用 ChebyShev 时间积分法求解 TDSE

1 算法原理

二维空间中，单粒子的薛定谔方程为

$$i\hbar\frac{\partial}{\partial t}\psi = -\frac{\hbar^2}{2m}\nabla^2\psi + V(x,y)\psi \tag{1}$$

调整量纲使得 $\hbar = 1$。基于最基本的五点差分法，我们可以将其离散化为

$$\psi_{i,j}^{n+1} - \psi_{i,j}^{n} = \frac{i\tau}{2mh^2}\left(\sum_{a,b}\psi_{a,b}^{n} - 4\psi_{i,j}^{n}\right) - i\tau V_{i,j}\psi_{i,j}^{n} \tag{2}$$

其中求和号表示对 $\psi_{i,j}$ 的最近邻求和。很明显它不保持

$$\int \rho(x,y)dxdy \to h^2\sum_{i,j}\psi_{i,j}^{*}\psi_{i,j} \tag{3}$$

不变，这意味着总粒子数改变了，不符合物理要求。为此，我们可以改用演化方程的形式解出薛定谔方程

$$\psi(\boldsymbol{r},t) = e^{-iHt}\psi(\boldsymbol{r},0) \tag{4}$$

记 $c_{i,j}\psi_{i',j'} = \delta_{ii'}\delta_{jj'}, c_{i,j}^{\dagger}|0\rangle = \left|\psi_{i,j}\right\rangle$，上式中

$$H_{i,j}^{i',j'} = \frac{-1}{2mh^2}\left(c_{i-1,j}^{\dagger}c_{i',j'} + c_{i+1,j}^{\dagger}c_{i',j'} + c_{i,j-1}^{\dagger}c_{i',j'} + c_{i,j+1}^{\dagger}c_{i',j'} - 4c_{i,j}^{\dagger}c_{i',j'}\right) + V_{i,j}c_{i,j}^{\dagger}c_{i',j'} \tag{5}$$

由于 e^{-iHt} 是幺正算符，保持总概率不变，故我们只要找到合适的办法计算矩阵指数来逼近它，就能得到一个保酉的算法。计算矩阵指数的方法有 Trotter-Suzuki 法和 Chebyshev 时间积分法等，这里选择后者。

Chebyshev 时间积分的思想是，找到一个合适的多项式逼近指数函数。我们知道，$e^{\beta t}$ 可以展成泰勒级数

$$e^{\beta t} = \sum_{n=0}^{\infty}\frac{\beta^n}{n!}t^n \tag{6}$$

它是收敛的，但我们在实际编程中必须把它截断；不幸的是，它的收敛速度很慢，而且只在 $t = 0$ 附近精确度较高，$n!$ 增长得又太快，不利于计算。因此，我们要找到另一个收敛较快的级数。这里，如果限制 $|t| < M$，就可以使用最佳平方逼近的办法。

选用切比雪夫多项式 T_n 作为正交基，定义内积为

$$\langle f,g\rangle = \int_{-1}^{1} f^{*}(x)g(x)\frac{1}{\sqrt{1-x^2}}dx \tag{7}$$

在 $t \in [-1,1]$ 上平方逼近：

$$\min \quad L = \left| e^{\beta t} - \sum_n c_n T_n(t) \right|^2 \tag{8}$$

故对每一个系数，有

$$\frac{\partial L}{\partial c_m} = -2\left\langle e^{\beta t} - \sum_n c_n T_n(t), T_m(t) \right\rangle = 0 \tag{9}$$

得

$$\left\langle e^{\beta t} - \sum_n c_n T_n(t), T_m(t) \right\rangle = \langle e^{\beta t}, T_m(t)\rangle - \sum_n c_n \langle T_n(t), T_m(t)\rangle = 0 \tag{10}$$

由切比雪夫多项式的正交性，

$$\langle T_m, T_n \rangle = \begin{cases} 0, & m \neq n \\ \dfrac{\pi}{2}, & m = n \neq 0 \\ \pi, & m = n = 0 \end{cases} \tag{11}$$

