计算传热学的近代进展

第七章　压力与速度耦合算法研究进展

主讲　陶文铨

西安交通大学能源与动力工程学院
热流科学与工程教育部重点实验室

2019年5月30日，西安

第7 章 压力与速度耦合算法研究进展

7.1 SIMPLE系列算法的基本思想

7.1.1 二维交叉网格上流场控制方程的离散及其求解

7.1.2 压力修正算法的基本思想

7.1.3 SIMPLE算法的实施步骤和两个重要假设

7.1 SIMPLE算法的基本思想

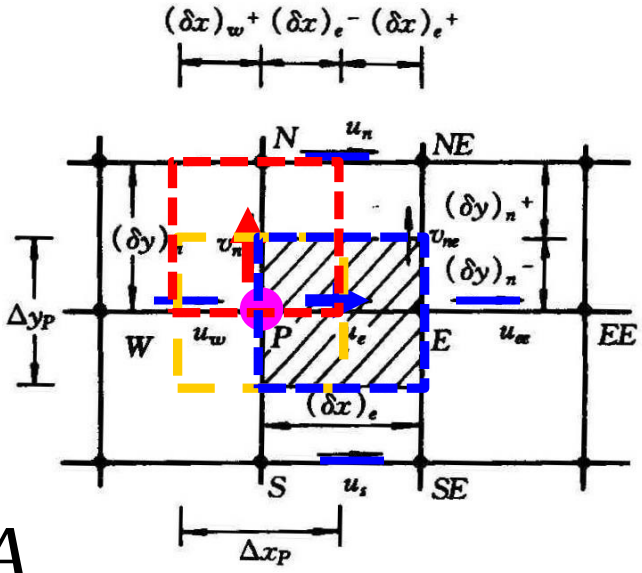
7.1.1 二维交叉网格上流场控制方程的离散及其求解

1. 方程及离散

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}=0$$

$$\frac{\partial(uu)}{\partial x}+\frac{\partial(vu)}{\partial y}=-\frac{1}{\rho}\frac{\partial p}{\partial x}+\nu\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)$$

$$\frac{\partial(uv)}{\partial x}+\frac{\partial(vv)}{\partial y}=-\frac{1}{\rho}\frac{\partial p}{\partial y}+\nu\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}\right)$$



$$a_e u_e=\sum a_{nb}u_{nb}+b+(p_P-p_E)A_e$$

$$a_n v_n=\sum a_{nb}v_{nb}+b+(p_P-p_N)A_n$$

$$[(\rho u)_e-(\rho u)_w]\Delta y+[(\rho v)_n-(\rho v)_s]\Delta x=0$$

2. 代数方程的求解

(1) 联立直接求解：不存在任何压力与速度的耦合关系处理问题；但计算机内存容量的限制以及问题的非线性限制了此法的应用；

(2) 分离式求解 (segregated method)：假设压力场 p^*，先解 u 方程，再解 v 方程，然后通过利用连续性方程来修正假定的压力，使与修正后压力对应的速度满足连续性要求。一般地，压力与速度的改进都通过加上一个小量来进行：p', u', v'。

7.1.2 压力修正算法的基本思想

1. 在流场迭代求解的每一个层次上，都必须满足连续性方程；

2. 在每一层次动量方程的求解中获得中间速度后对压力进行修正，使与修正后的压力，p^*+p'，相对应的速度，u^*+u'，v^*+v'，满足质量守恒要求。

7.1.3 SIMPLE算法的实施步骤和两个重要简化假设

1. 实施步骤

1） 假定一个速度场 $u^{(0)}$，$v^{(0)}$,计算a_e, a_n, a_{nb}, b等;

2） 假定一个压力场 p^*;（简化假设1：p^*独立设定）

3） 求解动量方程，得出临时速度场 u^*，v^*;

4） 据 u^*，v^*修正压力，获得修正分量p', 要求与p'对应的u', v' 使 (u^*+u'), (v^*+v') 满足质量守恒，由此导出压力修正方程;

$$a_e(u_e^* + u_e') = \sum a_{nb}(u_{nb}^* + u_{nb}') + b + A_e[(p_P^* + p_P') - (p_E^* + p_E')]$$

$$a_e u_e^* = \sum a_{nb} u_{nb}^* + b + A_e(p_P^* - p_E^*)$$

$$a_e u_e' = \sum a_{nb} u_{nb}' + A_e(p_P' - p_E')$$

略去邻点速度修正量的影响（第二个简化假设）可得：

$$u_e' = \frac{A_e}{a_e}(p_P' - p_E') = d_e(p_P' - p_E'), \quad d_e = \frac{A_e}{a_e}$$

$$v_n' = \frac{A_n}{a_n}(p_P' - p_N') = d_n(p_P' - p_N'), \quad d_n = \frac{A_n}{a_n}$$

5） 获得 p'后修正速度和压力；

$$u_e = u_e^* + d_e(p_P' - p_E') \quad v_n = v_n^* + d_n(p_P' - p_N') \quad p = p^* + \alpha_p p'$$

6）以u_e, v_n 以及 $p^*+ \alpha_p p'$开始下一层次的迭代计算。

2. 两个基本简化假设

假设 1：p^*独立设定；

假设 2：略去邻点速度修正量的影响。

7.2 SIMPLER、SIMPLEC、SIMPLEX与SIMPLE的比较

7.2.1 SIMPLER完全克服了第一个简化假设（1980）

7.2.2 SIMPLEC减轻了第二个简化假设的影响（1984）

7.2.3 SIMPLEX-SIMPLEC思想的拓展 （1986）

7.2.4 四种算法在密网格下收敛性的比较(2003)

7.2.5 压力修正算法是一种预估－校正算法

7.2 SIMPLER、SIMPLEC、SIMPLEX与SIMPLE 的比较

7.2.1 SIMPLER-Patankar 完全克服了第一个简化假设（1980）

1. 假定一个速度场 $u^{(0)}$，$v^{(0)}$,计算a_e, a_n, a_{nb}, b等;

2. 由假定的速度$u^{(0)}$，$v^{(0)}$，计算假拟速度 $u, \tilde{v}$，进而根据质量守恒导出一个压力方程，以确定 p^*;（完全克服了简化假设1）；

$$u_e = \sum \frac{a_{nb} u_{nb}^0 + b}{a_e} \qquad \tilde{v}_n = \sum \frac{a_{nb} v_{nb}^0 + b}{a_e}$$

$$a_P p_P = a_E p_E + a_W p_W + a_N p_N + a_S p_S + b$$

$$b = \frac{(\rho_P^0 - \rho_P)\Delta x \Delta y}{\Delta t} + [(\rho u)_w - (\rho u)_s]A_e + [(\rho \tilde{v})_s - (\rho \tilde{v})_n]A_n$$

3. 求解动量方程，得出临时速度场 u^*，v^*;

4. 据 u^*，v^*修正压力，获得修正分量p', 要求与p' 对应的u',v' 使 （u^*+u'),(v^*+v') 满足质量守恒，由此导出压力修正方程（同SIMPLE）；

5. 获得 p'后修正速度，但不修正压力;

$$u_e = u_e^* + d_e(p_P' - p_E') \qquad v_n = v_n^* + d_n(p_P' - p_N')$$

6. 以u_e, v_n 开始下一层次的迭代计算。

7.2.2 SIMPLEC-van Doormaal/Raithby 减轻了第二个简化假设的影响（1984）

1. 假定一个速度场;
2. 假定一个压力场;
3. 求解动量方程;

完全同SIMPLE

4. 与SIMPLE有区别:

在速度修正量方程两边各减去 $\sum a_{nb}u_e'$ 得:

$$a_e u_e' - \sum a_{nb}u_e' = \sum a_{nb}u_{nb}' - \sum a_{nb}u_e' + b + A_e(p_P' - p_E')$$

$$u_e'(a_e - \sum a_{nb}) = \sum a_{nb}(u_{nb}' - u_e') + b + A_e(p_P' - p_E')$$

可以认为：u_e' ,u_{nb}' 具有相同的量级，因而略去 $\sum a_{nb}(u_{nb}'-u_e')$ 而造成的影响比SIMPLE中略去 $\sum a_{nb}u_{nb}'$ 而造成的影响要小得多。于是有：

$$u_e' = \left(\frac{A_e}{a_e - \sum a_{nb}}\right)(p_P' - p_E') \qquad v_n' = \left(\frac{A_n}{a_n - \sum a_{nb}}\right)(p_P' - p_N')$$

SIMPLEC中的de

$$d_e = \frac{A_e}{a_e - \sum a_{nb}};\ d_n = \frac{A_n}{a_n - \sum a_{nb}}$$

此即SIMPLEC中速度修正值计算式。

5. 与SIMPLE有区别： p' 不再亚松弛。

$$u_e' = d_e(p_P' - p_E') = d_e \Delta p_e' \qquad u = u^* + u'$$

$$v_n' = d_n(p_P' - p_N') = d_n \Delta p_n' \qquad v = v^* + v'$$

$$p = p^* + p'$$

6. 以 u_e, v_n 以及 $p^*+ p'$ 开始下一层次的迭代计算。

7.2.3 SIMPLEX—Raithby SIMPLEC思想的拓展（1986）

SIMPLEC算法的实质性的一步是d的计算式的改进，用以部分考虑邻点的影响;

由此推广开去：如果能够形成一组关于计算 d 的方程，使其能部分考虑邻点的影响，则也可加速收敛。

由 $u'_e = d_e(p'_P - p'_E) = d_e \Delta p'_e$ 推广到 $\boxed{u'_{nb} = d_{nb} \Delta p'_{nb}}$

代入到： $a_e u'_e = \sum a_{nb} u'_{nb} + A_e(p'_P - p'_E)$

得 $a_e d_e \Delta p'_e = \sum a_{nb} d_{nb} \Delta p'_{nb} + A_e \Delta p'_e$

假定： $\boxed{\Delta p'_e = \Delta p'_{nb}}$

于是： $a_e d_e \Delta p'_e = \sum a_{nb} d_{nb} \Delta p'_{nb} + A_e \Delta p'_e$ （新的假定）

得确定系数d 的代数方程组： $\boxed{a_e d_e = \sum a_{nb} d_{nb} + A_e}$

根据已知的动量方程系数可以由此求出d。边界条件是：

“绝热型” ---d方程 的边界系数为0

得出上式时没有略去邻点的影响；但是引入了新的假定：$\Delta p_e' = \Delta p_{nb}'$ 因此也仅能部分克服第二个假设。

计算步骤如下：

(1) 假定速度初场u^0, v^0, 计算系数a, b

(2) 假定压力场 p^*；

(3) 求解动量离散方程，得u^*, v^*;

(4) 求解 d方程，然后求解压力修正方程，得p';

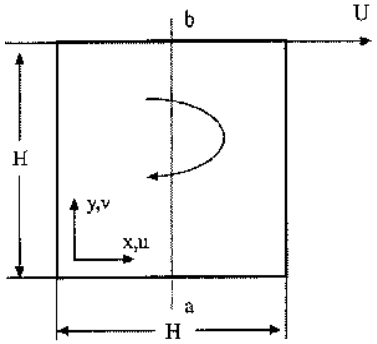
(5) 由 p' 修正速度，得u', v';

(6) 以（u^*+u'),(v^*+v') ,(p^*+p')作为本层次的速度场与压力，开始下一层次的迭代（p'不作亚松弛）。

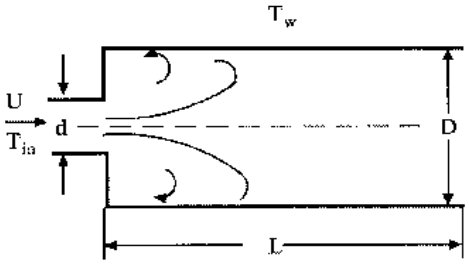
7.2.4 四种算法在密网格下收敛性的比较

算法	第一个简化假定	第二个简化假定
SIMPLE	p^*独立地假定，通过修正量改进并亚松弛	$\sum a_{nb}u'_{nb}=0, d_e=A_e/a_e$
SIMPLER	p^*由已知的速度算出，克服第一个假定，压力修正量不修正压力	$\sum a_{nb}u'_{nb}=0, d_e=A_e/a_e$
SIMPLEC	p^*独立地假定，通过修正量改进，但压力的修正量不亚松弛	$\sum a_{nb}(u'_{nb}-u'_e)=0,$ $d_e=A_e/(a_e-\sum a_{nb})$ 减轻第二个假定影响
SIMPLEX	p^*独立地假定，通过修正量改进，但压力的修正量不亚松弛	$\Delta p'_e=\Delta p'_{nb}$ 求解 de 减轻第二个假定影响

