方程求解

2-D Euler Equation

方程的空间离散化 方程的时间离散化 人工耗散项 时间步长的选取 边界条件设置 example code

方程求解

2-D Euler Equation

将二维欧拉方程求解方程写成如下形式:

$$rac{\partial}{\partial t}\int_{\Omega}m{W}d\Omega+\int_{\partial\Omega}\left(m{F}dy-m{G}dx
ight)=m{0}$$

其中各参数意义如下:

$$m{W} = egin{bmatrix}
ho \
ho U \
ho V \
ho E \end{bmatrix} \quad m{F} = egin{bmatrix}
ho U \
ho U^2 + P \
ho U V \
ho U H \end{bmatrix} \quad m{G} = egin{bmatrix}
ho V \
ho U V \
ho V^2 + P \
ho V H \end{bmatrix}$$

其中有:

$$\left\{egin{aligned} E = rac{P}{\gamma - 1} + rac{U^2 + V^2}{2} \ H =
ho E + P \end{aligned}
ight.$$

方程的空间离散化

$$rac{\partial}{\partial t}\int_{\Omega_k} oldsymbol{W} d\Omega pprox \Omega_k rac{\partial oldsymbol{W}}{\partial t}$$

所以可以将前式写作:

$$rac{\partial oldsymbol{W}}{\partial t} = -rac{1}{\Omega_k} \int_{\partial \Omega_k} \left(oldsymbol{F} dy - oldsymbol{G} dx
ight)$$

现在假设 Ω_k 是由一个多边形构成的小单元,记一共有 N_{edge}^k 个边,在其第j号边上,可以计算其边界通量。

假设该边界是由 (x_j^1, y_j^1) 指向 (x_j^2, y_j^2) 的一条边,记作 $j(1 \le j \le N_{\text{edge}}^k)$,假设该顺序是绕着凸多边形 Ω_k 的逆时针方向顺序,则其外法线方向为:

$$ec{l}_j = rac{1}{\sqrt{\Delta x_j^2 + \Delta y_j^2}} (-\Delta y_j, \Delta x_j)$$

其中有:

$$egin{aligned} \Delta x_j &= x_j^2 - x_j^1 \ \Delta y_j &= y_j^2 - y_j^1 \end{aligned}$$

则对于该边上的通量,可以近似认为F与G在小单元边界上保持不变,因此表示如下:

$$oldsymbol{Q}_{k}^{j} = oldsymbol{F} \Delta y_{i} - oldsymbol{G} \Delta x_{i}$$

我们考虑这个值在单元k和单元p上的计算,上述的F和G可以近似用j边两侧单元平均值表示。

为了书写方便,令:

$$Z_j = U_j \Delta y_j - V_j \Delta x_j$$

则可以将这第k个单元的流动方程积分表达式写作:

$$rac{doldsymbol{W}_k}{dt} = -rac{1}{\Omega}\sum_{k}^{N_{ ext{edge}}^k}egin{bmatrix} Z
hoU + P\Delta y \ Z
hoV - P\Delta x \ Z
hoH \end{bmatrix}_j$$

因此使用格心格式时,FVM迭代格式已经构造完成,方程计算如下:

$$rac{doldsymbol{W}_k}{dt} = -rac{1}{\Omega} \sum_{k}^{N_{ ext{edge}}^k} oldsymbol{Q}_k^j = -rac{oldsymbol{Q_k}}{\Omega_k} = oldsymbol{R}_k$$

方程的时间离散化

理论上, 我们在考虑时间步长 Δt 时, 会有:

$$egin{aligned} oldsymbol{W}_k^{t+\Delta t} - oldsymbol{W}_k^t \ rac{\Delta t} \end{aligned} = oldsymbol{R}_k$$

但这样做计算精度不高,我们可以考虑四阶龙格库塔时间离散方法:

$$egin{cases} W^{(0)} = W^t \ W^{(m)} = W^{(0)} + lpha_m \Delta t m{R}^{(m-1)} \quad m = 1 \sim 4 \ W^{t+\Delta t} = W^{(4)} \end{cases}$$