得

$$\begin{aligned} c_0 &= \frac{1}{\pi}\langle e^{\beta t}, T_0(t)\rangle \\ c_m &= \frac{2}{\pi}\langle e^{\beta t}, T_m(t)\rangle \quad (m \neq 0) \end{aligned} \tag{12}$$

计算积分：

$$\int_{-1}^{1} e^{\beta t} T_m(t) \frac{1}{\sqrt{1-t^2}} dt \xlongequal{t=\cos\theta} \int_0^{\pi} e^{\beta \cos\theta} T_m(\cos\theta) d\theta = \int_0^{\pi} e^{\beta\cos\theta} \cos(m\theta) d\theta \tag{13}$$

虚宗量贝塞尔函数的积分定义恰好有

$$\alpha I_n(\beta) = \frac{1}{\pi}\int_0^{\pi} \cos(nx) e^{\beta\cos x} dx \tag{14}$$

故

$$c_0 = I_0(\beta), \qquad c_m = 2I_m(\beta) \quad (m \neq 0) \tag{15}$$

因此，$e^{\beta t}$ 在 $[-1,1]$ 上的最佳平方逼近为

$$e^{\beta t} = I_0(\beta)T_0(t) + 2\sum_{n=1}^{\infty} I_n(\beta) T_n(t) \tag{16}$$

其收敛区域为整个复平面。由 T_n 的递推式可知，在相同的截断次数下，最佳平方逼近所需乘法次数与泰勒展开相等。如图，此时最佳平方逼近的近似效果更好

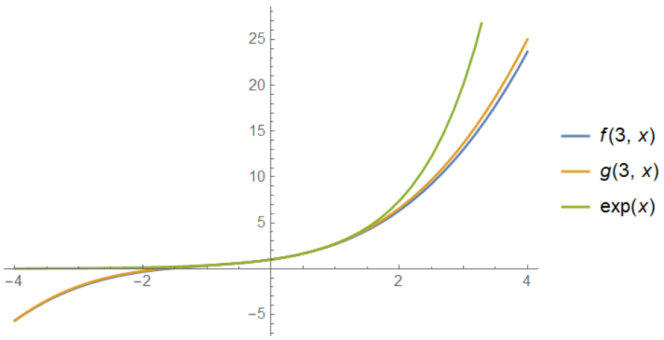


图 1: f 为泰勒级数，g 为最佳平方逼近

将这个结论推广到矩阵（厄米算符）的切比雪夫多项式，有

$$e^{-iHt} = I_0(-it)T_0(H) + 2\sum_{n=1}^{\infty} I_n(-it) T_n(H) \tag{17}$$

由整数阶虚宗量贝塞尔函数的奇偶性 $I_n(-x) = (-)^n I_n(x)$，以及 $I_n(x) = (-i)^n J_n(ix)$，得

$$e^{-iHt} = J_0(-it)T_0(H) + 2\sum_{n=1}^{\infty}(-i)^n J_n(t)T_n(H) \tag{18}$$

由此得离散化后的迭代公式为

$$\psi^{n+1} = J_0(\tau)T_0(H)\psi^n + 2\sum_{n=1}^{\infty}(-i)^n J_n(\tau)T_n(H)\psi^n \tag{19}$$

其中矩阵乘法 $H\psi = \sum_{i',j'} H_{i,j}^{i',j'}\psi_{i',j'}$ 按 (5) 式计算。

由 $-1 \le \cos(\theta) \le 1$ 知，$T_n(x)$ 将 $[-1,1]$ 映射到它自身，因此当 $t \in [-1,1]$ 时，$\left|I_n(\beta)T_n(t)\right|$（或 $\left|J_n(\beta)T_n(t)\right|$）的收敛速度将快于指数。考虑将这个结论推广到矩阵。由于 H 是厄米算符，设 Λ 为实对角矩阵，$H = U^\dagger \Lambda U$，则