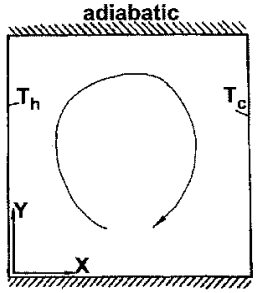
密网格下对四个算例比较结果



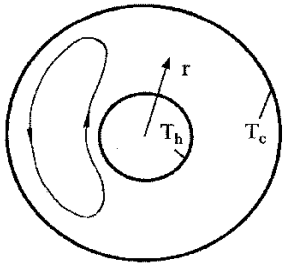
(1)lid-driven cavity flow



(2)flow in a tube with sudden expansion



(3)natural convection in a square cavity



(4)natural convection in a horizontal annular

1.CPU时间： SIMPLER/SIMPLEX $\geq$ SIMPLE/SIMPLEC

2.健壮性： SIMPLE<SIMPLER $\leq$ SIMPLEC/SIMPLEX

3.*d* 值： SIMPLEC/SIMPLEX大于 SIMPLE/SIMPLER

密网格下四种算法中SIMPLEC综合性能最优。

Zeng M, Tao W Q. A comparison study of the convergence characteristics and robustness for four variants of SIMPLE family at fine grids. Engineering Computations, 2003, 20(3/4):320-341

推荐阅读(8)

About *d* $u' = d(\Delta p')$ $d \uparrow$ $\Delta p' \downarrow$

Natural convection in a square cavity

	42×42		42×42		82×82		82×82	
	SIMPLE	SIMPLER	SIMPLEC	SIMPLEX	SIMPLE	SIMPLER	SIMPLEC	SIMPLEX
$d_u(10,6)$	0.5927	0.5927	2.964	2.928	0.2981	0.2981	1.490	1.474
$d_u(20,20)$	0.5960	0.5960	2.980	2.979	0.2975	0.2975	1.488	1.488

Natural convection in a square cavity

	42×42				82×82			
	SIMPLE	SIMPLER	SIMPLEC	SIMPLEX	SIMPLE	SIMPLER	SIMPLEC	SIMPLEX
$d_u(12,7)$	1.929	1.930	9.643	9.525	0.9999	0.9999	4.999	4.976
$d_u(22,22)$	1.874	1.873	9.368	9.265	0.9612	0.9612	4.803	4.798

Thus in SIMPLEC，SIMPLEX no underrelaxation is needed for *p* ’.

7.2.5 压力修正算法是一种预估－校正算法

压力修正算法的计算步骤可以分为两大步：

1. 预估步(prediction step)：1－3，由设定的u^0, v^0，到解出u^*, v^*;---得到速度点的预估值
2. 校正步(correction step)：4－6，解出p'，得出满足质量守恒的u，v。----得到速度的校正值。

对以上的四种算法，预估与校正均各实施一次，就转到下一层次计算。

1986年提出的PISO算法则进行两次校正，在一定程度上该进了算法的收敛性。

7.3 PISO算法

7.3.1 PISO算法的基本思想

7.3.2 PISO算法的实施步骤

1.预估步

2.第一校正步

3.第二校正步

7.3.3 校正步数目的影响

7.3 PISO算法

7.3.1 PISO算法的基本思想

一步预估，两步校正；在第一次校正中获得一个修正后压力及相对应的速度，该速度满足质量守恒并显式地满足动量守恒；如此连续几次校正可望能更好地同时满足质量与动量守恒的速度场。

7.3.2 PISO算法的实施步骤

1.预估步 完全同SIMPLE；

采用PISO算法中的符号，将预估步的算式表示为：

$$a_e u_e^* = \sum a_{nb} u_{nb}^* + b + A_e (p_P^{(k)} - p_E^{(k)})$$
$$a_e v_n^* = \sum a_{nb} v_{nb}^* + b + A_n (p_P^{(k)} - p_N^{(k)}) \quad (1)$$

这样得出的速度满足线性动量守恒方程，但未必满足质量守恒。所谓线性动量守恒方程是指动量方程中的系数固定为常数，不受速度变化的影响这一特性。

2.第一校正步　基本同SIMPLE

设改进后的压力为 $p^* = p^{(k)} + p'$，相应的速度为u^{**}, v^{**}, 要求u^{**},v^{**}显式地满足线性动量守恒方程：

$$a_e u_e^{**} = \sum a_{nb} u_{nb}^* + b + A_e (p_P^* - p_E^*)$$
$$a_e v_n^{**} = \sum a_{nb} v_{nb}^* + b + A_n (p_P^* - p_N^*) \quad (2)$$

显式地满足线性动量守恒方程

将式（2）减式（1），邻点项可自动消去：

$$a_e u_e^{**} = \sum a_{nb} u_{nb}^{*} + b + A_e (p_P^{*} - p_E^{*})$$

$$a_e u_e^{*} = \sum a_{nb} u_{nb}^{*} + b + A_e (p_P^{(k)} - p_E^{(k)})$$

$$a_e (u^{**} - u^{*}) = A_e [(p_P^{*} - p_P^{(k)}) - (p_E^{*} - p_E^{(k)})]$$

$$u' \qquad p_P' \qquad p_E'$$

$$a_e u_e' = A_e (p_P' - p_E') \qquad u_e' = \frac{A_e}{a_e}(p_P' - p_E')$$

$$a_n v_n' = A_n (p_P' - p_N') \qquad v_n' = \frac{A_n}{a_n}(p_P' - p_N')$$

将(u*+u’),(v*+v’) 代入质量守恒方程：

$$[(\rho u)_e-(\rho u)_w]\Delta y+[(\rho v)_n-(\rho v)_s]\Delta x=0$$ （稳态问题）

可得

$$a_P p_P^{'}=a_E p_E^{'}+a_W p_W^{'}+a_N p_N^{'}+a_S p_S^{'}+b$$

$$a_P=a_E+a_W+a_N+a_S$$

$$a_E=d_e A_e\rho_e \quad a_W=d_w A_w\rho_w \quad a_n=d_n A_n\rho_n \quad a_S=d_s A_s\rho_s$$

$$b=[(\rho u^*)_w-(\rho u^*)_e]A_e+[(\rho v^*)_s-(\rho v^*)_n]A_n$$

求解压力修正值方程后，由第一校正步得出：

$$u_e^{**}=u_e^{*}+d_e(p_P^{'}-p_E^{'}) \quad v_n^{**}=v_n^{*}+d_n(p_P^{'}-p_N^{'})$$

这一速度场满足质量守恒，显式地满足动量守恒：

$$a_e u_e^{**} = \sum a_{nb} u_{nb}^{*} + b + A_e (p_P^{*} - p_E^{*}) \qquad (2)$$

3.第二校正步　PISO特有

设这一步改进后压力为p^{**},速度场为u^{***},v^{***},同样要求速度场满足质量守恒，显式地满足动量守恒。

$$a_e u_e^{***} = \sum a_{nb} u_{nb}^{**} + b + A_e (p_P^{**} - p_E^{**}) \qquad (3)$$

$$a_e v_n^{***} = \sum a_{nb} v_{nb}^{**} + b + A_n (p_P^{**} - p_N^{**})$$

将式（3）减式（2），邻点项均属已知：

$$a_e(u_e^{***}-u_e^{**})=\sum a_{nb}(u_{nb}^{**}-u_{nb}^{*})+A_e[(p_P^{**}-p_P^{*})-(p_E^{**}-p_E^{*})]$$

u'' 已知项 p_P'' p_E''

$$u_e''=\sum[a_{nb}(u_{nb}^{**}-u_{nb}^{*})+A_e(p_P''-p_E'')]/a_e$$

类似地 $$v_n''=\sum[a_{nb}(v_{nb}^{**}-v_{nb}^{*})+A_n(p_P''-p_N'')]/a_n$$

要求速度场 u*,v*** 满足质量守恒，写成一般形式：**

$$c_e u_e^{***}-c_w u_w^{***}+c_n v_n^{***}-c_s v_s^{***}=0$$

将u*=u**+u”等代入，可得p”的方程：**

$$a_P p_P''=\sum a_{nb}p_{nb}''+b$$

系数计算式同前，但源项 b 为：

$$b = T_w - T_e + T_s - T_n$$

其中 $T_e = \frac{c_e}{a_e}[\sum a_{nb}(u^{**}_{nb} - u^{*}_{nb})_e]$ $T_n = \frac{c_n}{a_n}[\sum a_{nb}(v^{**}_{nb} - v^{*}_{nb})_n]$

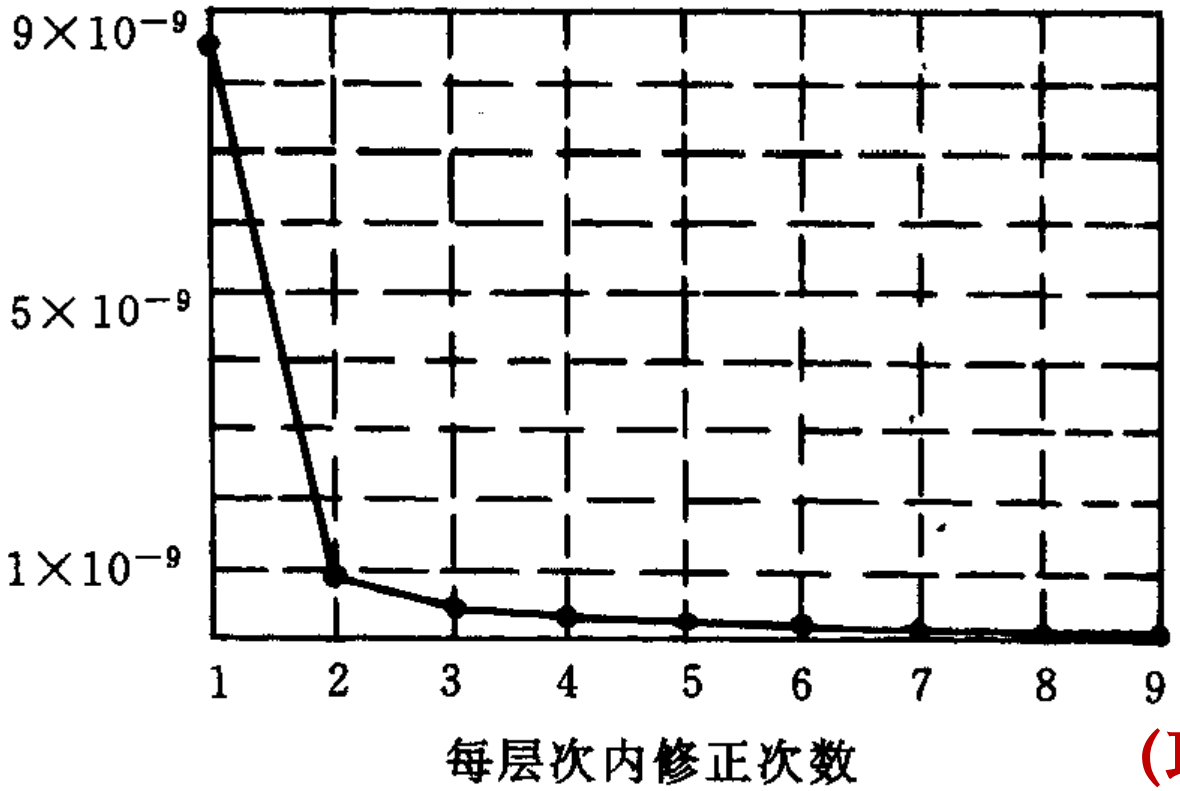
获得 p'' 后第二次改进的速度为：

$$u^{***}_e = u^{**}_e + \frac{1}{a_e}[\sum a_{nb}(u^{**}_e - u^{*}_e) + A_e(p''_P - p''_E)]$$

$$v^{***}_e = v^{**}_e + \frac{1}{a_n}[\sum a_{nb}(v^{**}_n - v^{*}_n) + A_n(p''_P - p''_N)]$$

7.3.3 校正步数目的影响

一般两个校正步即可，第三步的影响就不明显。



（取自党政的大作业）

PISO算法的多步计算只在校正步中实施。我们后来的实践经验表明同时在预估与校正步中实施多步修正效果更加明显。

7.4 CLEAR Algorithm

7.4.1 Efforts to overcome the 2nd assumption

7.4.2 Basic idea of the fully implicit algorithm—CLEAR

7.4.3 Basic numerical steps of one iteration in CLEAR

7.4.4 Improvement of the robustness of CLEAR

7.4.5 Computational procedures of one iteration in CLEAR

7.4.6 Discussion on CLEAR algorithm

7.4.7 Numerical examples of CLEAR

7.4 CLEAR Algorithm

7.4.1 Efforts to overcome the 2nd assumption

1) SIMPLEC by van Doormaal/Raithby (1984)

2) SIMPLEX by van Doormaal/Raithby (1986)

3) SIMPLE-with Date modification (1986)

4) Explicit correction step method by Yen and Liu(1993)

5) SIMPLE-with Sheng et al. modification (1998)

6) MSIMPLER by Yu et al. (2001)

All these variants can only partially overcome the effect of 2nd assumption. **Why?**

Analysis: When the update of velocity is conducted along the line by adding a small correction to the previous solution, this assumption can never be overcome for the practical management of the solution of the pressure correction equations.