其中:

$$egin{cases} lpha_1=rac{1}{4}\ lpha_2=rac{1}{3}\ lpha_3=rac{1}{2}\ lpha_4=1 \end{cases}$$

人工耗散项

前述式子在计算中不考虑耗散,因此离散误差、舍入误差等会随时间累计。 为了消除这个错误,人工耗散项由此加入,如下式所示:

$$oldsymbol{R}_k = -rac{oldsymbol{Q}_k - oldsymbol{D}_k}{\Omega_k}$$

由Jameson提出的二姐和四阶混合人工耗散项由下式所示:

$$oldsymbol{D}_k = \sum_{j=1}^{N_{ ext{edge}}^k} oldsymbol{d}_j^{(2)} + \sum_{j=1}^{N_{ ext{edge}}^k} oldsymbol{d}_j^{(4)}$$

其中:

$$egin{aligned} oldsymbol{d}_j^{(2)} &= lpha_j arepsilon_j^{(2)} (oldsymbol{W}_p - oldsymbol{W}_k)_j \ oldsymbol{d}_j^{(4)} &= -lpha_j arepsilon_j^{(4)} ig(
abla^2 oldsymbol{W}_p -
abla^2 oldsymbol{W}_kig)_j \end{aligned}$$

其中给定:

$$egin{align}
abla^2 oldsymbol{W}_k &= \sum_{j=1}^{N_{ ext{edge}}^k} (oldsymbol{W}_j - oldsymbol{W}_k) \ &arepsilon_j^{(2)} = k^{(2)}
u_j \ &arepsilon_j^{(4)} = \max(0, k^{(4)} - arepsilon_j^{(2)})
onumber \end{split}$$

(Scaling factor of shock sensor)

$$u_j = rac{|P_p - P_k|}{|P_p + P_k|}$$
 $lpha_j = \left(|U\Delta y - V\Delta x| + c\sqrt{\Delta x^2 + \Delta y^2}
ight)_j$

其中 c_j 表示当地声速,计算方式为: $c_j = \sqrt{\gamma \frac{P_j}{
ho_j}}$ 。

另外在论文中, $k^{(2)} \in [0.5, 1.0]$, $k^{(4)} \in [1/256, 1/32]$,这是由经验取值。

时间步长的选取

对于k号单元,采用如下的选取格式:

$$\Delta t_k = CFLrac{\Omega_k}{lpha_k}$$

于是在该时间层上的 Δt 取为 $\min \Delta t_k$ 。

边界条件设置

因为在通量计算中,压力项P的存在,这导致在物面边界上存在通量。

对于远场边界条件,则存在入流与出流的不同,超音速出流的 edge 取值不依赖于下游的量,而亚音速则仍可以用格心格式进行平均。

example code

```
function laplace(grid, W_cur)
% 计算ΔW
end
function [Q, D2, D4, t] = each_step(case, grid, W_cur)
Q = zeros(grid.ncells, 4);
D2 = zeros(grid.ncells, 4);
D4 = zeros(grid.ncells, 4);
t = zeros(grid.ncells);
W_laplace = laplace(grid, W_cur)
% 需要计算粘性项D2、D4以及时间t
t = case.CFL * grid.vol / t;
end
function Q = each_step_non_viscosity(case, grid, W_cur)
% 不需要计算粘性项D2、D4及时间t
end
coeff = [1/4, 1/3, 1/2, 1];
function W_next = rk4(case, grid, W_cur)
Q, D2, D4, t = each_step(case, grid, W_cur);
dt = min(t);
W_next = W_cur - (Q - D2 - D4) / grid.vol * coeff(1) * dt;
for j = 2: 4
    Q = each_step(case, grid, W_next);
    W_next = W_cur - (Q - D2 - D4) / grid.vol * coeff(j) * dt;
end
end
```