$$T_n(H) = U^\dagger T_n(\Lambda)U \tag{20}$$

其中 $T_n(\Lambda)$ 对 Λ 的每个对角元取切比雪夫多项式。可见，收敛速度取决于 H 的模长最大的本征值。厄米矩阵的本征值 λ 有

$$|\lambda| \le \|H\|_\infty = \max_{i',j'}\sum_{i,j} H_{i,j}^{i',j'} \tag{21}$$

特别地，当 $V = 0$ 时，$|\lambda| \le \frac{4}{mh^2}$。要保证 $|\lambda| \le 1$，只要预先做替换

$$H \to \frac{H}{\|H\|_\infty}, \qquad \tau \to \|H\|_\infty \tau \tag{22}$$

即可。

2 编程要点

$\psi_{i,j}$ 包含整个平面网格上的波函数值，假设网格的规模为 1024×1024，数据为 double 型复数值，则存储一个就要占用 16MB 的内存，而 H 的大小更是达到 16TB！因此，H 不能显式地存储在内存当中，必须以函数指针的形态出现，即

$$H^n\psi = H(H(...H(\psi))) \tag{23}$$

其中，设某种空间差分格式 C 将 ψ 映射到一步迭代以后，则相应的 $H = C - I$，即 $H(\psi) = C(\psi) - \psi$。切比雪夫多项式的递推关系有

$$T_{n+1}(H)\psi = 2H\left[T_n(x)\psi\right] - T_{n-1}(H)\psi \tag{24}$$

这意味着我们要开 3 个缓存，来存储相邻三阶的结果。每次迭代后，下一阶的结果将顶替上一阶在缓存中的位置。但是，如此巨大的内存，无论是移动、复制还是擦除，都会造成严重的性能浪费。这启发我们设立一个简单的内存池：在所有计算开始时预先分配大块内存，代表 ψ 的对象构造时，须向内存池申请空间，内存池返回指针，并将此处标记为占用；当 ψ 复制时，只传递指针而无需通知内存池；当 ψ 参与运算时，传递函数指针通知内存池进行相应运算；当 ψ 析构时，内存池将空间标记为空闲即可。这样一来，内存池持有大量内存，但从不移动、复制和擦除，而 ψ 实际占用的空间很小，可以随意进行这些操作。内存池还有助于排查内存泄漏。

由 (19) 式中的 $(-i)^n$ 知，ψ 可能的相位有 4 种。若在分别存储 ψ 的实部和虚部的同时，另设一变量代表其相位，直到做加减法时，再按照相位分类讨论处理，可以免于对大量数据执行浮点数乘以-1 的操作。但这样一来，计算实部和虚部的演化不得不成为串行的两步。为此，可以将实

部和虚部存储在同一个矩阵里，例如，在初始化时，奇数位存虚部，偶数位存实部，之后由相位负责解释谁代表虚部，谁代表实部，谁的加法必须按减法执行，等等。

另外，单次迭代很适合并行运算，可使用 CUDA 显卡编程加速运算：在.cu 文件中将内核需要执行的函数封装成 extern "C" 接口，并在.cuh 文件中声明以备调用。（**例外：cudaMalloc 函数不可封装为接口，必须在 cpp 文件中直接调用，否则后面的 cudaMemcpy 都会失败！**）

综上，我们需要三个层次的对象来管理数据，其可能的定义和部分方法简述如下（与所附代码中的具体实现略有不同）：

```
class GpuMat{
    double* field; //内存池返回的指针，指向被cudaMalloc分配过的显存
    int loc; //在内存池中的相对位置，以便申请删除
};
class Psi{
    Pool* pool;
    GpuMat* mat;
    char phase; //相位有 1 i -1 -i 四种，加减法也应有四种实现（对应四种相位差）
    Psi H(){...} //基本运算在这一级实现
    Psi operator-(Psi&){...}
    void die(){...}
};
class Solver{
    Psi* psi;
    double* psi_cpu; //数据输入输出的缓冲区
};
```

