# 计算流体力学作业二

黄晃 数院 1701210098

2018 年 6 月 7 日

## 1 问题

$[-\pi, \pi]^3$ 中周期边值的不可压流体的计算. 考虑 3D 涡-向量势公式

$$\begin{cases} \dfrac{\partial \omega}{\partial t} + \nabla \times (\omega \times \mathbf{u}) = \nu \triangle \omega \\ \omega = \nabla \mathbf{u}, \nabla \cdot \mathbf{u} = 0 \\ \mathbf{u}(\mathbf{x}, 0) = \mathbf{u}_0(\mathbf{x}). \end{cases}$$

### 1.1 数值算法思路

以一阶 Euler 为例 (之后替换成四阶显示 Runge-Kutta 方法):

- 解 $-\triangle_h \psi^n = \omega^n$
- 计算 $\mathbf{u}^n = \nabla_h \psi^n$
- 通过 $\frac{\omega^{n+1} - \omega^n}{\tau} + \nabla_h \times (\omega^n \times \mathbf{u}^n) = \nu \triangle \omega^n$ 计算 $\omega^{n+1}$

### 1.2 四阶 Runge-Kutta

记 $f(\omega) = \nu \triangle \omega - \nabla_h \times (\omega \times \mathbf{u})$ 则半离散问题为

$$\frac{d\omega}{dt} = f(\omega)$$

四阶 Runge-Kutta 方法可以写作

$$\begin{aligned}\omega^{n+1} &= \omega^n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)\\ k_1 &= \tau f(\omega^n),\\ k_2 &= \tau f(\omega^n + \frac{k_1}{2}),\\ k_3 &= \tau f(\omega^n + \frac{k_2}{2}),\\ k_4 &= \tau f(\omega^n + k_3).\end{aligned}$$

# 2 伪谱方法

对于周期函数 f(x), 其傅里叶展开为

$$f(x) = \sum_{m=-\infty}^{\infty} \hat{f}_m e^{1jmx},\ x \in (-\pi, \pi)$$

其中

$$\hat{f}_m = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(x)e^{-1jmx}dx.$$

伪谱法实际上相当于考虑 f 在空间 $S_n$ 中的投影

$$I_n(f(x)) = \sum_{m=-n}^{n-1} \hat{f}_m e^{1jm)x},\ x \in (-\pi, \pi)$$

**基函数** 取插值节点为 $x_{i,j,k} = (\frac{(i+1/2)\pi}{n}, \frac{(j+1/2)\pi}{n}, \frac{(j+1/2)\pi}{n})$ $i, j, k = -n, -n+1, \cdots, n-1$ 那么有

$$I_n(f(x)) = \sum_{i,j,k=0}^{2n-1} f(x_{i,j,k})g_{i,j,k}(x),\ x \in (-\pi, \pi)$$

其中 $g_{i,j,k}(x)$ 是 $S_n$ 中满足 $g_{i,j,k}(x_{i1,j1,k1}) = \delta_{(i,j,k)(i1,j1,k1)}$ 的三角多项式.

利用正交性, 我们有

$$g_{i,j,k}(x) = \frac{1}{(2n)^3}\sum_{p,l,m=-n}^{n-1} e^{j(p,l,m)(x-x_{i,j,k})}$$

为了简洁起见, 下面在不引起混淆的情况下, 用 p 代替 (p,l,m), 用 i 代替 (i,j,k).

因此我们有

$$
\begin{aligned}
I_n f(x) &= \sum_{i=0}^{2n-1} f(x_i) \frac{1}{(2n)^3} \sum_{p=-n}^{n-1} e^{jp(x-x_i)} \\
&= \sum_{p=-n}^{n-1} e^{jp(x+\pi)} \frac{1}{(2n)^3} \sum_{i=0}^{2n-1} f(x_i) e^{-jp(x_i+\pi)}
\end{aligned} \tag{1}
$$

若我们记

$$\hat{f}_p = \frac{1}{(2n)^3} \sum_{i=0}^{2n-1} f(x_i) e^{-jp(x_i+\pi)}$$

则

$$I_n(f(x)) = \sum_{p=-n}^{n-1} \hat{f}_p e^{jp(x+\pi)}$$

注意到在节点 $x_i$ 上有 $I_n f(x_i) = f(x_i))$ 成立, 所以我们建立 $f_i$, $i = 0, 1, \cdots, 2n-1$ 与 $\hat{f}_p$, $p = -n, -n+1, \cdots, n-1$ 之间的一个一一映射.

将 $x_i$ 的值代入, 我们有

$$
\begin{aligned}
f_i &= \sum_{p=-n}^{n-1} \hat{f}_p e^{jp(i+1/2)\pi/n},\ i = -n, -n+1 \cdots, n-1 \\
\hat{f}_p &= \frac{1}{(2n)^3} \sum_{i=0}^{2n-1} f(x_i) e^{-jp(i+1/2)\pi/n},\ p = -n, -n+1 \cdots, n-1
\end{aligned} \tag{2}
$$

投影 $I_n(f)$ 写成

$$I_n f(x) = \sum_{p=-n}^{n-1} \hat{f}_p e^{jpx}$$

## 2.1 1 维情况下对奇偶性质的保持

**偶函数** 对 1 维的长为 2n 的偶序列 $\{f_{-n}, f_{-n+1}\cdots, f_{n-1}\}$, 即 $f_k = f_{-1-k}$, 有

$$
\begin{aligned}
\hat{f}_p &= \frac{1}{2n}\sum_{i=-n}^{n-1} f_i e^{-jp(i+1/2)\pi/n} \\
&= \frac{1}{2n}\sum_{i=0}^{n-1} f_i e^{-jp(i+1/2)\pi/n} + f_{-1-i}e^{-jp(-1-i+1/2)\pi/n} \\
&= \frac{1}{2n}\sum_{i=0}^{n-1} f_i (e^{-jp(i+1/2)\pi/n} + e^{-jp(-i-1/2)\pi/n}) \\
&= \frac{1}{2n}\sum_{i=0}^{n-1} 2f_i\cos(p(i+1/2)\pi/n)
\end{aligned}
$$