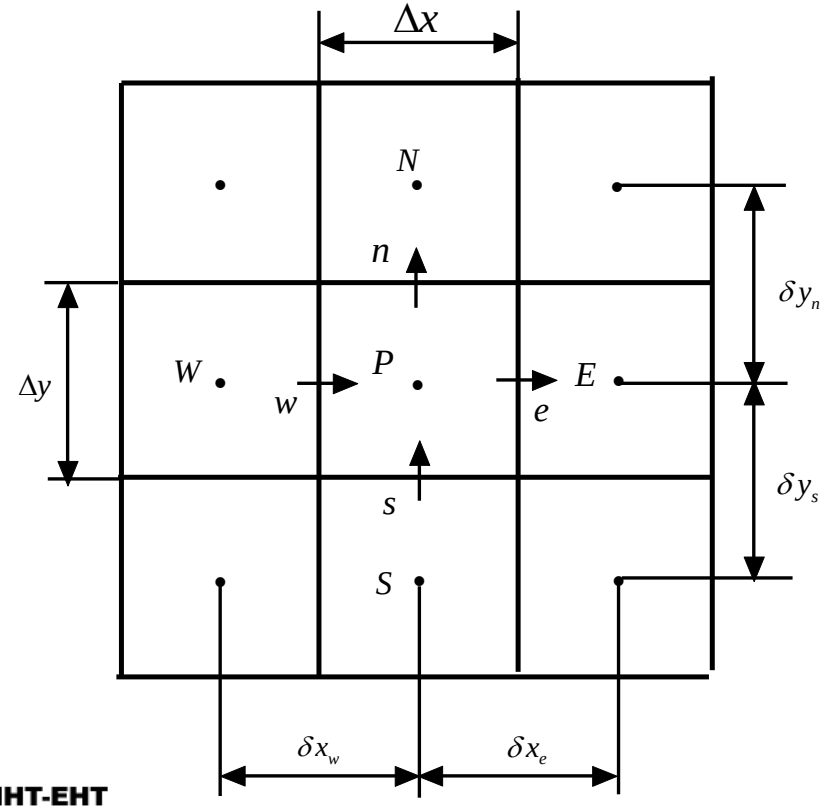
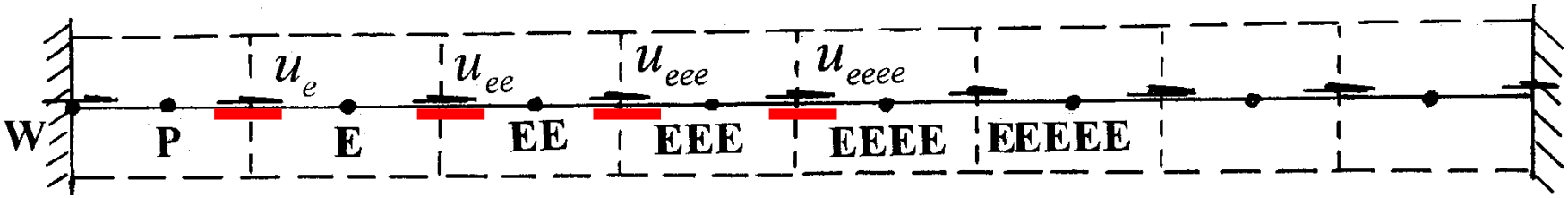


Fig. 2-D staggered grid



$$a_e u'_e = \sum a_{nb} u'_{nb} + A_e (p'_P - p'_E)$$

$$(a_e u'_e)_{ee} = (\sum a_{nb} u'_{nb} + A_e (p'_P - p'_E))_{ee}$$

$$(a_e u'_e)_{eee} = (\sum a_{nb} u'_{nb} + A_e (p'_P - p'_E))_{eee}$$

$$(a_e u'_e)_{eeee} = (\sum a_{nb} u'_{nb})_{eeee} + (A_e (p'_P - p'_E))_{eeee}$$

Finally the pressure corrections at all points in the x-direction will be involved.

In years of 2004 and 2008, our group developed CLEAR and IDEAL, respectively. Both versions completely delete the 2nd assumption, making the algorithm fully-implicit.

7.4.2 Basic idea of the fully implicit algorithm—CLEAR

In the CLEAR algorithm, the improved pressure and velocity are solved directly, rather than by adding a correction term to the intermediate solution —a key point of CLEAR (Z G Qu,屈治国).

Tao WQ, Qu ZG, He YL , A novel segregated algorithm for incompressible fluid flow and heat transfer problems - Clear (coupled and linked equations algorithm revised) part I: Mathematical formulation and solution procedure ,Numerical Heat Transfer, Part B, 2004, 45 (1): 1-17

7.4.3 Basic numerical steps of one iteration in CLEAR

The basic numerical steps in CLEAR algorithm:

Step 1: Assuming an initial velocity field u^0, v^0 ; and calculating the coefficient of the discretized momentum equation and pseudo-velocity:

$$\tilde{v}_n^0 = \frac{\sum a_{nb} v_{nb}^0 + b}{a_n} \qquad u_e^0 = \frac{\sum a_{nb} u_{nb}^0 + b}{a_e}$$

Step 2: Solving the pressure equation and obtaining the temporary pressure field p^*;

Step 3: Solving the momentum equation to get u^*, v^*;

（以上同SIMPLR）

Step 4: Recalculating the coefficient of momentum equation and the pseudo-velocity u^*, $\tilde{v}^*$ based on the intermediate velocity solution u^*, v^*, and the improved velocity is expressed by

$$u_e = u_e^* + d_e(p_P - p_E) \qquad v_n = \tilde{v}_n^* + d_n(p_P - p_N)$$

where u_e^* and $\tilde{v}_n^*$ are actually representing the effects of the neighboring grid points,

and re-solving pressure equation for the improved pressure field based on u_e^*, $\tilde{v}_n^*$:

$$a_P p_P = \sum a_{nb} p_{nb} + b$$

$$a_E = (\rho A d)_e \qquad a_N = (\rho A d)_n \qquad a_P = a_E + a_W + a_N + a_S$$

$$b = (\rho u^* A)_w - (\rho u^* A)_e + (\rho \tilde{v}^* A)_s - (\rho \tilde{v}^* A)_n$$

Step 5: Improving the velocity to obtain the solution of the present iteration.

$$u_e = u^* + d_e(p_P - p_E) \qquad v_n = \tilde{v}^* + d_n(p_P - p_N)$$

Step 6: Returning to step 1 and repeating until convergence is reached.

The major assumption made in the SIMPLE-family is totally discarded and a significant acceleration of the convergence speed is obtained.

This algorithm is called CLEAR-standing for Coupled & Linked Equations Algorithm Revised. The pressure equation are solved twice in CLEAR.

Numerically CLEAR can still be regarded as the Prediction—Correction method. But correction is

conducted by directly solve a new pressure rather than by adding a small correction value .

Numerical practices have found that the above computational procedure is sometimes less robust than SIMPLER, probably because its too big change between the solutions of two successive iterations.

7.4.4 Improvement of the robustness of CLEAR

In order to increase its robustness, apart from the underrelaxaton of velocity which is incorporated into the solution process as SIMPLE-like algorithm, a second relaxation is introduced in the determination of the pseudo-velocity in the correction step.

1. **New equation for the pseudo-velocity in the correction step.**

The discretized momentum equation with incorperated underrelaxation procedure is：

$$\frac{a_e}{\alpha}u_e = \sum a_{nb}u_{nb} + b + (p_P - p_E)A_e + \frac{1-\alpha}{\alpha}a_e u_e^0$$

Replacing *u* and u^0 at the right hand of the above equation by *u and rewrite into following form:**

$$u_e = \frac{\sum a_{nb}u_{nb}^* + b + \frac{1-\alpha}{\alpha}a_e u_e^*}{a_e/\alpha} + d_e(p_P - p_E)$$

Taking it as the second pseudo-velocity based on *u,*v**,and replacing Alfa by Beta**

$$u_e^* = \frac{\sum a_{nb} u_{nb}^* + b + \frac{1-\beta}{\beta} a_e u_e^*}{a_e / \beta} \qquad \tilde{v}_n^* = \frac{\sum a_{nb} v_{nb}^* + b + \frac{1-\beta}{\beta} a_n v_n^*}{a_n / \beta}$$

The improved velocity in the correction step is then:

$$u_e = u_e^* + d_e (p_P - p_E) \qquad v_n = \tilde{v}_n^* + d_n (p_P - p_N)$$

where the coefficients a_e, a_n, a_{nb} are determined based on the intermediate velocity u^*, v^*.

2. Analysis of the role of the 2nd relaxation factor

Beta occurs in both denominator and nominator. The effect of Beta in the denominator is predominant.

$$u_e^* = \frac{\sum a_{nb} u_{nb}^* + b + \frac{1-\beta}{\beta} a_e u_e^*}{a_e / \beta}$$

Beta ↓, denominator ↑, hence peusdo-velocity ↓

Beta ↑, denominator ↓, hence peusdo-velocity ↑

Since $u_e = u_e^* + d_e(p_P - p_E)$ $\quad v_n = \tilde{v}_n^* + d_n(p_P - p_N)$

For a certain values of *u* and *v*, decreasing pesudo-v. will leads to an increase in pressure, and vice versa.

Thus:

$\beta \downarrow \quad u_e^*, \tilde{v}_n^* \downarrow \quad \Delta p \uparrow \longrightarrow p \uparrow$

$\beta \uparrow \quad u_e^*, \tilde{v}_n^* \uparrow \quad \Delta p \downarrow \longrightarrow p \downarrow$

Beta can be regarded as a relaxation factor for pressure, and the larger the Beta, the smaller the pressure.

Recommended value of beta:

$$\beta = \begin{cases} 0.5, \ if \ 0 \le \alpha \le 0.5 \\ 1.0, \ if \ 0.5 \le \alpha \le 1.0 \\ > 1 \end{cases}$$

V. is underrelaxed heavily , *p* can be a bit larger;

V. is underrelaxed mildly , *p* is also in middle range;

When iteration is difficult to converge, pressure should be underrelaxed heavily.

7.4.5 Computational procedures of one iteration in CLEAR

Step 1: Assuming an initial velocity field u^0, v^0 ; and calculating the coefficient of the discretized momentum equation and pseudo-velocity:

$$u_e^0 = \frac{\sum a_{nb} u_{nb}^0 + b + \frac{1-\alpha}{\alpha} a_e u_e^0}{a_e / \alpha}$$

$$\tilde{v}_n^0 = \frac{\sum a_{nb} v_{nb}^0 + b + \frac{1-\alpha}{\alpha} a_n v_n^0}{a_n / \alpha}$$

Step 2: Solving the pressure equation and obtaining the temporary pressure field p^*, u_e^0, $\tilde{v}_n^0$ are in the source term *b*;

Step 3: Solving the momentum equation to get *u, *v**;**

Step 4: Recalculating the coefficient of momentum equation and the pseudo-velocity u_e^*, $\tilde{v}_n^*$ based on the intermediate velocity solution u^*, v^*, and solving the equation of the improved pressure *p*; where

in the source term b is calculated from the improved pseudo-velocity u_e^*, $\tilde{v}_n^*$, which are:

$$u_e^* = \frac{\sum a_{nb} u_{nb}^* + b + \frac{1-\beta}{\beta} a_e u_e^*}{a_e / \beta}$$

$$\tilde{v}_n^* = \frac{\sum a_{nb} v_{nb}^* + b + \frac{1-\beta}{\beta} a_n v_n^*}{a_n / \beta}$$

Step 5: Improving the velocity to obtain the solution of the present iteration.

$$u_e = u_e^* + d_e (p_P - p_E) \quad v_n = \tilde{v}_n^* + d_n (p_P - p_N) \quad d_e = \frac{A_e}{a_e / \alpha}$$

Step 6: Taking *u*, *v*, *p* as the solutions of this iteration, returning to step 1 and repeating until convergence is reached.