内存池类的定义如下：

```
class Pool{
    double* head; //大片内存的首地址（从不移动）
    double* ptrs; //申请时返回其中的一个指针
    bool* occupied; //标记上面的指针中那些有数据填充
};
```

计算切比雪夫多项式的缓存则由 Chebyshev 对象自己保管：

```
class Cheby{
    Psi x0, x1, x2;
    void die(){...}
};
```

注意任何直接或间接地持有 GpuMat 的函数都要实现一个 die() 方法——它们自己的析构函数只会销毁指针，以防数据被意外删除；而当真正希望清除数据时，要手动联系内存池释放空间。

曾发生过迭代仍在进行但结果不变化的异常，经查，cudaMalloc 分配过的指针莫名其妙变成 NULL，但其错误码又是 Success！推测可能的原因是显存占满，但显存泄漏不易排查，可使用条件编译，在 CPU 端执行类似的串行代码排查内存泄漏，再类比到 GPU 的有关操作上。

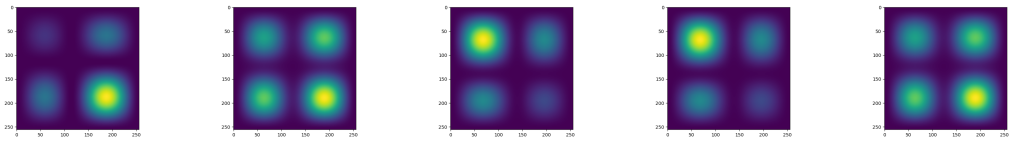
3 性能测试

首先将函数指针 $H(\cdot)$ 替换成恒等变换，初始化 $\psi \equiv 1$，即可计算 $\cos(n), \sin(n), (n \in \mathbb{N}_+)$ 的值（封装在 test() 中），并与标准值比较，验证 CUDA 的读写正常，Chebyshev 时间积分的流程正确，内存池、相位标记与加减法运行正常。

在 256×256 网格上，记 $f(x,t) = e^{-i\eta\pi^2 t}\sin 2\pi x - 3e^{-i4\eta\pi^2 t}\sin 2\pi x$，计算叠加态

$$\psi(x,y,0) = f(x,0)f(y,0), \qquad (x,y) \in [0,1]\times[0,1] \tag{25}$$

的演化，其解析解为 $\psi(x,y,t) = f(x,t)f(y,t)$，周期为 $\frac{2}{3\eta\pi}$，其中 $\eta = 1/2m$，若预先设置扩散系数为 $D = 1/2mh^2$（例如，0.1），则 $\eta = Dh^2$。如图所示：



采用 20 阶 Chebyshev 时间积分，每次迭代的步长为 10^6 周期（每 500 步输出一次）。结果在定性上与上图相同；区域内概率密度积分的变化如图所示：

