hat{Q}_k \vec{z} \right\|_2^2 \tag{4}$$

即寻找 $\vec{z}$ 使上式右侧达到最小值或者 0。

那么 PIE 迭代过程中的 (2) 式用以上的表示方法可以写为:

$$\vec{z}(\vec{r}) = \vec{z}(\vec{r}) + \alpha \left(\frac{\hat{Q}_k}{[\hat{Q}_k]_{MAX}} \frac{\hat{Q}_k^T}{|\hat{Q}_k|^2 + \beta} \right) [\mathcal{P} \mathcal{M}_k(\vec{z}) - \hat{Q}_k \vec{z}] \tag{5}$$

其中 $\hat{Q}_k^T$ 表示 $\hat{Q}_k$ 的复共轭转置。

Subsection 2

梯度下降方法

Input: $\mathtt{f}(x)$：目标函数，x_0：初始预测值，λ：检索步长 ε：计算精度

for $k = 0, 1, 2, \cdots$ **do**

  $g_k = \mathtt{g}(x_k) = \nabla \mathtt{f}(x_k)$

  if $\|g_k\| < \varepsilon$ **then**

    $x^* = x_k$

  else

    $x_{k+1} = x_k - \lambda g_k$

  end

end

Output: φ：线性方程组的解

Algorithm 1: 梯度下降方法

考虑 (4) 式的梯度：

$$\nabla\Phi(\vec{z}) = \frac{\partial\Phi(\vec{z})}{\partial\vec{z}} = \sum_{k}^{N} \hat{Q}_k[\mathcal{P}\mathcal{M}_k(\vec{z}) - \hat{Q}_k^T\vec{z}] \tag{6}$$

PIE 迭代的内循环（即对所有的 N 个照明区域都做 (5) 式），在基于梯度的最优化问题中可以表示为：

$$\vec{z} = \vec{z} - \gamma\nabla\Phi(\vec{z}) \tag{7}$$

即在每一次外循环中，PIE 等价于向负梯度方向移动了一个小步长。

目录

Subsection 1

基础算法构成

在数值分析中，多重网格法是一种使用层次离散化来求解微分方程的方法，是多分辨率方法的一个例子。由于直接在高分辨率（用于求解的间隔小）上进行求解时对于低频部分收敛较慢，与间隔的平方成反比，便想可以到先在低分辨率（间隔较大）上进行求解，然后再进行插值，提高其分辨率，再在更高分辨率进行计算，这样的方法就叫做多重网格方法，且迭代过程主要包含以下几个重要的组成部分：

- 平滑（smoothing）：用线性求解器进行迭代，降低高频误差。
- 计算残差（Residual Computation）：在平滑之后计算剩余误差（残差）$r_i = b - Ax_i$。
- 限制（Restriction）：降低采样率，把残差放到更加粗糙的网格中。
- 延拓（prolongation）：将粗网格计算的修正值插值到更精细的网格中
- 修正（Correction）：将延拓的修正值添加到精细网格中。

Input: φ：初始预测值，f：常数项，h：最精细网格精度，h_m：最粗糙网络精度

Function VCycle(φ, f, h):