所以有 $\hat{f}_p = \hat{f}_{-p}$. 且 $\hat{f}_{-n} = \frac{1}{2n}\sum\limits_{i=0}^{n-1} 2f_i\cos(-(i+1/2)\pi) = 0$

而对 $\hat{f}_p = \hat{f}_{-p}$ 的频谱, 假设其满足 $\hat{f}_{-n} = 0$, 则其逆变换的结果 $f_i$ 有

$$
\begin{aligned}
f_i &= \sum_{p=-n}^{n-1} \hat{f}_p e^{jp(i+1/2)\pi/n} \\
&= \sum_{p=1}^{n-1} \hat{f}_p e^{jp(i+1/2)\pi/n} + \hat{f}_{-p}e^{j(-1-p)(i+1/2)\pi/n} + \hat{f}_0 \\
&= \sum_{p=1}^{n-1} \hat{f}_p (e^{jp(i+1/2)\pi/n} + e^{j(-1-p)(i+1/2)\pi/n}) + \hat{f}_0 \\
&= \sum_{p=1}^{n-1} \hat{f}_p 2\cos(p(i+1/2)\pi/n) + \hat{f}_0
\end{aligned}
$$

则 $f_k = f_{-1-k}$ 成立

**奇函数** 对于奇序列, 即 $f_k = -f_{-1-k}$,

$$\begin{aligned}
\hat{f}_p &= \frac{1}{2n}\sum_{i=-n}^{n-1} f_i e^{-jp(i+1/2)\pi/n} \\
&= \frac{1}{2n}\sum_{i=0}^{n-1} f_i e^{-jp(i+1/2)\pi/n} + f_{-1-i}e^{-jp(-1-i+1/2)\pi/n} \\
&= \frac{1}{2n}\sum_{i=0}^{n-1} f_i(e^{-jp(i+1/2)\pi/n} - e^{-jp(-i-1/2)\pi/n}) \\
&= \frac{1}{2n}\sum_{i=0}^{n-1} -2jf_i \sin(p(i+1/2)\pi/n)
\end{aligned}$$

其中 $\hat{f}_0 = 0$, 我们实际存储 $\{b_0, b_1, \cdots, b_{n-1}\} = \{1j\hat{f}_1, 1j\hat{f}_2, \cdots, 1j\hat{f}_{n-1}, -1j\hat{f}_{-n}\}$ 则有 $b_p = \frac{1}{2n}\sum_{i=0}^{n-1} 2f_i \sin((p+1)(i+1/2)\pi/n)$

而对 $\hat{f}_p = -\hat{f}_{-p}$ 的频谱, 假设其满足 $\hat{f}_0 = 0$, 则其逆变换的结果 $f_i$ 有

$$\begin{aligned}
f_i &= \sum_{p=-n}^{n-1} \hat{f}_p e^{jp(i+1/2)\pi/n} \\
&= \sum_{p=1}^{n-1} \hat{f}_p e^{jp(i+1/2)\pi/n} + \hat{f}_{-p}e^{j(-1-p)(i+1/2)\pi/n} + \hat{f}_{-n}e^{-j(i+1/2)\pi} \\
&= \sum_{p=1}^{n-1} \hat{f}_p(e^{jp(i+1/2)\pi/n} - e^{j(-1-p)(i+1/2)\pi/n}) + (-1)^{i-1} * 1j\hat{f}_{-n} \\
&= \sum_{p=1}^{n-1} 2j\hat{f}_p \sin(p(i+1/2)\pi/n) + (-1)^i * (-1j)\hat{f}_{-n} \\
&= \sum_{p=1}^{n-1} 2b_{p-1} \sin(p(i+1/2)\pi/n) + (-1)^i b_{n-1} \\
&= \sum_{p=0}^{n-2} 2b_p \sin((p+1)(i+1/2)\pi/n) + (-1)^i b_{n-1}
\end{aligned}$$

则 $f_k = -f_{-1-k}$ 成立

## 2.2 FFTW

提供了 1 维的实奇 (偶) 函数的 DFT(Type-2,3DCT,DST)

$$\begin{aligned} REDFT01:\ &(a_0,a_1,\cdots,a_{n-1})\rightarrow(b_0,b_1,\cdots,b_{n-1})\\ b_p=\ &2\sum_{i=0}^{n-1}a_i\cos(p(i+1/2)/n),\ p=0,1\cdots,n-1 \end{aligned} \tag{3}$$

$$\begin{aligned} REDFT10:\ &(a_0,a_1,\cdots,a_{n-1})\rightarrow(b_0,b_1,\cdots,b_{n-1})\\ b_p=\ &a_0+\sum_{i=1}^{n-1}a_i\cos((p+1/2)i/n),\ p=0,1\cdots,n-1 \end{aligned} \tag{4}$$

$$\begin{aligned} RODFT01:\ &(a_0,a_1,\cdots,a_{n-1})\rightarrow(b_0,b_1,\cdots,b_{n-1})\\ b_p=\ &2\sum_{i=0}^{n-1}a_i\sin((p+1)(i+1/2)/n),\ p=0,1\cdots,n-1 \end{aligned} \tag{5}$$

$$\begin{aligned} RODFT10:\ &(a_0,a_1,\cdots,a_{n-1})\rightarrow(b_0,b_1,\cdots,b_{n-1})\\ b_p=\ &(-1)^p a_{n-1}+\sum_{i=0}^{n-2}a_i\sin((p+1/2)(i+1)/n),\ p=0,1\cdots,n-1 \end{aligned} \tag{6}$$

对于 1 维情形, 令 N=2n, 对偶的序列有

$$\begin{aligned} [\hat f_0,\hat f_1,\cdots,\hat f_{n-1}]=\ &\frac{1}{2n}REDFT01([f_0,f_1,\cdots,f_{n-1}])\\ [f_0,f_1,\cdots,f_{n-1}]=\ &REDFT10([\hat f_0,\hat f_1,\cdots,\hat f_{n-1}]) \end{aligned} \tag{7}$$