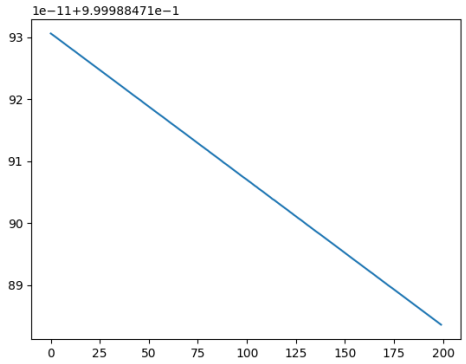


图 2: 横轴为迭代次数

可见，该算法确实是保酉的，总概率的变化只有 10^{-11} 量级。侧面观如图所示：

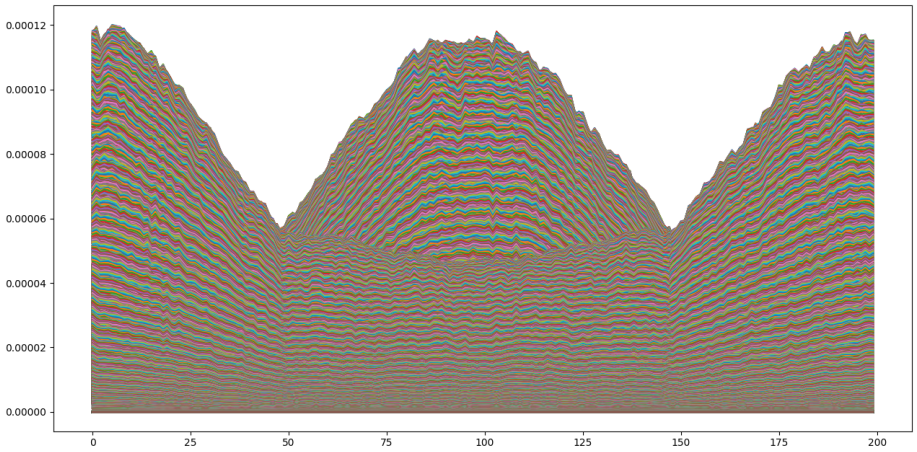


图 3: 横轴为迭代次数

可见，数值解不如解析解光滑，但其单调性较好，局部波动不如同等网格和时间步长的 Trotter-Suzuki 法（二阶）大。（高阶 Trotter-Suzuki 法的局部波动可能变得更大。）

以下以概率密度 $\rho(x, y)$ 在绝对误差最大处的相对误差来衡量算法的误差。在网格的中心区域和靠近边缘的区域各抽样两点，分别计算数值解与解析解，如图所示：

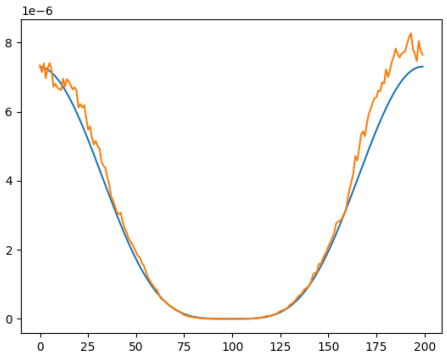


图 4: (140,140)

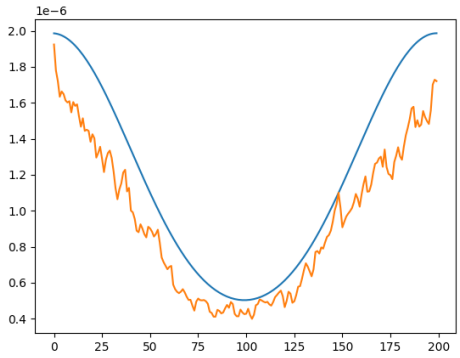


图 5: (240,240)