  $\varphi = \texttt{smoothing}(\varphi, f, h)$ ' 预平滑（pre-smoothing）

  $res = \texttt{residual}(\varphi, f, h)$

  $rhs = \texttt{restriction}\ (res)$

  $eps = \texttt{zeros}\ (\text{size}\ (rhs))$

  if $h < h_m$ **then**

    VCycle$(eps, rhs, 2h)$

  else

    smoothing$(eps, rhs, 2h)$

  end

  $\varphi = \varphi + \texttt{prolongation}\ (eps)$

  $\varphi = \texttt{smoothing}(\varphi, f, h)$ ' 后平滑（post-smoothing）

  return φ

end

Output: φ：方程的解

Algorithm 2: $\nabla^2 \varphi = f$ 的 V 循环多重网格方法

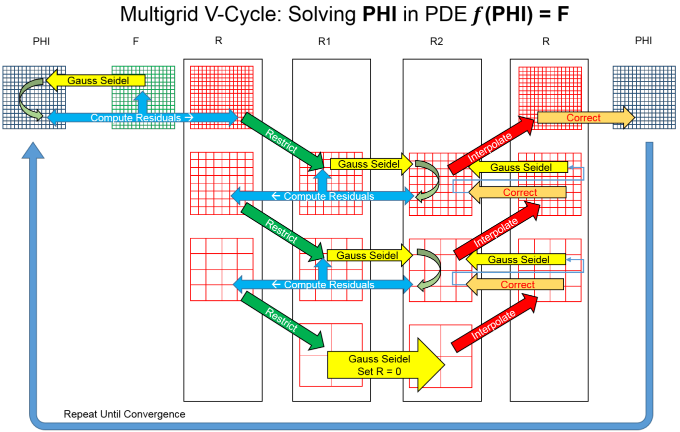


图 2: V 循环多重网格方法的示意图

基础算法构成

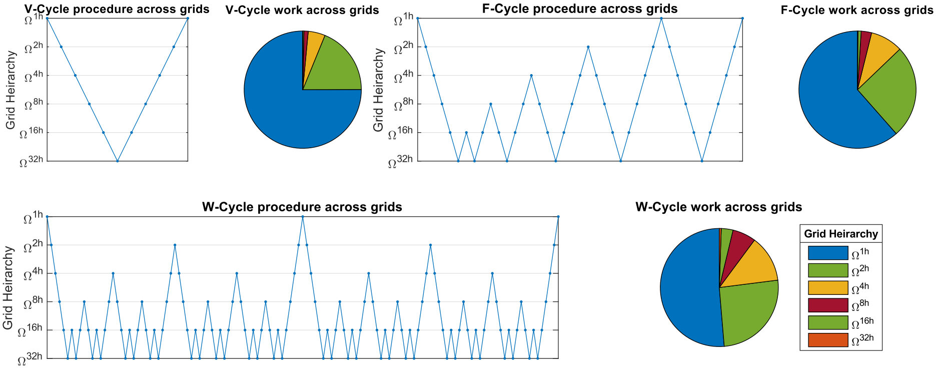


图 3: 三种多重网格方法的迭代图

Subsection 2

求解泊松方程的举例

利用多重网格-五点差分-共轭梯度法和直接共轭梯度方法（只在一级网格中进行迭代）分别对 (8a) 式进行求解，设置求解区域为 $S = \{(x,y) | -1 \le x \le 1, -1 \le y \le 1\}$ 的正方形区域，边界条件设置为 (8b)(8c) 式，在两个维度上的网格精度均为 10^{-3}，即求解区域的网格大小为 2000×2000，

$$
\begin{cases}
\dfrac{\partial^2 \varphi}{\partial x^2} + \dfrac{\partial^2 \varphi}{\partial y^2} = -\dfrac{\rho}{\varepsilon} = -2\pi^2 \sin(\pi x)\sin(\pi y) & \text{(8a)} \\
\varphi = 0,\ x = \pm 1 & \text{(8b)} \\
\varphi = 0,\ y = \pm 1 & \text{(8c)}
\end{cases}
$$

该系统具有解析解，并可推导得到电场强度矢量的表达式，

$$
\varphi = \sin(\pi x)\sin(\pi y) \qquad E = -\pi\cos(\pi x)\sin(\pi y)\vec{e_x} + \pi\sin(\pi x)\cos(\pi y)\vec{e_y}
$$

图 4: 多重网格-五点差分-共轭梯度法的求解结果

多重网格法的后平滑迭代过程在三级网格中经历了 1 次，在二级网格中经历了 7 次，在一级网格中经历了 43 次，平均总计耗时 6.89692s；直接共轭梯度法经历 55 次迭代，平均总计耗时 10.2639s。

图 5: 两种算法的误差下降曲线