对奇的序列有

$$\begin{aligned} [1j\hat f_1,1j\hat f_2,\cdots,\hat f_{n-1},-1j\hat f_{-n}]=\ &\frac{1}{2n}RODFT01([f_0,f_1,\cdots,f_{n-1}])\\ [f_0,f_1,\cdots,f_{n-1}]=\ &RODFT10([1j\hat f_1,1j\hat f_2,\cdots,\hat f_{n-1},-1j\hat f_{-n}]) \end{aligned} \tag{8}$$

## 2.3 三维 FFT 的实现

回到我们需要的三维情形

$$\begin{aligned}
f_{i,j,k} &= \sum_{p,l,m=-n}^{n-1} \hat{f}_{p,l,m} e^{j((p+1/2)(i+1/2)+(l+1/2)(j+1/2)+(m+1/2)(k+1/2))\pi/n} \\
&= \sum_{p=-n}^{n-1} \sum_{l=-n}^{n-1} \sum_{m=-n}^{n-1} \hat{f}_{p,l,m} e^{j((i+1/2)+(l+1/2)(j+1/2)+(m+1/2)(k+1/2))\pi/n} \\
&= \sum_{p=-n}^{n-1} \left[ \sum_{l=-n}^{n-1} \left( \sum_{m=-n}^{n-1} \hat{f}_{p,l,m} e^{j(m+1/2)(k+1/2))\pi/n} \right) e^{j(l+1/2)(j+1/2)\pi/n} \right] e^{j(p+1/2)(i+1/2)\pi/n}
\end{aligned} \tag{9}$$

$$\begin{aligned}
\hat{f}_{i,j,k} &= \frac{1}{(2n)^3} \sum_{p,l,m=-n}^{n-1} f_{p,l,m} e^{-j((p+1/2)(i+1/2)+(l+1/2)(j+1/2)+(m+1/2)(k+1/2))\pi/n} \\
&= \sum_{p=-n}^{n-1} \sum_{l=-n}^{n-1} \sum_{m=-n}^{n-1} f_{p,l,m} e^{-j((i+1/2)+(l+1/2)(j+1/2)+(m+1/2)(k+1/2))\pi/n} \\
&= \sum_{p=-n}^{n-1} \left[ \sum_{l=-n}^{n-1} \left( \sum_{m=-n}^{n-1} f_{p,l,m} e^{-j(m+1/2)(k+1/2))\pi/n} \right) e^{-j(l+1/2)(j+1/2)\pi/n} \right] e^{-j(p+1/2)(i+1/2)\pi/n}
\end{aligned} \tag{10}$$

可以看到, 三维的 DFT 其实就是按照三个方向依次进行一维的 FFT, 逆变换也类似.

## 2.4 混淆误差

对于非线性项 $\omega \times u$, 由于 $e^{1jkx_i}$ 关于 k 以 2n 为周期, 所以做积会有混淆误差的产生, 实验结果说明, 混淆误差极大的影响问题的计算, 会使 u 的范数迅速趋于无穷

**混淆误差的处理** 我们将频谱宽度加倍. 即对 $\omega, u$ 分别进行长度为 2n 的 fft, 然后在频谱的两边各补 n 个 0, 然后做 ifft.(实际等于在原空间内做了插值), 然后再做乘法, 接着对乘积做长为 4n 的 fft, 然后截取频谱的中间 n 位, 再做长为 2n 的 ifft 作为乘积的结果.

**对称性的保持** 我们原本展开的频谱 $k = -n, -n+1, \cdots, n-1$, 实际的补 0 过程中, 我们仍然保持对称性, 即使 $\hat{f}_n = hatf_{-n}, -hatf_{-n}$. 因为我们特殊的存储方式, 在实际计算中, 只需要对长为 n 的数组后直接加上 n 个 0 即可.

容易看到, 这样的操作不会破坏数据本身的 (反) 对称性质.

## 2.5 离散微分算子

$\Delta f$ 对于 $\Delta$ 算子, 我们有

$$\Delta I_n(f(x)) = \sum_{p=-n}^{n-1} -p^2 \hat{f}_p e^{jpx}$$

由此, 我们得到了 $f(x_i)$ 与 $\Delta f(x_i)$ 的 Fourier 系数的关系. 下面推导离散的算子

假设 f 的对称性为 $(\delta_1, \delta_2, \delta_3)$,$\delta = 1, -1$ 分别表示关于 x 对称与反对称,2,3 对应于 y,z.

在 $[-\pi, \pi]^3$ 内一共 $(2n)^3$ 的点, 三维 DFT 由三个方向的 DFT 复合而成, 由 DFT 的线性性质, 我们知道对 x 方向的变换, 不会改变 y,z 方向上的对称性.

- 若 $\delta_1 = 1$, 则每一行关于 x 的 DFT 可由上面的 DCT 运用在前 n 个点上得到, 结果为 $[\hat{f}_0, \hat{f}_1, \cdots, \hat{f}_{n-1}]$
- 若 $\delta_1 = 1$, 则每一行关于 x 的 DFT 可由上面的 DOT 运用在前 n 个点上得到, 结果为 $[1j\hat{f}_1, 1j\hat{f}_2, \cdots, 1j\hat{f}_{n-1}, -1j\hat{f}_{-n}]$

需要注意的是, 两种情况我们都可以从中恢复整个频谱. 然后对得到数据继续 y,z 上的操作, 最后得到的结果的 (i,j,k) 位置实际为 f 的三维的傅里叶系数的频率为 $(\phi(i,\delta_1), \phi(j,\delta_2), \phi(k,\delta_3))$ 的系数信息

$$\phi(i,\delta) = \begin{cases} & i,\ \delta = 1 \\ \begin{cases} i+1,\ i < n-1 \\ -n,\ i = n-1 \end{cases} & \delta = -1 \end{cases}$$

然后在每个位置乘对应的 $-\phi(i,\delta_1)^2 - \phi(j,\delta_2)^2 - \phi(k,\delta_3)^2$,(在没有显式存储的位置也乘以对应的因子) 易见这样不会改变频谱每个方向上的对称性, 所以直接对此时所存储的频率做 z 方向的逆变换，相当于对 $\Delta f$ 的傅里叶系数做 z 方向的逆变换, 之后对 y,x 有相同的结果. 所以最终得到的是 $\Delta f_i, j, k$