可见中心区域的计算效果较好。以求解区域中心点附近的 128×128 区域为中心区，分别计算所有时刻的相对误差，如图所示：

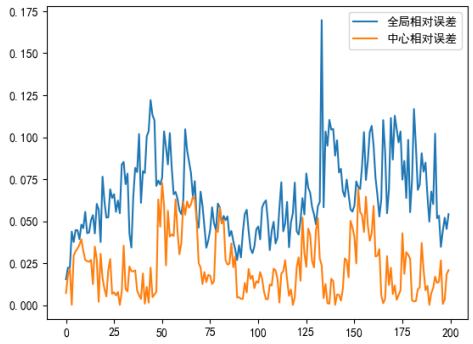


图 6: Chebyshev 法

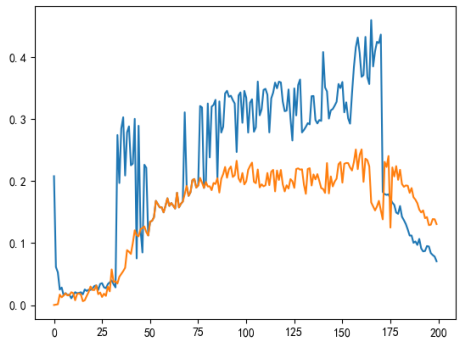


图 7: Trotter-Suzuki 法

可见，Chebyshev 法的误差比 Trotter-Suzuki 法小一个数量级。

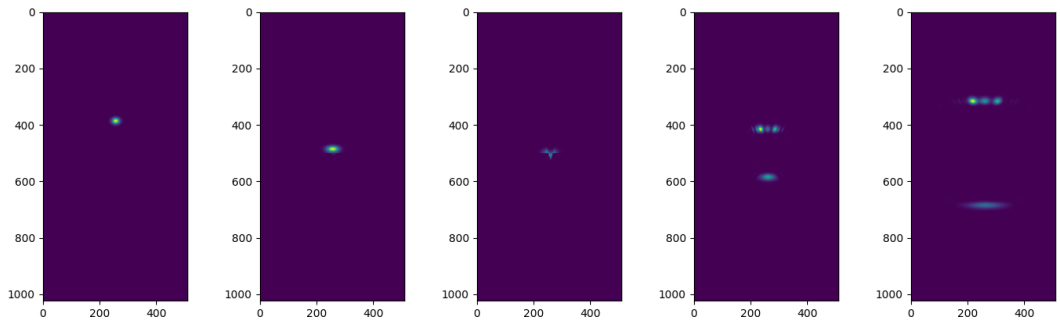
Chebyshev 时间积分法相对于 Trotter-Suzuki 法的优势在于，在每个时间步，$H^n\psi$ 项将一点处的信息传到 L^1 距离为 n 格远的地方，体现了薛定谔方程的抛物性，而不是只有最近邻两格发生相互作用，具有双曲性。但是在边界处，信息向远处传播可能受阻。若边界处的网格不划分得更密，近似效果就会比较差了。

运行速度的瓶颈主要在于 GPU 端到 CPU 端的数据传输，GPU 的占用反而很小。

4 结果

4.1 单缝衍射 1

设动量 $p = 512\pi$（采样定理所容许的最大动量），波包初始宽度 $\sigma = 0.02$。由于采用将其所在区域的波函数直接置零来模拟不可穿透的墙壁，在粒子通过波导的几帧里，总概率迅速下降，所幸下降不多，只有 3.4%。结果如图所示：



可见，当粒子接近墙壁时，波包受斥力而变扁；有一定概率反射或透射，反射波包分为若干束；透射波和反射波都在一锥形区域内，将所有帧的概率密度叠加可见：

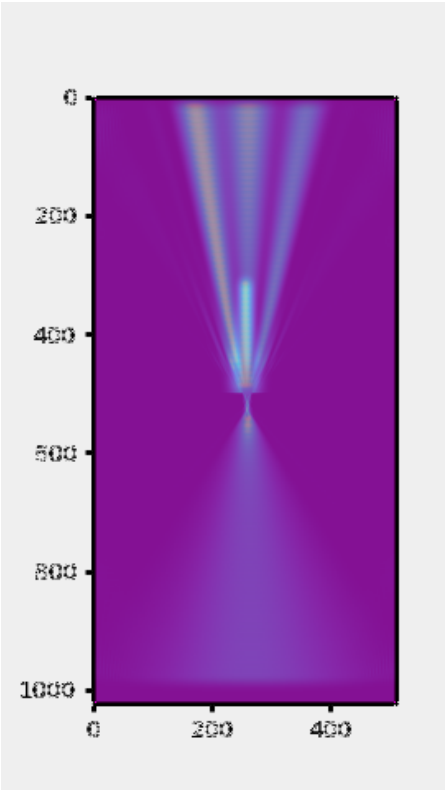


图 8: 长时间曝光效果

全过程详见 res1.mp4。

4.2 单缝衍射 2

设动量 $p = 256\pi$，波包初始宽度 $\sigma = 0.1$，结果详见 res2.mp4。

4.3 双缝干涉

结果详见 res3.mp4。