7.4.6 Discussion on CLEAR algorithm

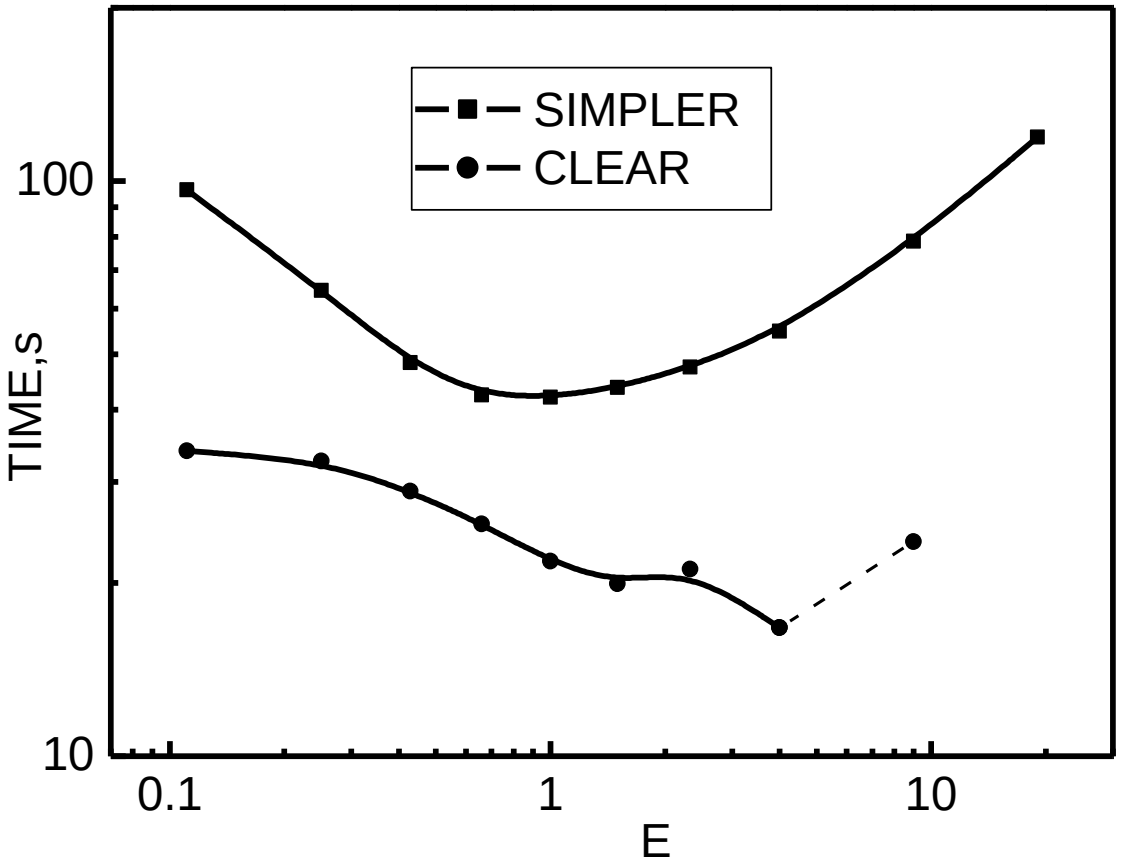
1. In one iteration two pressure equations are solved, while in SIMPLER one pressure equation and one pressure correction equation are solved. Hence the computational costs of CLEAR and SIMPLER are more or less the same, but the improvements made are different. Usually the improvement in CLEAR is better than that in SIMPLER;

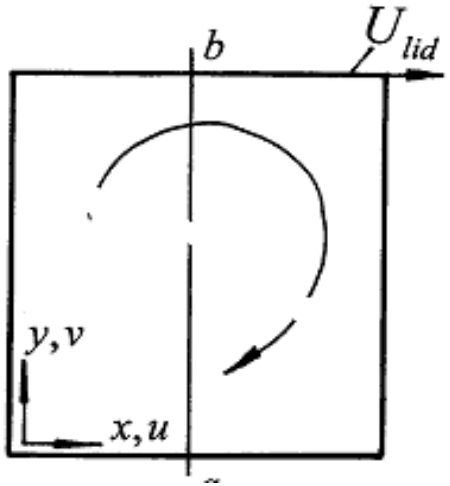
2. In one iteration the coefficients of momentum equation are calculated twice: one for solving the momentum equation, the other for calculation of $u_e^*, \tilde{v}_n^*$, not for solving the momentum equation.

Thus the increase in the computational cost is not significant.

7.4.7 Numerical examples of CLEAR

Example 1

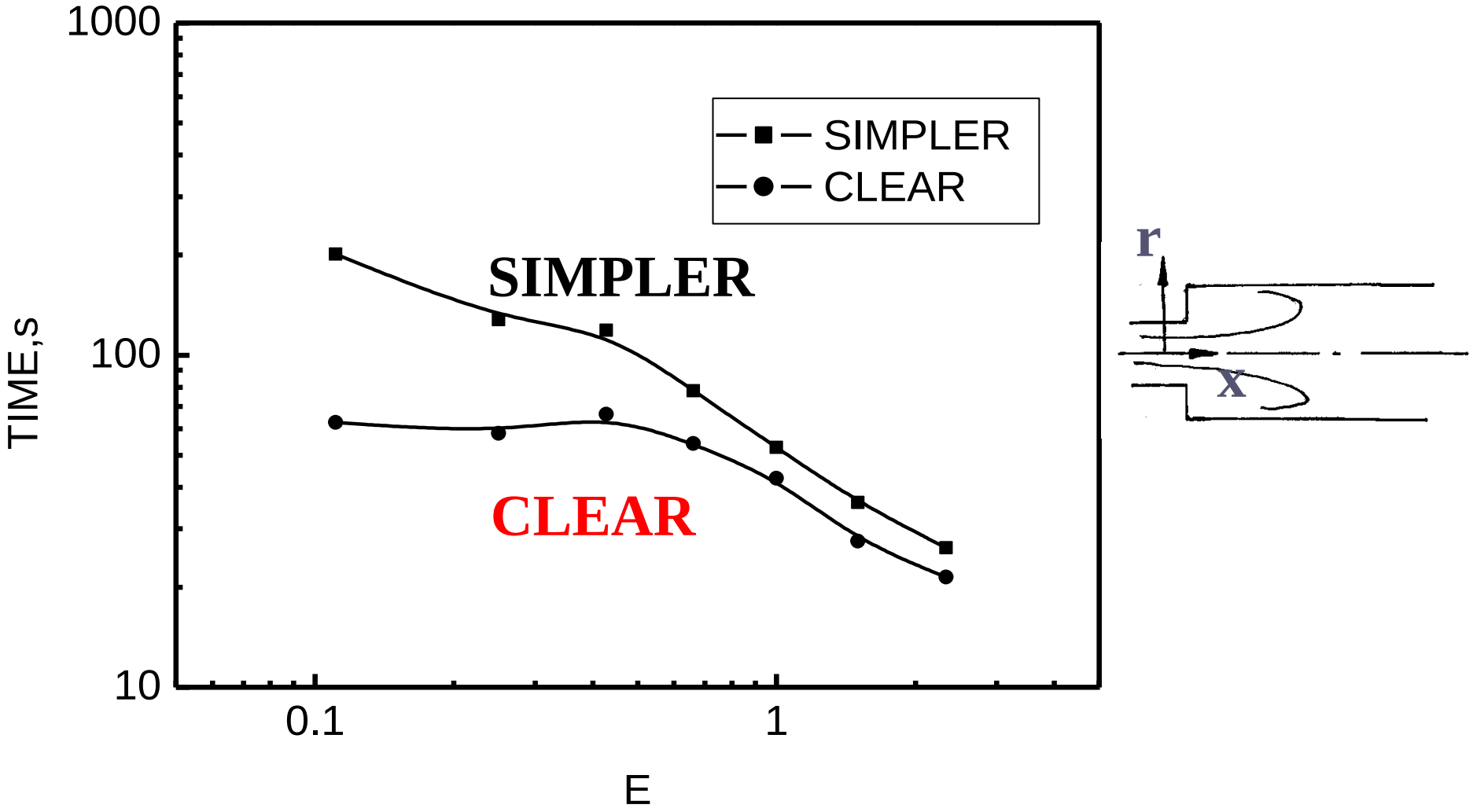




$$E = \frac{\alpha}{1-\alpha}$$

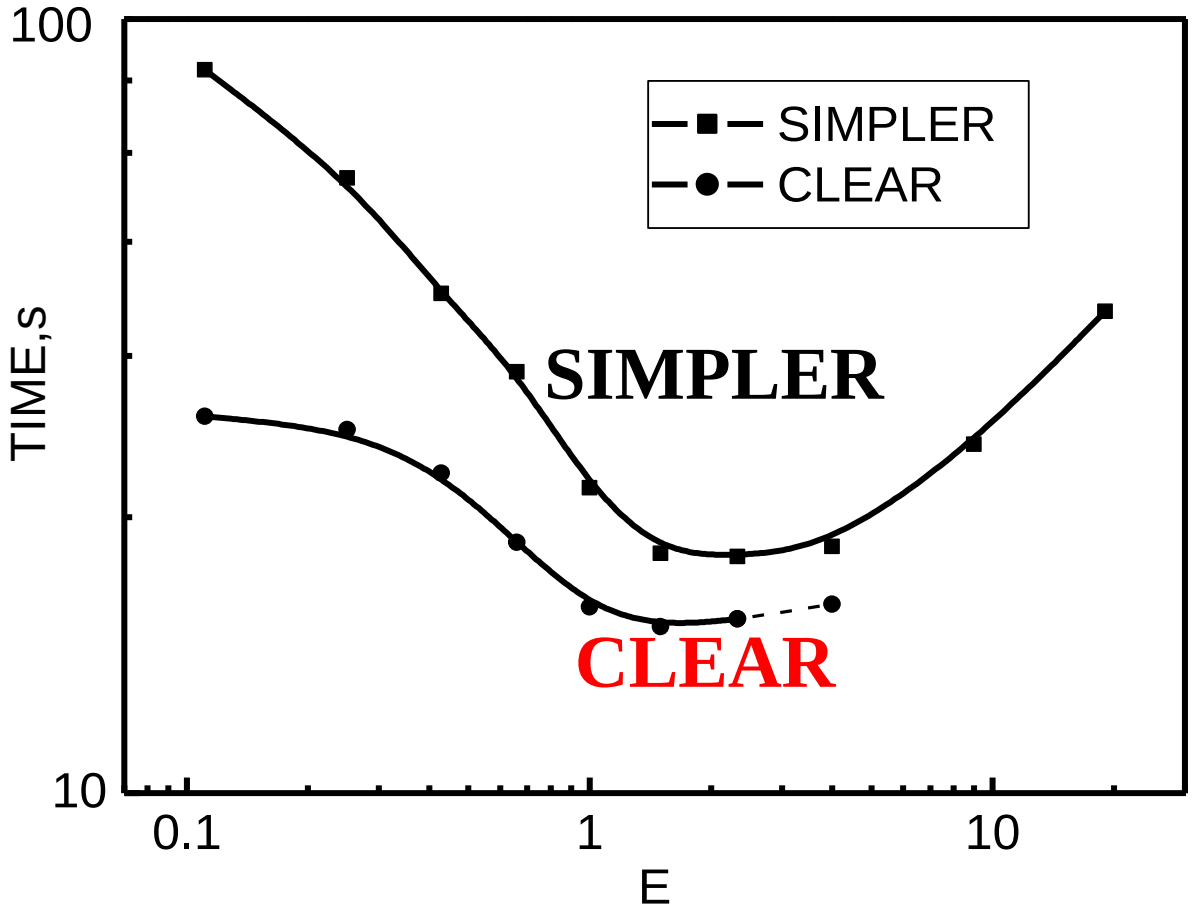
Example 1:Flow in lid-driven cavity (Re=1000)