**$\partial_x f$** 对于 1 维情况下的求导, 我们有

$$\partial I_n(f(x))/\partial x = \sum_{p=-n}^{n-1} 1jp\hat{f}_p e^{jpx}$$

只用考虑一行的 DFT 即可, 实际的数据列为 $f_{-n}, f_{-n+1}, \cdots, f_{n-1}$, 存储的仅有 $f_0, f_1, \cdots, f_{n-1}$. 分奇偶两种情况

**偶的情形** f 为偶, 即 $f_i = f_{-1-i}$ 时,DFT 结果为 $[\hat{f}_0, \hat{f}_1, \cdots, \hat{f}_{n-1}]$, 且 $\hat{f}_i = \hat{f}_{-i}, \hat{f}_{-n} = 0$, 乘以 p 后得到 $f^0 = \partial I_n(f(x))/\partial x$ 的傅里叶系数 $\frac{\hat{f^0}_p}{1j}$, 容易看出 $\hat{f^0}_i = -\hat{f^0}_{-i}$, 是一个满足 $\hat{f^0}_0 = 0$ 奇序列.

要从中得到 $f_0^0, f_0^0, \cdots, f_{n-1}^0$, 我们需要对 $[1j\hat{f^0}_1, 1j\hat{f^0}_2, \cdots, \hat{f^0}_{n-1}, -1j\hat{f^0}_{-n}]$ 做 IDST

其中 $\hat{f^0}_{-n} = nj{*}\hat{f}_{-n} = 0$. 而我们存储的结果此时为 $[\hat{f^0}_0/1j, \hat{f^0}_1/1j, \cdots, \hat{f^0}_{n-2}/1j, \hat{f^0}_{n-1}/1j]$, 将其前移一位, 然后在最后补零, 再乘 $-1 = (1j)^2$, 则变为 $[1j\hat{f^0}_1, 1j\hat{f^0}_2, \cdots, 1j\hat{f^0}_{n-1}, 0] = [1j\hat{f^0}_1, 1j\hat{f^0}_2, \cdots, 1j\hat{f^0}_{n-1}, -1j\hat{f^0}_{-n}]$. 对其进行 IDST 则有 $f_i^0$.

**奇的情形** f 为奇, 即 $f_i = -f_{-1-i}$ 时,DFT 结果为 $[1j\hat{f}_1, 1j\hat{f}_2, \cdots, 1j\hat{f}_{n-1}, -1j\hat{f}_{-n}]$, 且 $\hat{f}_i = -\hat{f}_{-i}, \hat{f}_0 = 0$, 乘以 p(在存储的 n 个位置其实是乘 $[1, 2, \cdots, n-1, -n]$) 后得到 $f^0 = \partial I_n(f(x))/\partial x$ 的傅里叶系数 $\hat{f^0}_p$(频率-n 的项为 $-\hat{f^0}_{-n}$), 容易看出 $\hat{f^0}_i = -\hat{f^0}_{-i}$, 但是它不满足 $\hat{f^0}_{-n} = 0$, 所以不能直接通过 IDCT 来得到 $f_i^0$, 只能将 $\hat{f^0}_{-n}$ 强行设为 0 来完成. 会产生额外的误差. 但由于已知 $f^0$ 是个偶函数, 所以这样设置是合理的.

要从中得到 $f_0^0, f_0^0, \cdots, f_{n-1}^0$, 我们需要对 $[\hat{f^0}_0, \hat{f^0}_1, \cdots, \hat{f^0}_{n-1}]$ 做 IDCT, 存储的为 $[\hat{f^0}_1, \hat{f^0}_2, \cdots, \hat{f^0}_{n-1}, -\hat{f^0}_{-n}]$, 后移一位, 然后在最开始补零即可. 对其进行 IDCT 即可.

**note** 上面的操作, 都是对实数进行的, 所以我们的数据可以存储为实数

# 3 计算中对称性

## 3.1 初值的对称性

$$\mathbf{u}(\mathbf{x},0)=\begin{pmatrix}u(\mathbf{x},0)\\v(\mathbf{x},0)\\w(\mathbf{x},0)\end{pmatrix}=\begin{pmatrix}\sin x(\cos 3y\cos z-\cos y\cos 3z)\\\sin y(\cos 3z\cos x-\cos z\cos 3x)\\\sin z(\cos 3x\cos y-\cos x\cos 3y)\end{pmatrix}$$

**u** 中,u 关于 x 反对称, 关于 y,z 对称,2,3(v,w) 分量关于 x,y,z 轮换

$$\begin{aligned}\omega(\mathbf{x},0)&=\begin{pmatrix}\omega_1(\mathbf{x},0)\\\omega_2(\mathbf{x},0)\\\omega_3(\mathbf{x},0)\end{pmatrix}=\nabla\times\mathbf{u}(\mathbf{x},0)\\&=\begin{pmatrix}\partial_y w(\mathbf{x},0)-\partial_z v(\mathbf{x},0)\\\partial_z u(\mathbf{x},0)-\partial_x w(\mathbf{x},0)\\\partial_x v(\mathbf{x},0)-\partial_y u(\mathbf{x},0)\end{pmatrix}=\begin{pmatrix}-2\cos 3x\sin y\sin z+3\cos x(\sin 3z\sin y+\sin z\sin 3y)\\-2\cos 3y\sin z\sin x+3\cos y(\sin 3x\sin z+\sin x\sin 3z)\\-2\cos 3z\sin x\sin y+3\cos z(\sin 3y\sin x+\sin y\sin 3x)\end{pmatrix}\end{aligned}$$

$\omega$ 中,$\omega_1$ 关于 x 对称, 关于 y,z 反对称,2,3 分量关于 x,y,z 轮换

**$\psi$ 的确定** $-\triangle\psi=\omega$, 满足周期边值, 然后, 由于方程本身不关注 $\psi$ 的函数值, 而只是需要通过其求 $\mathbf{u}=\nabla\psi$, 所以可限制 $\psi$ 的傅里叶系数中常数项系数为 0. 我们断言 $\psi$ 的三个分量有与 $\omega$ 相同的对称性, 这会后面给予证明.

因此, 关于初始值, 我们有:$\mathbf{u}(\omega,\psi)$ 的第一个分量关于 x 反对称 (对称), 关于 y,z 对称 (反对称), 第二, 三个分量有类似的结果 (将 x,y,z 顺序轮换即可).

## 3.2 计算过程中对对称性的保持

空间上我们采用拟谱方法进行计算

**DFT 的性质** x(n) 是长度为 N 的序列,X(k)=DFT[x(n)], 则

- 若 x 纯实 (虚), 且 x 对称, 即 x(n)=x(N-1-n), 则 X(k)=X(N-1-k), 且纯实 (虚).

- 若 x 纯实 (虚), 且 x 反对称, 即 x(n)=-x(N-1-n), 则 X(k)=-X(N-1-k), 且纯虚 (实).

显然, 对于逆变换 IDFT, 也有一样的结果

### 3.2.1 DFT(IDFT) 对对称性的保持

对于 $u \in \mathbb{R}^{n\times n\times n}$,u 在 x,y,z 有对称性或者反对称性, 则对 u 的所有行做一次 DFT(IDFT), 由上面的性质可知, 关于 x 的 (反) 对称性是保持的, 此外, 由 DFT(IDFT) 的线性性质, 此时关于 y,z 的 (反) 对称性是保持的. 所以对单个方向做 DFT(IDFT) 不会破坏三个方向上的对称性

而一个三维 DFT(IDFT) 可以通过对三个方向依次做 DFT(IDFT) 实现, 所有三维情况也保持对称性

### 3.2.2 $\triangle$ 以及 $\triangle^{-1}$ 对对称性的保持

对于 $\triangle\omega$ 的第一个分量 $\triangle\omega_1$, 有

$$\triangle\omega_1 = -IDFT(K \circ DFT(\omega_1))$$

其中 $K(i,j,k) = \lambda(i)^2 + \lambda(j)^2 + \lambda(k)^2$

$$\lambda(i) = i + \frac{1}{2} - n$$

易见 K 关于 x,y,z 均是对称的, 所以 $K(K^{-1}) \circ u$ 不改变 u 的对称性. 其中 $K^{-1}$ 只会出现在解关于 $\psi$ 的 Poisson 方程,$\psi_1 = -\triangle^{-1}\omega_1 = -IDFT(K^{-1} \circ DFT(\omega_1))$, 由之前的分析, 可直接令 $\psi$ 的频率中对应 K 中 0 的项恒为 0, 所以是良定义且保持对称性的.

### 3.2.3 $\nabla \times \psi$ 对对称性的影响

假设 $\psi$ 具有与初值中所具有的对称性

$$\mathbf{u} = \begin{pmatrix} u \\ v \\ w \end{pmatrix} = \nabla \times \psi = \begin{pmatrix} \partial_y\psi_3 - \partial_z\psi_2 \\ \partial_z\psi_1 - \partial_x\psi_3 \\ \partial_x\psi_2 - \partial_y\psi_1 \end{pmatrix}$$

在这里 $\partial$ 也是通过 DFT 实现的, 首先考察 $\partial_x u$ 对对称性的影响

**$\partial_x u$ 对 u 对称性的影响** 我们知道 $\partial_x u = 1i * IDFT(M \circ DFT(u))$, 这里 DFT 以及 IDFT 都是一维的, 其中 $M(i,j,k) = \lambda(i)$, 所以 M 关于 x 反对称, 关于 y,z 对称, 因此, 有:$\partial_x u$ 保持 y,z 方向上的 (反) 对称, 同时将 x 方向的对称 (反对称) 改变为反对称 (对称).

回到 $\nabla \times \psi$, 第一个分量 $u = \partial_y \psi_3 - \partial_z \psi_1$ , 其中 $\psi_3$ 关于 z 对称, 关于 x,y 反对称;$\psi_2$ 关于 y 对称, 关于 z,x 反对称. 所以 $\partial_y \psi_3$ 关于 x 反对称, 关于 y,z 对称;$\partial_z \psi_1$ 关于 x 反对称, 关于 y,z 对称. 因此有 u 关于 x 反对称, 关于 y,z 对称. 类似可以推导出第 2,3 个分量的相应结果

因此有 $\nabla \times \psi$ 第一个分量具有与 $\psi_1$ 恰好相反的对称性 (只在在这一节的假设下成立).

#### 3.2.4 $\omega$ 对称性的保持

$\mathbf{u}, \psi$ 的对称性的保持已经在上面给予了说明, 下面考察 $\omega$. 由于时间上的离散采用的 Runge-kutta 方法, 所以只需要说明 $\nabla \times (\omega \times \mathbf{u}) - \nu \triangle \omega$ 具有和 $\omega$ 相同的对称性即可.

显然我们只需单独说明其中的非线性项 $\nabla \times (\omega \times \mathbf{u})$. 对于第一个分量, 有:

$$(\omega \times \mathbf{u})_1 = \omega_2 w - \omega_3 v$$

其中 $\omega_2$ 关于 y 对称, 关于 z,x 反对称,$\omega_3$ 关于 z 对称, 关于 x,y 反对称;v 关于 y 反对称, 关于 z,x 对称,w 关于 z 反对称, 关于 x,y 对称. 所以 $\omega_2 w$ 关于 x 反对称, 关于 y,z 对称.

所以 $(\omega \times \mathbf{u})_1$ 具有与 $\omega_1$ 相反的对称性, 类似可知第 2,3 个分量有相应的结果. 根据之前所推导的 $\nabla \times$ 的性质, 我们有 $\nabla \times (\omega \times \mathbf{u})$ 保持 $\omega$ 的对称性

### 3.3 三个分量关于 x,y,z 轮换的性质

对于初值我们还有

$$u(x,y,z) = v(z,x,y) = w(y,z,x)$$
$$\omega_1(x,y,z) = \omega_2(z,x,y) = \omega_3(y,z,x)$$
$$\psi_1(x,y,z) = \psi_1(z,x,y) = \psi_1(y,z,x)$$

我们断言这种轮换的性质会在计算中得到保持

首先易见 $\triangle, \triangle^{-1}$ 以及数乘, 加减都是保持三个分量的轮换的, 只需考察 $\times$ 的影响.