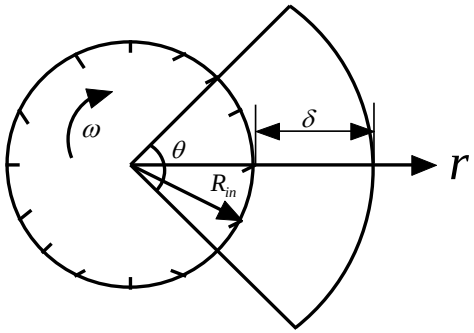
Example 2



Example 2: Heat transfer in a sudden-enlarged tube

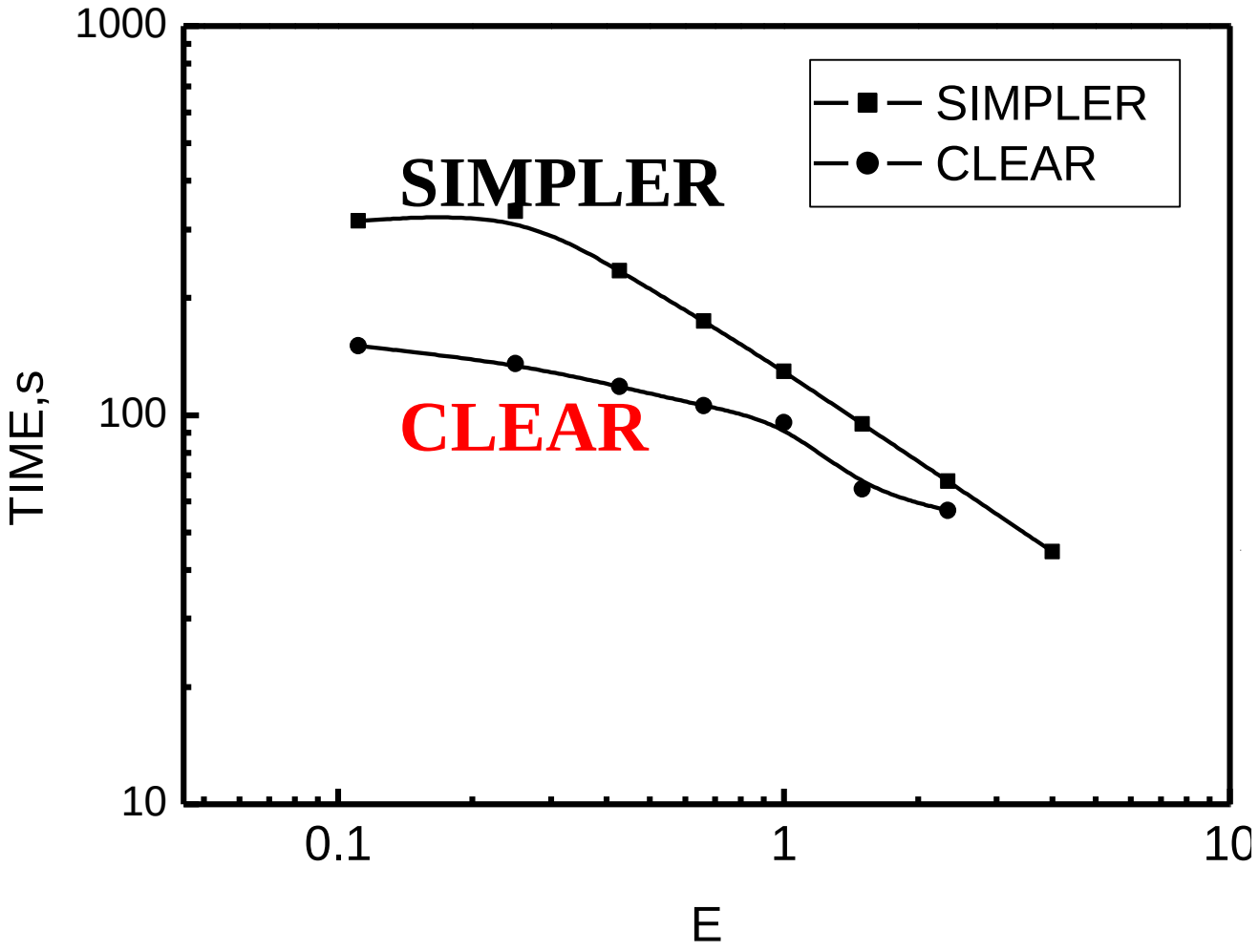
Example 3





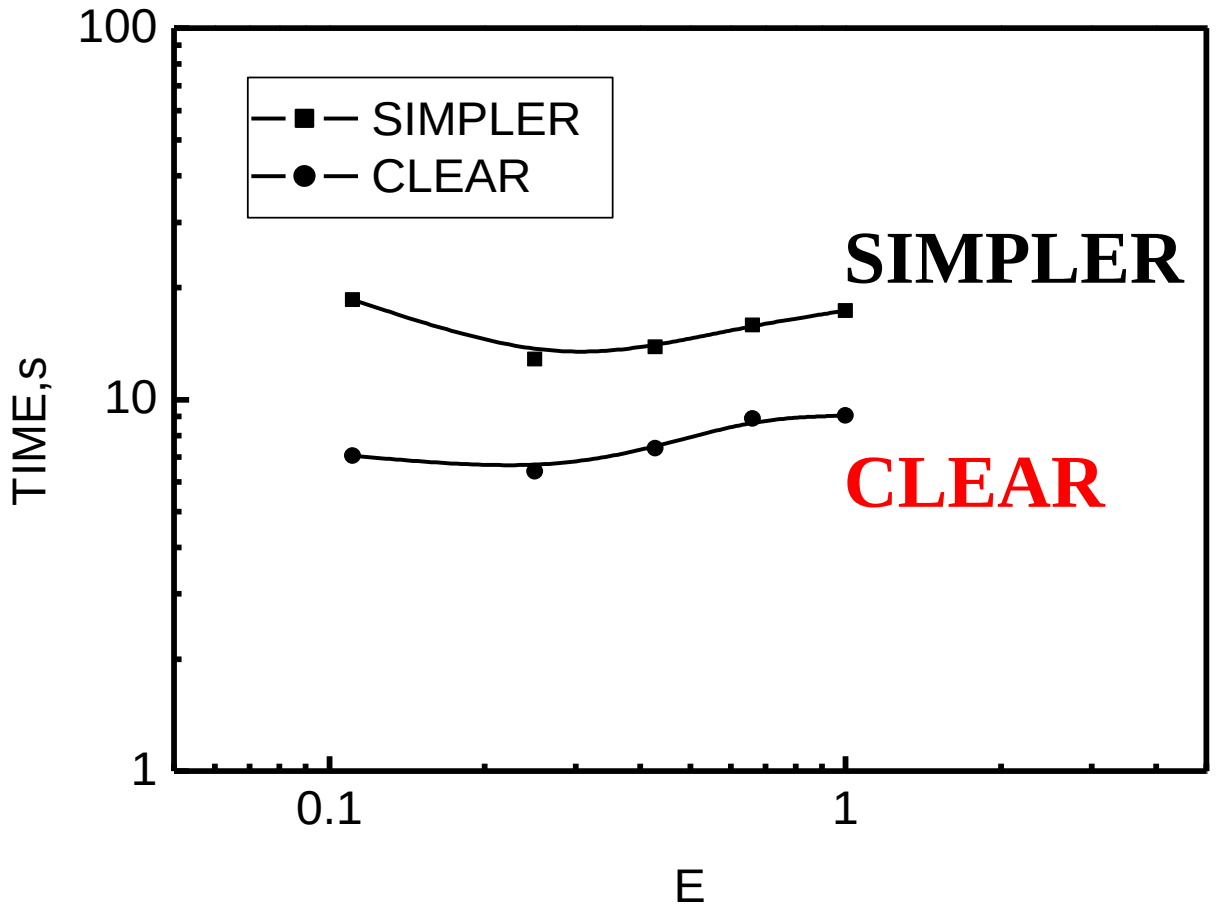
Example 3 Flow in lid-driven annular cavity

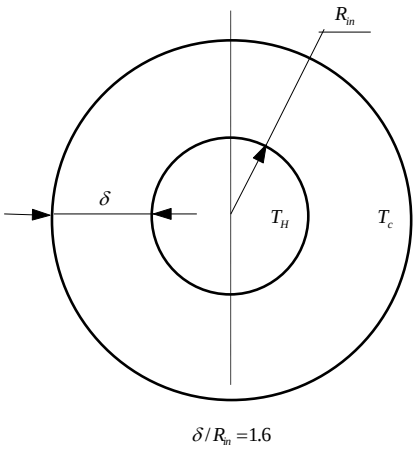
Example 4



Example 4: Flow over rectangular backstep

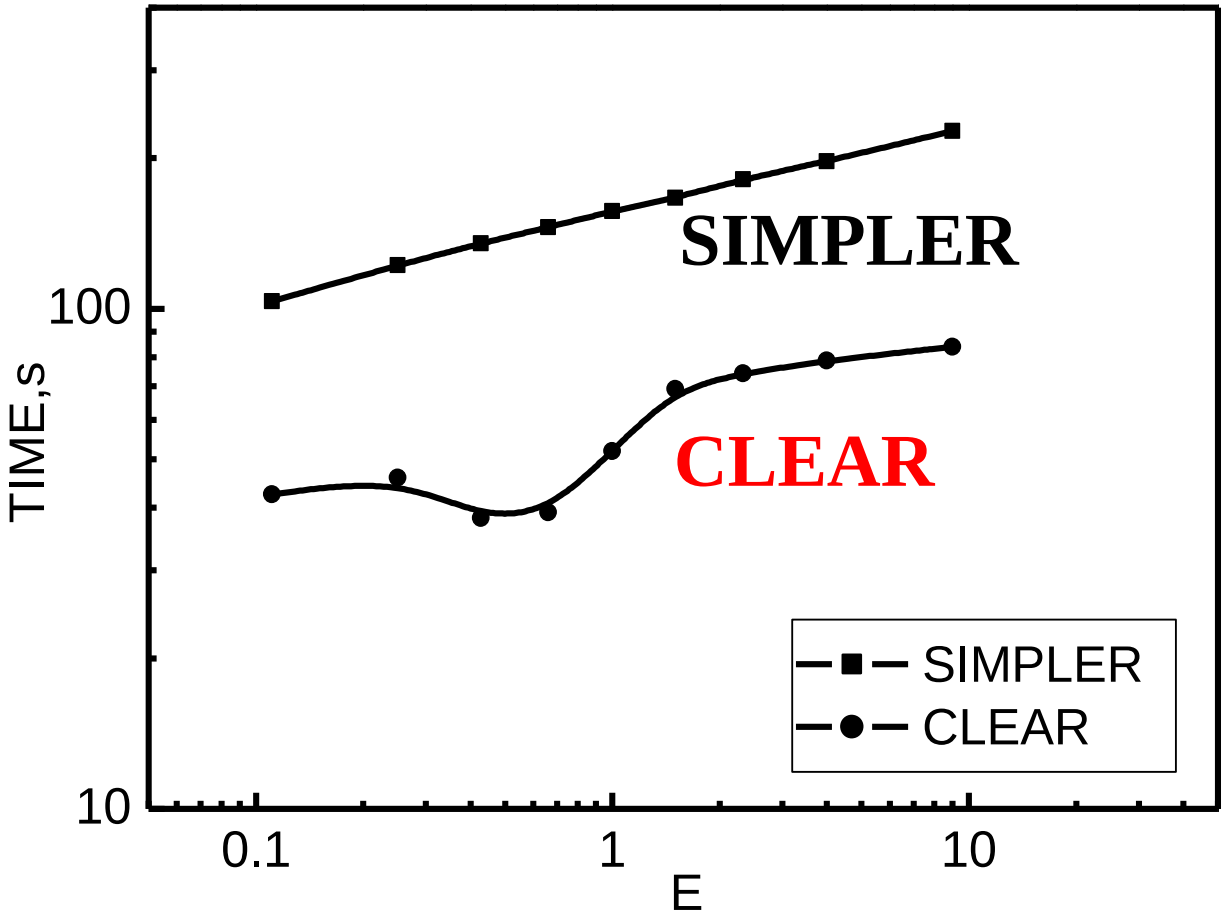
Example 5

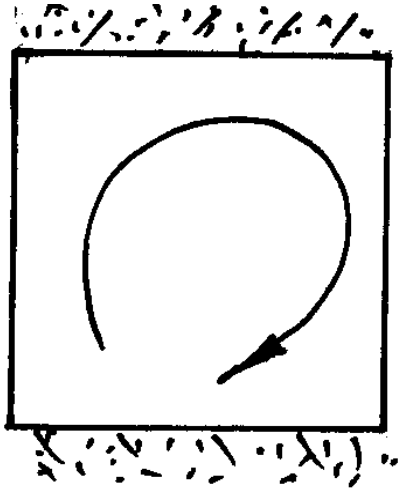




Example 5: Natural convection in annulus

Example 6





Example 6 :Natural convection in square cavity

The six comparison examples in orthogonal coordinates with staggered grids show that ratios of ITER and CPU times of CLEAR over SIMPLER are:

ITER=0.15~0.84; CPU=0.19~0.92

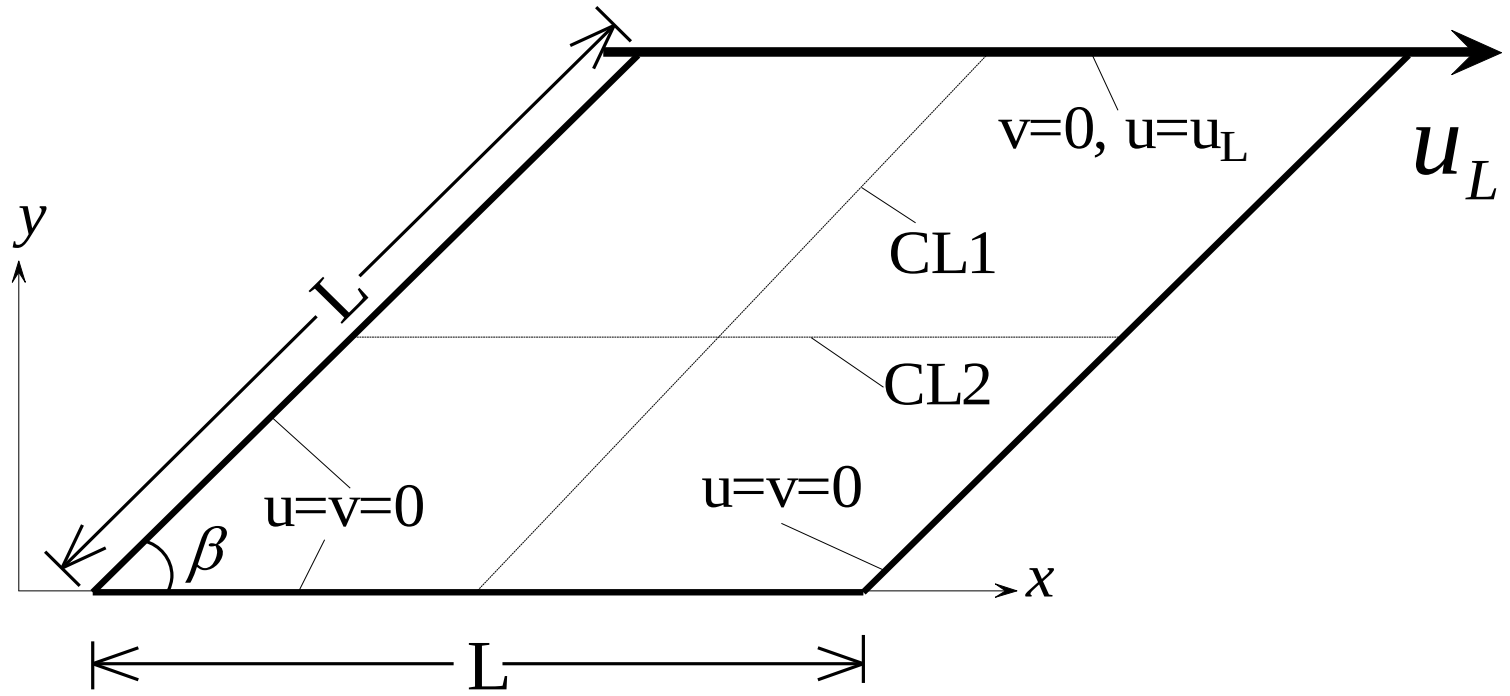
Saving in CPU time is very appreciable.