#### 3.3.1 $\nabla \times \mathbf{u}$

以 $\mathbf{u}$ 为例, 轮换性质可以写成

$$w(x,y,z) = u(z,x,y), v(x,y,z) = u(y,z,x).$$

这在 x,y,z 是离散点对的时候成立. 对于连续情况

$$\begin{aligned}\nabla \times \mathbf{u} &= \begin{pmatrix}\partial w/\partial y - \partial v/\partial z\\ \partial u/\partial z - \partial w/\partial x\\ \partial v/\partial x - \partial u/\partial y\end{pmatrix} = \begin{pmatrix}\partial u(z,x,y)/\partial y - \partial u(y,z,x)/\partial z\\ \partial u(x,y,z)/\partial z - \partial u(z,x,y)/\partial x\\ \partial u(y,z,x)/\partial x - \partial u(x,y,z)/\partial y\end{pmatrix}\\ &= \begin{pmatrix}\partial_3 u(z,x,y) - \partial_2 u(y,z,x)\\ \partial_3 u(x,y,z) - \partial_2 u(z,x,y)\\ \partial_3 u(y,z,x) - \partial_2 u(x,y,z)\end{pmatrix}\end{aligned}$$

易见连续的 $\nabla\times$ 算子是保持轮换性质的, 而我们利用 DFT 进行离散替代其中连续的 $\partial$ 时, 若将其中的 $u(x,y,z)$ 视为离散的 $n^3$ 个点的值, 显然有上面的等式依旧成立.

对于 $\omega \times \mathbf{u}$, 证明方式与上类似, 不再赘述.

# 4 算法设计

由于出现的所有向量的三个分量都具有轮换性质, 所以在计算中, 我们都只需要计算第一个分量. 然后利用数据的对称性, 我们只需要存储 1/8 个正方体, 选择存储 $[-\pi, 0]^3$ 的数据 (有 $(N+1)^3$ 个点), 然后仅在需要做 DFT 时, 我们每次将一个长 N+1 的数组利用 (反) 对称补全成 2N 的数组, 然后存储 (相) 频率空间中的前 N+1 位. 为了方便, 下面我们记 N=N+1 为整体的规模大小

## 4.1 并行部分

**进程数** 支持进程数为 $size^3$ 的所有选择, 但是要求 $size^2 \| N$ 要成立. 下面记 $n = \frac{N}{size}$

#### 4.1.1 子进程存储

我们将进程号 myid 表示成 $myid = myorder[0] * size^2 + myorder[1] * size + myorder[2]$, 用该进程存储整个正方体中 z 方向第 myorder[0] 层,y 方向第 myoder[1] 层,x 方向第 myorder[2] 层的一个小正方体的数据 U(存储 u),W(存储 $\omega_1$). 正方体规模为 n.

**FFT 所需的存储** 我们需要一个 x 方向长 N,y 方向长 n,z 方向高为 $\frac{n}{size}$ 的存储空间 B, 用来进行某一个方向的 fft 以及 ifft.A,B 的存储顺序均为 $x > y > z$, 即保证 x 方向数据是连续存储的.

**$\triangle, \triangle^{-1}$ 所需存储** 只需用到第一个分量, 所以不需要额外存储

**计算 $\omega \times \mathbf{u}$ 所需存储** 进行这个计算时, 之后还需要 $\omega$ 但是不在需要 $\mathbf{u}$, 所以可以覆盖 U, 但是得保留 W. 然后计算第一个分量理论上需要第二三个分量, 虽然我们可以通过轮换性质从第一个分量中得到, 但在并行中, 这里所需的数据需要从别的进程中获得 (下一节会详述), 所以我们需要额外提供 W2,W3,U2,U3 来存储接收到的数据.

不能用 U 来存储 U2,U3 中的某一个, 是因为, 接收的同时, 也在将 U 发给别的数据, 形成了一个环, 且必须在发送 U 后才允许改写 U. 但是可以将之后叉乘的结果存在 U 中.

计算 $\nabla \times \mathbf{u}, \nabla \times \psi$ 也类似, 除了 U2,U3 外, 不需要额外的存储.

此外, 我们将每次新的 $\psi_1$ 直接存储在 U 内, 然后再将 $\nabla \times \psi$ 的结果 u 存储在 U 内

**总存储** 综上, 每个进程中我们需要 6 个 $n^3$ 的正方体以及一个长方体 B. 共 $7n^3$ 的存储. 处理混淆误差时, 需要前面提到的对应的 7 个容器,n 变为 2n, 所以需要 $56n^3$ 的存储. 全部的存储要求为 $63n^3$

#### 4.1.2 子进程计算任务

数乘以及加减直接在每个子进程内完成相应的部分.

**单个方向 fft**

- 在进行 x 方向的 fft 时, 该进程将获取同处于 x 方向上的共 size 个进程中的 z 方向上第 myorder[2] 部分的数据
- 在进行 y 方向的 fft 时, 该进程将获取同处于 y 方向上的共 size 个进程中的 z 方向上第 myorder[1] 部分的数据
- 在进行 z 方向的 fft 时, 该进程将获取同处于 z 方向上的共 size 个进程中的 y 方向上第 myorder[0] 部分的数据

以 n=4,size=2 为例, 详述第 0 号进程的信息交互

- 在进行 x 方向的 fft 时, 进程 0 获取同处于 x 方向上进程 1 中最上两层的数据, 即 [:,:,0:1], 并将其直接接受在长方体 B 的 [n:2n-1,:,:] 部分, 同时将 0 进程中 A 的最上两层存入 B 的 [0:n-1,:,:] 中. 这样进程 0 中的 B 中就有了连续存储的 n/size*n 个长为 N 的向量, 正好是 x 方向上这 size 个进程中需要做 1 维 fft 的上两层的向量.
- 在进行 y 方向的 fft 时, 进程 0 将获取同处于 y 方向上进程 1 中最上两层的数据, 即 [:,:,0:1], 与 x 方向不同的是, 我们将利用 mpi 的传输, 将其转置存入 B 中, 这将使得 B 中存有的连续的长为 N 的数据恰好是整个大正方体中 y 方向的向量.
- 在进行 z 方向的 fft 时, 进程 0 将获取同处于 y 方向上进程 1 中 y=0,1 对应的两层数据, 即 [:,0:1,:], 与 x 方向不同的是, 我们将利用 mpi 的传输, 将其转置存入 B 中, 这将使得 B 中存有的连续的长为 N 的数据恰好是整个大正方体中 z 方向的向量.