Extension to non-staggered grid, and non-orthogonal coordinates has been made, and the same results have been obtained.

However, for some cases the robustness of CLEAR is inferior to SIMPLER!

7.4.8 Comparison between SIMPLER and CLEAR in curvilinear non-orthogonal coordinates

Lid-driven flow in a tilted cavity



$\beta = 30^{\circ}$, Re=1000

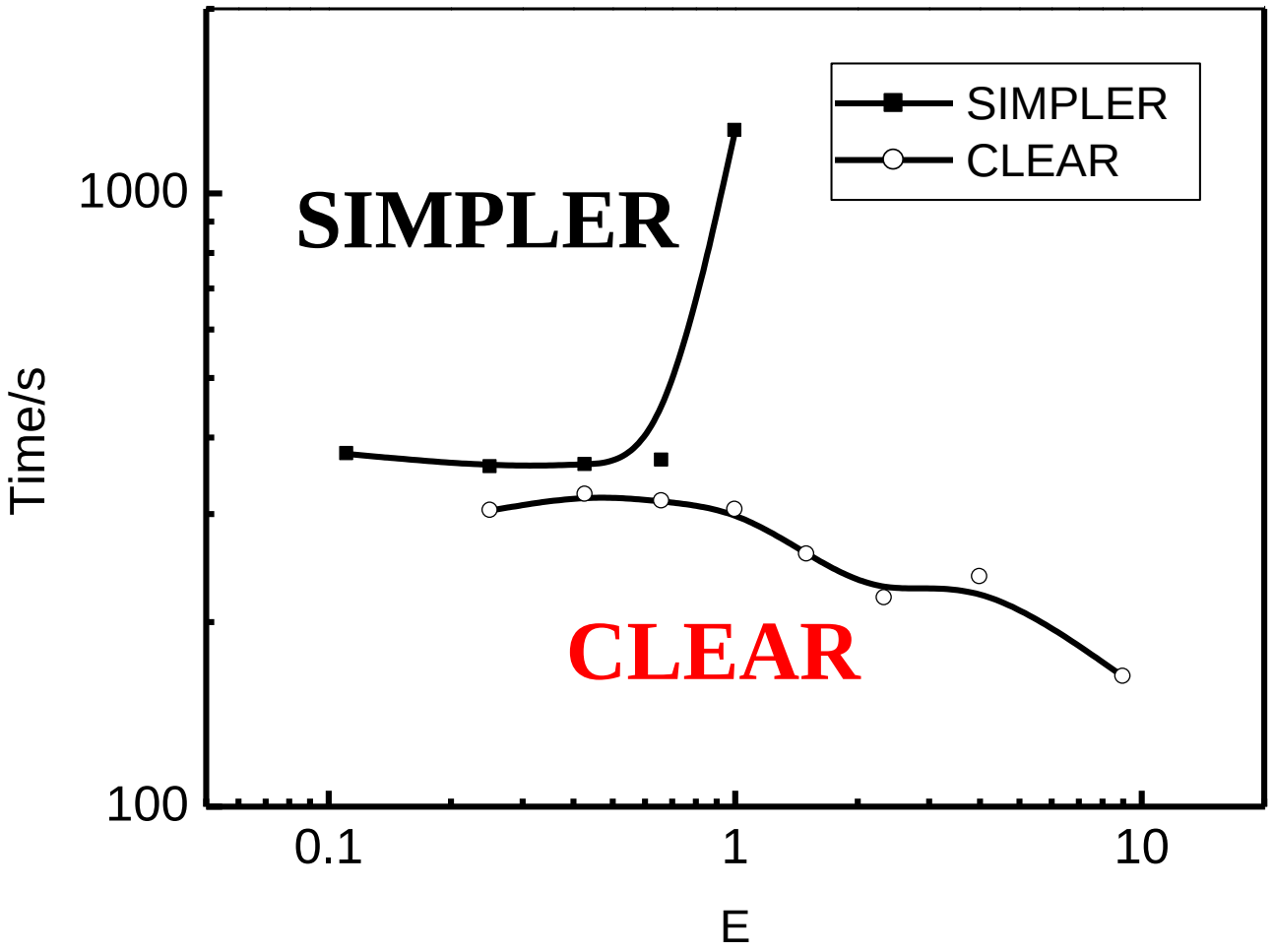
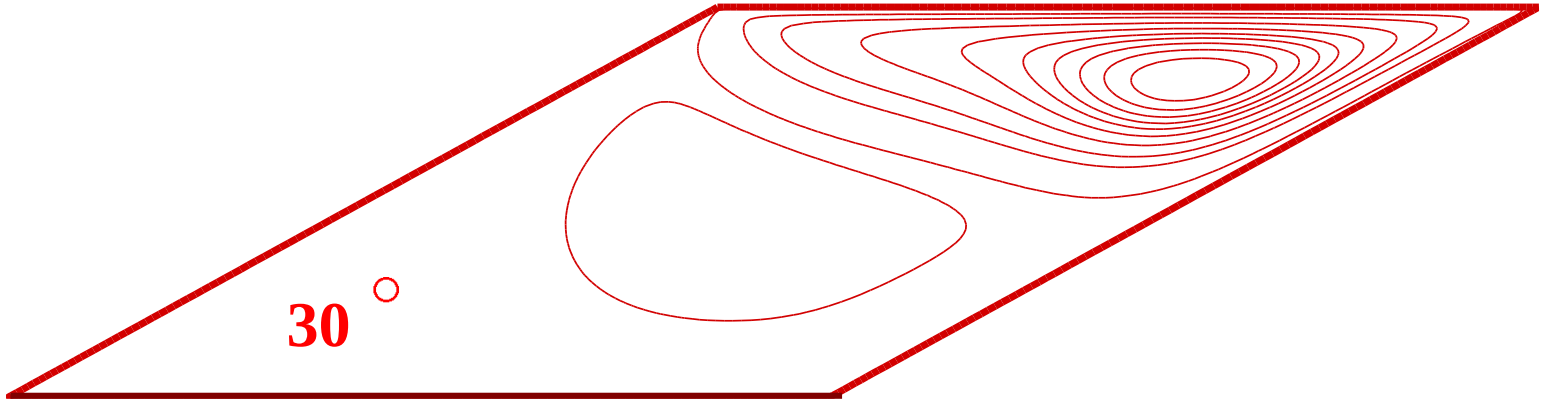


Fig. Comparison of CPU time



The smallest angle at which SIMPLER can work is 30 degrees;

The smallest angle at which CLEAR can work is 5 degrees !

Fig. Comparison in non-orthogonal coordinates

At this angle the grid lines of the two coordinates are nearly parallel, showing very good robustness!

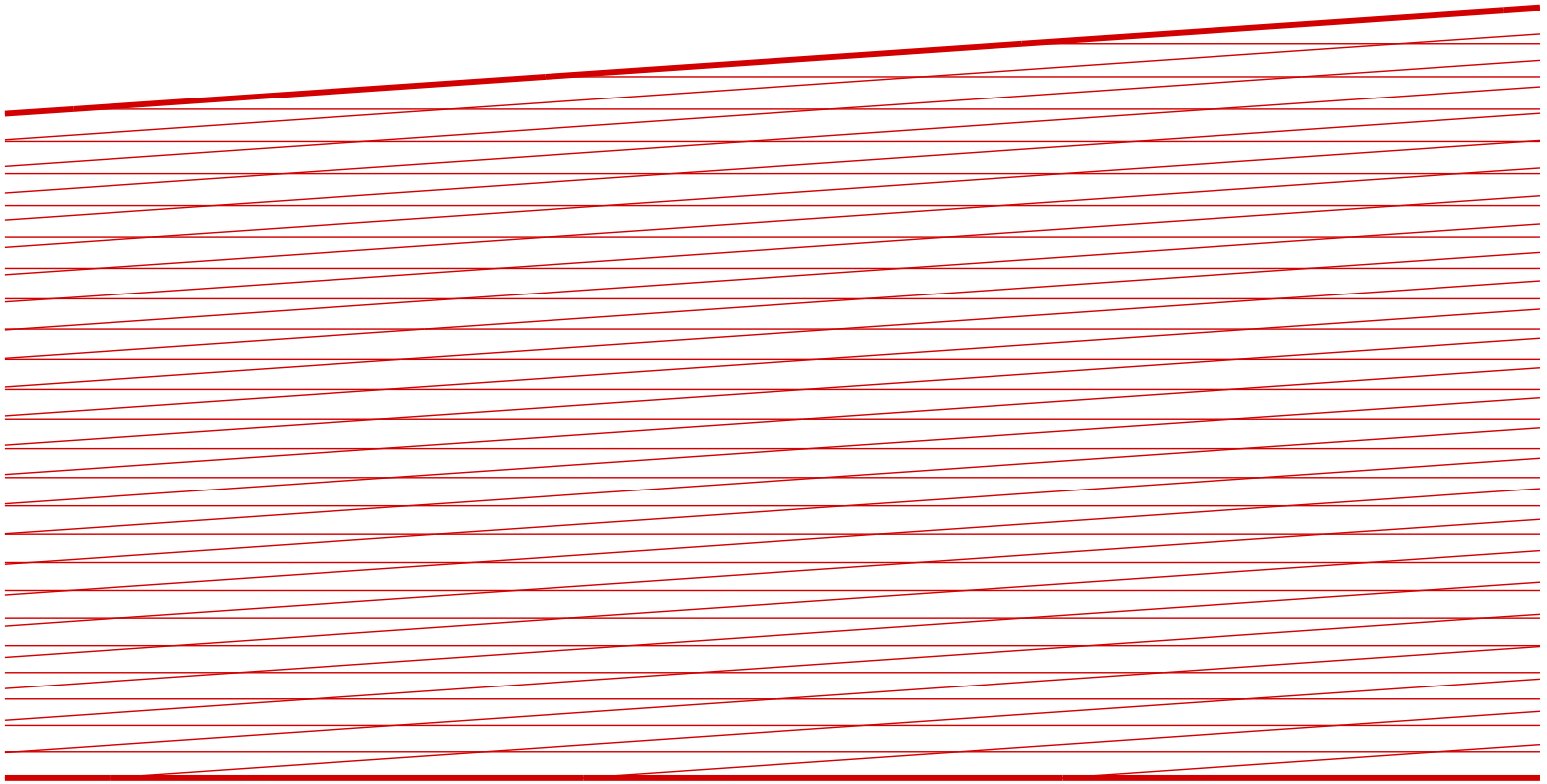


Fig. Grid lines of $\beta = 5^{\circ}$

Tao WQ, Qu ZG, He YL, A novel segregated algorithm for incompressible fluid flow and heat transfer problems - Clear (coupled and linked equations algorithm revised) part II: Application examples, **Numerical Heat Transfer, Part B,** 2004, 45 (1): 19-48

Tao WQ, Qu ZG, He YL , A novel segregated algorithm for incompressible fluid flow and heat transfer problems - Clear (coupled and linked equations algorithm revised) part I: Mathematical formulation and solution procedure ,**Numerical Heat Transfer, Part B,** 2004, 45 (1): 1-17

Qu ZG, Tao WQ, He YL. Implementation of CLEAR algorithm on collocated grid system and application examples. **Numerical Heat Transfer, B,** 2005 (1):65-96.