**三维 fft** 在如上进行完 x 方向上的 fft 后, 我们将数据返回原来的进程中的 A 的对应位置, 然后进行 y 方向, 再进行 z 方向 fft.ifft 的做法与之类似.

$\nabla \times \mathbf{u}$ 先从相应的子进程获得该进程所需的 $v, w$. 具体的

- 从 $myid_v = myorder[2] * size^2 + myorder[0] * size + myorder[1]$ 按 $z > x > y$ 的顺序传出 U, 然后在 myid 进程内以正常的 $x > y > z$ 顺序接收到 U2 中;
- 从 $myid_w = myorder[1] * size^2 + myorder[2] * size + myorder[0]$ 按 $y > z > x$ 的顺序传出 U, 然后在 myid 进程内以正常的 $x > y > z$ 顺序接收到 U3 中;

相应的, 将 U 从 myid 按 $z>x>y$ 发送到 $myorder[1]*size^2+myorder[2]*size+myorder[0]$ 的 U2, 按 $z>x>y$ 发送到 $myorder[2]*size^2+myorder[0]*size+myorder[1]$ 的 U3.

之后需要的求导 $(\partial w/\partial y-\partial v/\partial z)$, 可以直接通过一维 FFT 进行

**$\omega\times\mathbf{u}$** 与上面类似, 获得相应的 $\omega_2,\omega_3,v,w$ 后, 在对应位置相乘即可. 所得结果存储在 U 中

**传输顺序** 详细的, 一个进程一次 fft 需要进行 size 次传输 (本进程内从 A 到 B 对应数据用 MPI 进程到自身的数据传输完成), 为了使交互顺利进行, 我们对这 size 个进程编号:$1,2\cdots,size$, 每个进程 i 用 sendrecv 函数依次向 $i,i+1,i+2,\cdots,i-1$ 发送数据, 同事从 $i,i-1,i-2\cdots,i+1$ 接受数据.

**B 中的 FFT** 对 B 中的第 i 个向量做 FFT 时, 直接对长为 N 的数组做 DCT(偶对称),DST(奇对称). 直接给 fftw 的形参 in,out 均赋值第 i 个向量头所在的地址, 即一个 double* 即可实现该向量的 fft, 同时结果直接覆盖原数据, 而不需要另外的存储抑或是赋值操作.

# 5 数值实验

计算无粘问题 $\nu=0$, 分别取了 $n=,8,16,32,64,128,256;\tau=0.01,0.1$. 在 n=128 和 256 时, 计算到 T=10, 此时变化幅度太小, 所以我们单独观察 $\omega$, 而不将 u 一起绘出, 结果见图 **??**; 对较小的 n 计算到了 T=1000, 见 **??**. 此外, 选取 $\tau=10$, 对 n=16 计算至 T=3700, 见图 **??**(随后范数迅速趋于无穷)

## 5.1 结论

- 在不是特别长的时间内, 比如 n=8 时,T=1000 之内,$u,\omega$ 的无穷范数的变化比较稳定, 表现的像一条直线, 实际上更接近一条二次曲线. 其一阶变化率处于 $10^{-7}$, 二阶变化率处于 $10^{-12}$.
- 在上述情况下 (无穷范数变化稳定, 接近直线),$u_1,\omega_1$ 最大模所在的位置都保持不变

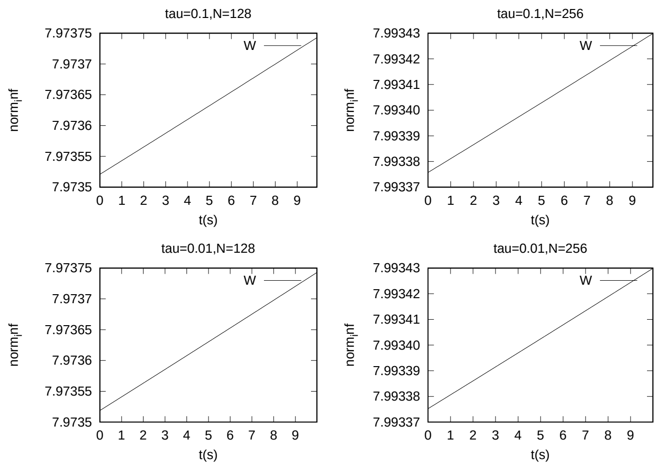
tau=0.1,N=128
tau=0.1,N=256
tau=0.01,N=128
tau=0.01,N=256
W
norm_inf
t(s)

图 1:

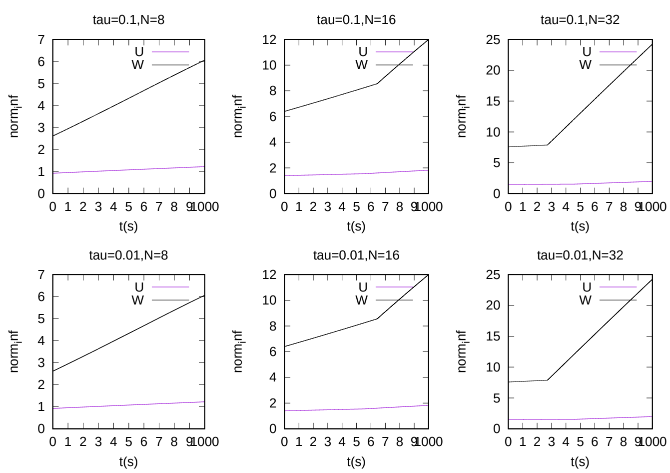
tau=0.1,N=8
tau=0.1,N=16
tau=0.1,N=32
tau=0.01,N=8
tau=0.01,N=16
tau=0.01,N=32
U
W
norm_inf
t(s)