Qu Z G, He Y L, Zhao C Y, Tao W Q. Implementation of CLEAR algorithm on non-orhtogonal curvilinear coordinates for solution of incompressible flow and heat transfer. **Int J Numerical Methods in Fluids**. 2007, 53:1077-1105

7.5 IDEAL 算法

3.5 IDEAL Algorithm

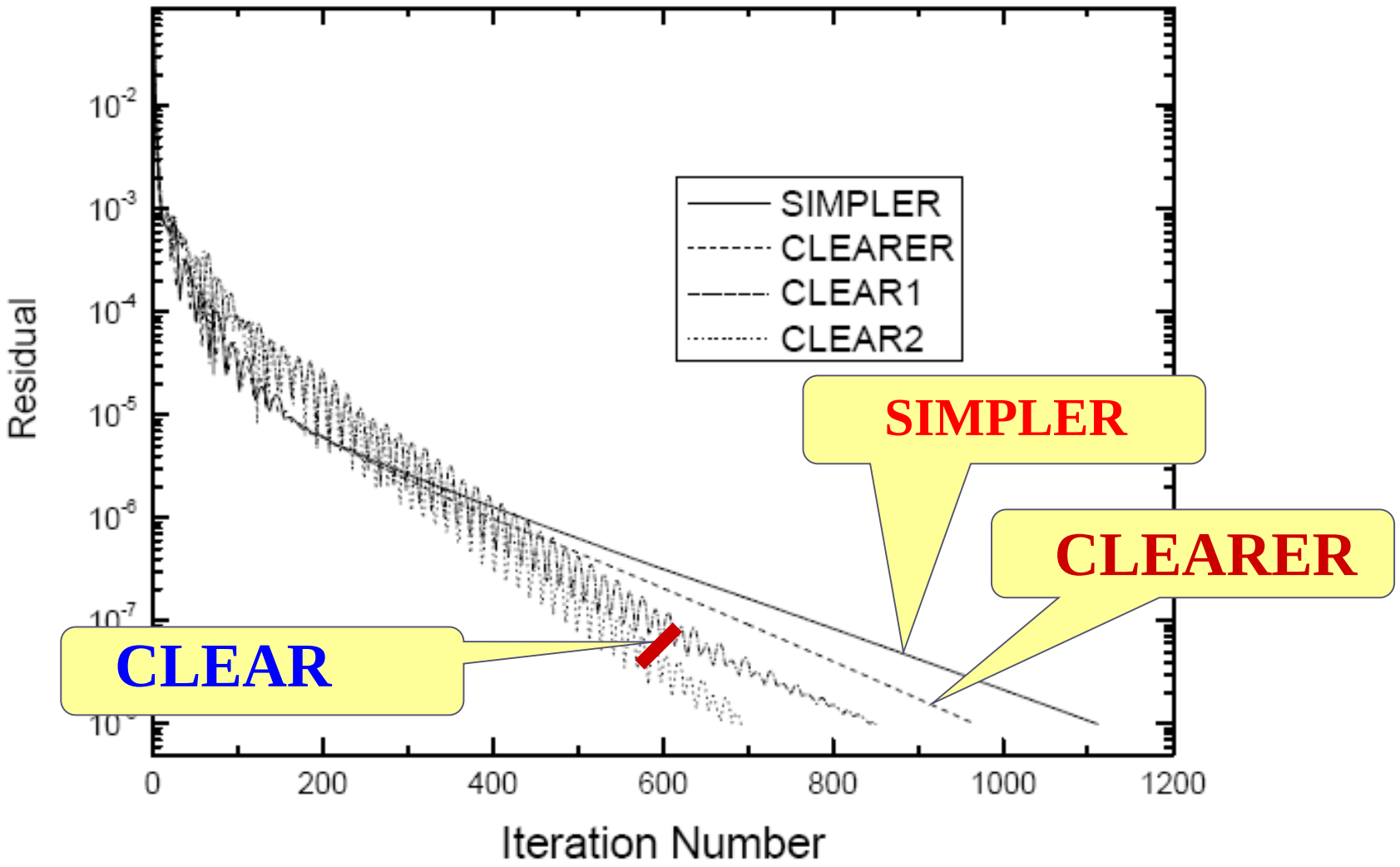
7.5.1 Analysis of the weakness of CLEAR algorithm

The robustness of CLEAR is somewhat worse than SIMPLER. Then what is the reason?

This issue has attracted researchers in the international community. For example, CHENG and LEE in Singapore University presented following analysis and proposed what they call CLEARER algorithm.

For the lid-driven cavity flow they recorded the convergence history of SIMPLER, CLEAR and CLEARER and obtained following results.

Lid-driven cavity flow, Re=1000



They attributed the oscillation of the numerical solution during iteration to the fact that in SIMPLER the revised velocity is composed of two parts: a major part—intermediate velocity u*,v* and the minor part —corrections *u'*, *v'*,

$$u_e = u_e^* + d_e(p_P' - p_E')$$

while in CLEAR the corrected velocity is :

$$u_e = u_e^* + d_e(p_P - p_E) \quad v_n = u_n^* + d_n(p_P - p_N)$$

where the two parts of each equation are estimated in the same order. Thus any oscillation in pressure will lead to the bumpiness of velocity.

They proposed a combination of SIMPLER and CLEAR by computing the revised velocity in the following way:

$$u_e = u_e^* + d_e(p_P^* - p_E^*) + d_e(p_P' - p_E')$$

$$v_n = \tilde{v}_n^* + d_n(p_P^* - p_N^*) + d_n(p_P' - p_N')$$

where *p'* is solved by the pressure correction equation.

This revised version is named as CLEARER.

Cheng Y P, Lee T S, Low H T, Tao W Q. An efficient and robust numerical scheme for the SIMPLER algorithm on non-orthogonal curvilinear coordinates: CLEARER. Numerical Heat Transfer, B, 2007, 51:433-461

Our preliminary considerations

Even though CHENG-LEE's revised version possesses some advantage, our consideration would go another direction:

Since the introduction of the velocity correction components makes the algorithm semi-implicit again and it is our believe that semi-implicit poses some limitation to the convergence .

In the PISO algorithm: one more correction step helps to improve convergence; One more correction step implies a better satisfaction of both momentum and mass conservation!

From CLEAR and PISO we can get a hint: a perfect algorithm should possess following features,

7.5.2 Basic features of an ideal algorithm

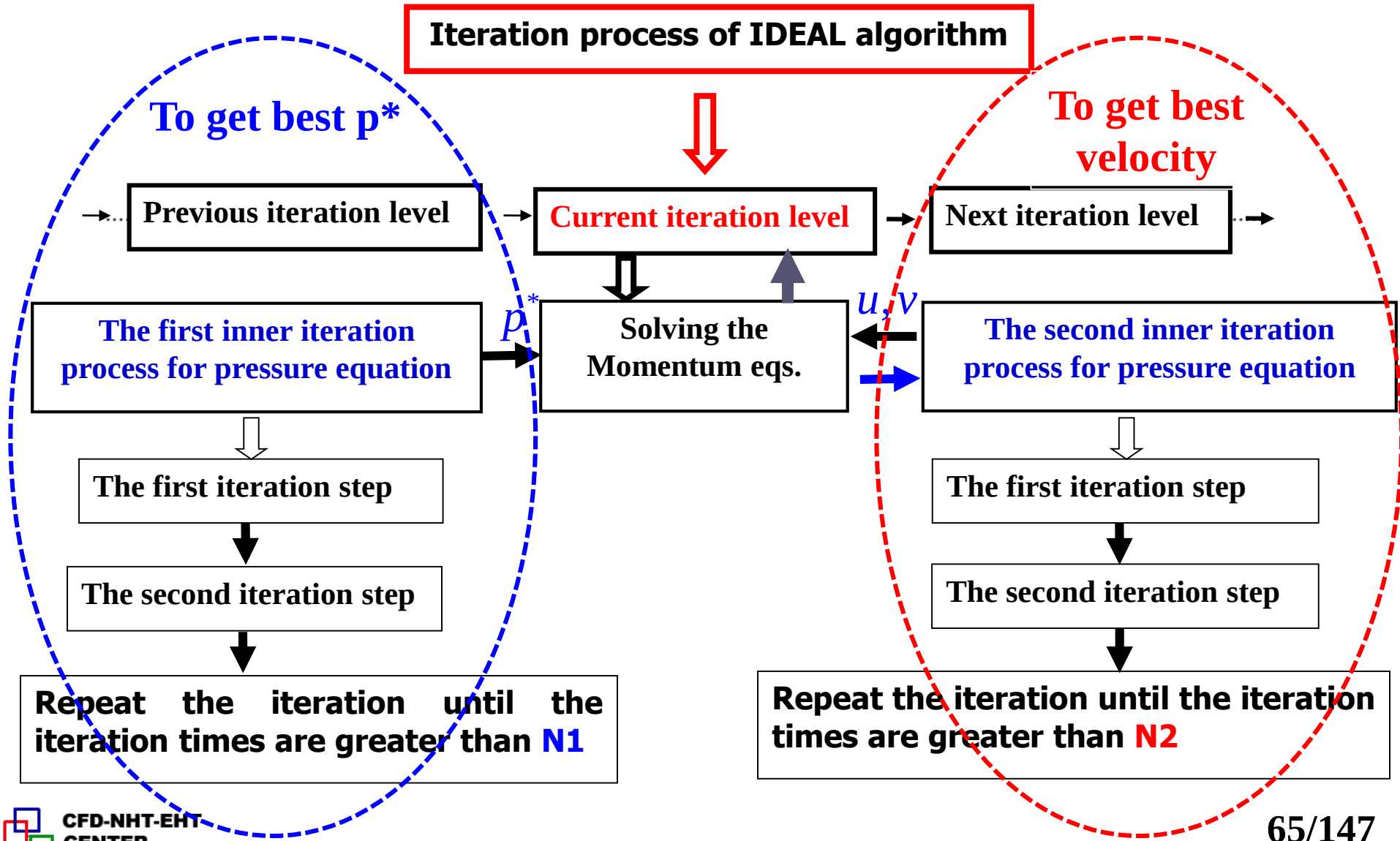
An“ ideal algorithm” should possess three basic features:

(1) The initial pressure field should be close to the final result of the current iteration level as possible as it can.

(2) The final velocity and pressure of each iteration level should satisfy both the momentum and continuity equations as fully as possible. ---from PISO

(3) In every step of iteration both pressure and velocity should be directly solved, no any correction term should be involved. ---from CLEAR

7.5.3 Solution Procedure of the IDEAL Algorithm



The purpose of the two inner iterations are different:

The first inner iteration process for pressure equation is to get a good pressure field for the solution of the momentum equations, simply speaking, to get p^* ; ---Multiple prediction

The second inner iteration process for pressure equation is to improve the velocity fields such that both mass & momentum conservations can be better satisfied. The improvement of velocity field is not proceeded by adding a small correction term, rather, the updated velocity field is obtained via the improved pressure field. ---Multiple correction

7.5.4 The first inner iteration process

$$u_e^{Temp} = \tilde{u}_e^0 + d_e(p_P^{Temp} - p_E^{Temp})$$

$$\tilde{u}_e^0 = \frac{\sum a_{nb} u_{nb}^0 + b}{a_e / \alpha_u}$$

$$v_n^{Temp} = \tilde{v}_n^0 + d_n(p_P^{Temp} - p_N^{Temp})$$

$$\tilde{v}_n^0 = \frac{\sum a_{nb} v_{nb}^0 + b}{a_n / \alpha_v}$$

$$a_P p_P^{Temp} = \sum a_{nb} p_{nb}^{Temp} + b$$

$$b = (\rho\tilde{u}^0 A)_w - (\rho\tilde{u}^0 A)_e + (\rho\tilde{v}^0 A)_s - (\rho\tilde{v}^0 A)_n$$

假定的速度初场

the first ite. step

为简便计亚松弛项已经包括在b 之内

SIMPLER uses this pressure to solve mom. eq.

$$u_e^{Temp} = \tilde{u}_e^{PTemp} + d_e(p_P^{Temp} - p_E^{Temp})$$

$$\tilde{u}_e^{PTemp} = \frac{\sum a_{nb} u_{nb}^{PTemp} + b}{a_e / \alpha_u}$$

$$v_n^{Temp} = \tilde{v}_n^{PTemp} + d_n(p_P^{Temp} - p_N^{Temp})$$

$$\tilde{v}_n^{Ptemp} = \frac{\sum a_{nb} v_{nb}^{Ptemp} + b}{a_n / \alpha_v}$$

$$a_P p_P^{Temp} = \sum a_{nb} p_{nb}^{Temp} + b$$

$$b = (\rho\tilde{u}^{PTemp} A)_w - (\rho\tilde{u}^{PTemp} A)_e + (\rho\tilde{v}^{PTemp} A)_s - (\rho\tilde{v}^{PTemp} A)_n$$

由解出的压力而获得的速度

the second iteration step

N1.....