图 2:

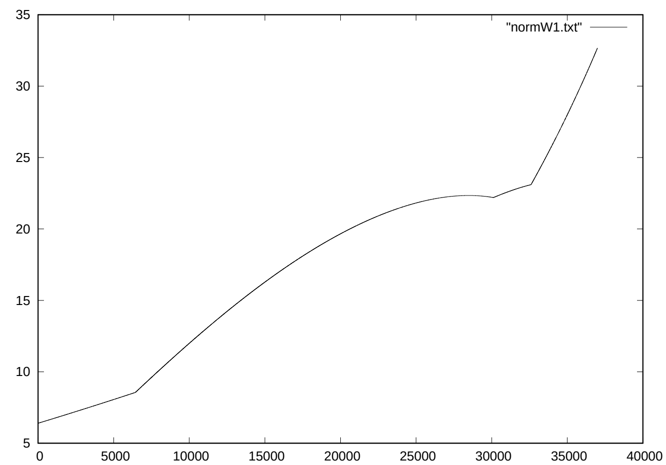
"normW1.txt"
35
30
25
20
15
10
5
0
5000
10000
15000
20000
25000
30000
35000
40000

图 3:

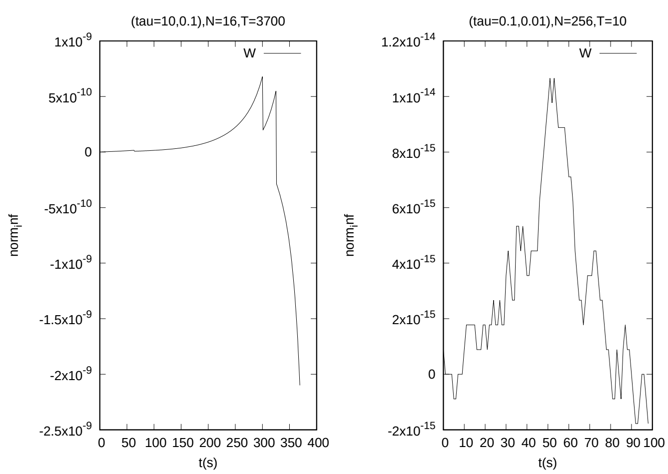
(tau=10,0.1),N=16,T=3700
W
norm_inf
t(s)
(tau=0.1,0.01),N=256,T=10
W
norm_inf
t(s)

图 4:

- 在 $\omega$ 的无穷范数的变化率发生肉眼可见的变化时, 其最大模所在的位置发生一次改变.
- u 的最大模的初始位置永远会在第一次迭代后发生变化, 这可能是因为, 对于 $\omega$, 我们实际记录的是 $t = 1, 2, 3, \cdots, T$ 时刻的状态, 而记录的第 i 个 u, 其实是通过 $\nabla_h \omega^{i-1}$ 计算得到的. 我们接下来忽视 $u_0$ 的情况
- u 的最大模的位置也会发生变化, 但与 $\omega$ 没有直接的联系. 两者不在同一时间发生变化, 甚至发生变化的次数也不相同. 在 n=32,$\tau = 0.1, T = 1000$ 的算例中,$\omega$ 发生一次变化, 而 u 没发生变化
- u 的无穷范数变化远比 $\omega$ 稳定, 在 T=3700,n=16 的例子里,$\omega$ 即将趋于无穷, 而 u 的无穷范数还稳定在 1.5 附近.

**发生变化前最大模所处的位置** 我们存储点的位置在 $(\frac{k}{n} + \frac{1}{2n})\pi$ 上:

- n=256 时,U 的最大模在 $[(\frac{1}{2} - \frac{1}{n})\pi, (\frac{38}{128} + \frac{1}{n})\pi, \frac{1}{n}\pi]$;$\omega$ 的最大模在 $[\frac{1}{n}\pi, (\frac{1}{2} - \frac{1}{n})\pi, (\frac{1}{2} + \frac{1}{n})\pi]$.
- n=128 时,U 的最大模在 $[(\frac{1}{2} - \frac{1}{n})\pi, \frac{1}{n}\pi, (\frac{19}{64} + \frac{1}{n})\pi]$;$\omega$ 的最大模在 $[\frac{1}{n}\pi, (\frac{1}{2} - \frac{1}{n})\pi, (\frac{1}{2} + \frac{1}{n})\pi]$.
- n=64 时,U 的最大模在 $[(\frac{1}{2} - \frac{1}{n})\pi, \frac{1}{n}\pi, (\frac{9}{32} + \frac{1}{n})\pi]$;$\omega$ 的最大模在 $[\frac{1}{n}\pi, (\frac{1}{2} - \frac{1}{n})\pi, (\frac{1}{2} + \frac{1}{n})\pi]$.
- n=32 时,u 的最大模在 $[(\frac{1}{2} - \frac{1}{n})\pi, \frac{1}{n}\pi, (\frac{4}{16} + \frac{1}{n})\pi]$;$\omega$ 的最大模在 $[\frac{1}{n}\pi, (\frac{1}{2} - \frac{1}{2n})\pi, (\frac{1}{2} + \frac{1}{n})\pi]$.

**n 的影响** 对同一个 $\tau$, 随着 n 的增大,$\omega$ 的范数曲线发生转折 (对应最大模位置的移动) 的时间点在前移.

**$\tau$ 的影响** 见图 **??** 对同一个 n, 不同的 $\tau$ 算至同一时间 T, 无穷范数的曲线基本重合, 说明算法对于 ODE 问题

$$\frac{d\omega}{dt} = f(\omega)$$

的计算较精确. 可以观察随着 $\tau$ 的增大, 两组 $\tau$ 对应时间点上无穷范数的差随着 $\tau$ 的差距变大, 说明 RungeKutta 方法对于该 ode 至少是有阶的. 由于不是主要讨论的问题, 不做详细叙述.