Taking the last solution of pressure of the 1st inner iteration as the *p for the solution of momentum equation;**

Solving the momentum equation to get the intermediate velocities *u, *v**;**

Based on the intermediate velocities *u, *v** conducting the second inner iteration.**

7.5.5 The second inner iteration process

$$u_e^{Temp} = \tilde{u}_e^* + d_e(p_P^{Temp} - p_E^{Temp}) \qquad \tilde{u}_e^* = \frac{\sum a_{nb} u_{nb}^* + b}{a_e / \alpha_u}$$

$$v_n^{Temp} = \tilde{v}_n^* + d_n(p_P^{Temp} - p_N^{Temp}) \qquad \tilde{v}_n^* = \frac{\sum a_{nb} v_{nb}^* + b}{a_n / \alpha_v}$$

$$a_P p_P^{Temp} = \sum a_{nb} p_{nb}^{Temp} + b$$

$$b = (\rho \tilde{u}^* A)_w - (\rho \tilde{u}^* A)_e + (\rho \tilde{v}^* A)_s - (\rho \tilde{v}^* A)_n$$

由动量方程解出，不满足质量守恒

the first iteration step

质量守恒

SIMPLER-like takes such results as the solution of this iteration and goes to next iteration.

$$u_e^{Temp} = \tilde{u}_e^{PTemp} + d_e(p_P^{Temp} - p_E^{Temp}) \qquad \tilde{u}_e^{PTemp} = \frac{\sum a_{nb} u_{nb}^{PTemp} + b}{a_e / \alpha_u}$$

$$v_n^{Temp} = \tilde{v}_n^{PTemp} + d_n(p_P^{Temp} - p_N^{Temp}) \qquad \tilde{v}_n^{Ptemp} = \frac{\sum a_{nb} v_{nb}^{Ptemp} + b}{a_n / \alpha_v}$$

$$a_P p_P^{Temp} = \sum a_{nb} p_{nb}^{Temp} + b$$

$$b = (\rho \tilde{u}^{PTemp} A)_w - (\rho \tilde{u}^{PTemp} A)_e + (\rho \tilde{v}^{PTemp} A)_s - (\rho \tilde{v}^{PTemp} A)_n$$

由解出的压力而获得的速度

the second iteration step

N2……

Taking the last solution of pressure of the 2nd inner iteration to improve the solution of velocity such that both the mass conservation and momentum equations are expected to be satisfied quite well:

$$u_e = \tilde{u}_e^{PTemp} + d_e(p_P - p_E)$$

$$v_n = \tilde{v}_n^{PTemp} + d_n(p_P - p_N)$$

7.5.6 Three kinds of "iteration" in IDEAL

1. The 1st kind: Outer iteration (for nonlinearity) controlled by ITER, to up-date the coefficient of momentum equations;

2. The 2nd kind: Two Inner iteration (for pressure & velocity) controlled by N1,N2, to find a good initial pressure and to obtain a velocity field which can fairly well satisfy both mass and momentum conservations;

3. The 3rd kind: Inner iteration (for solving algebraic equations) controlled by NTIMES, to solve the algebraic equation by ADI with block correction technique.

7.5.7 Discussion of the IDEAL algorithm

1. The pressure and velocity equations are solved by inner doubly-iterative methods. Thus it is called **I**nner **D**oubly-iterative **E**fficient **A**lgorithm for **L**inked-equations **(IDEAL)**

2. The conservation condition of mass and momentum is almost fully satisfied at each iteration level, the velocity and pressure under-relaxation factors may be set to a quite large value and need not to be adjusted. A value of 0.9 (**and even 1.0**) for both the velocity and pressure under-relaxation factors can be used.

3. The inner doubly-iterative times N1, N2 need to be adjusted so that the conservation of mass and momentum can be almost fully guaranteed at each iteration level. In our study, the inner doubly-iterative times N1 and N2 are usually set as 4;

For some situations such as very high-*Re*/*Ra* or very fine-mesh flow cases larger values may be needed.

In IDEAL algorithm the convergence and stability of iteration process can be controlled by adjusting the inner doubly-iterative times N1, N2.

4. The IDEAL algorithm can be regarded as a multi-step solution algorithm in both prediction and correction stages.

7.5.8 Application Examples of IDEAL

Comparison conditions

- **(1) Discretization scheme;**
- **(2) Solution of the algebraic equations;**
- **(3) Under-relaxation factor;**
- **(4) Grid system;**
- **(5) Convergence criterion:**

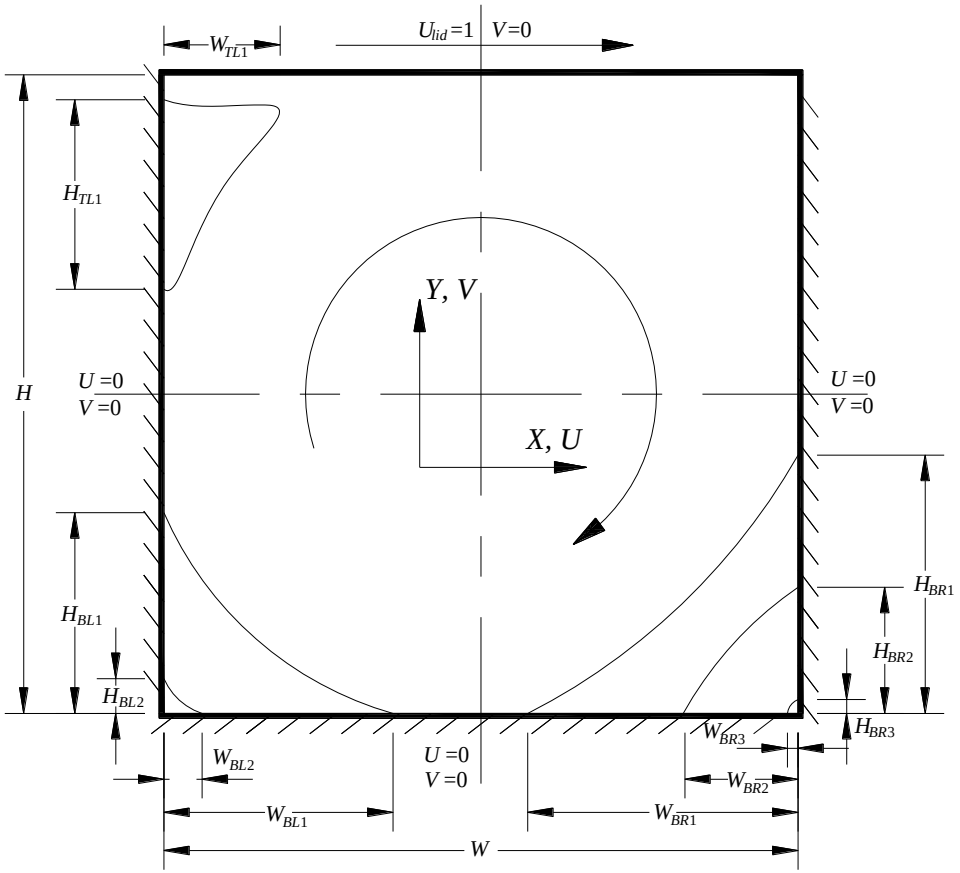
$$\mathrm{Rs}_{\mathrm{Mass}} = \frac{\mathrm{MAX}[|(\rho u^* A)_w - (\rho u^* A)_e + (\rho v^* A)_s - (\rho v^* A)_n|]}{q_m}$$

$$\mathrm{Rs}_{V\mathrm{Mom}} = \frac{\mathrm{MAX}\{|a_n v_n^0 - [\sum_{nb} a_{nb} v_{nb}^0 + b + A_n (p_P - p_N)]|\}}{\rho u_m^2}$$

$$\mathrm{Rs}_{U\mathrm{Mom}} = \frac{\mathrm{MAX}\{|a_e u_e^0 - [\sum_{nb} a_{nb} u_{nb}^0 + b + A_e (p_P - p_E)]|\}}{\rho u_m^2}$$

Problem 1: 2D lid-driven cavity flow in a square cavity

For *Re*=100 ~10000, grid numbers = 129× 129 ~ 260 × 260.

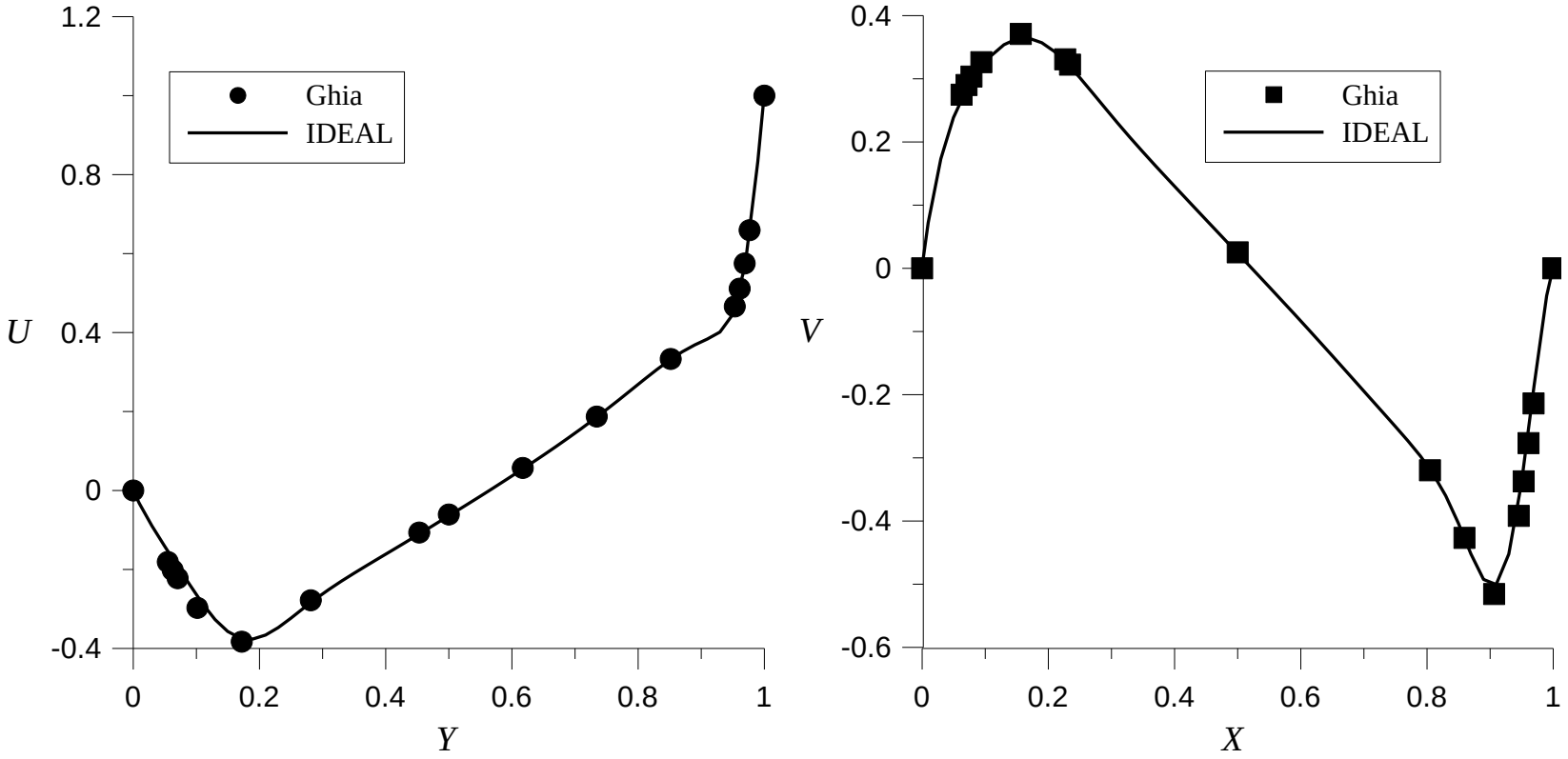


Grid Numbers	Comparison terms		*Re*						
			100	400	1000	3200	5000	7500	10000
52×52	Time(s)	IDEAL	2.89	2.39	2.03	---	---	---	---
		SIMPLER	7.92	7.35	16.14	---	---	---	---
		Ratio	0.365	0.325	0.126	---	---	---	---
	Iteration Numbers	IDEAL	234	184	163	---	---	---	---
		SIMPLER	1264	1152	2604	---	---	---	---
		Ratio	0.185	0.160	0.063	---	---	---	---
	N1, N2 used in IDEAL		4, 4	4, 4	4, 4	---	---	---	---

Grid Numbers	Comparison terms		*Re* 100	400	1000	3200	5000	7500	10000
82×82	Time(s)	IDEAL	18.95	13.57	12.64	---	---	---	---
		SIMPLER	56.7	46.95	69.7	---	---	---	---
		Ratio	0.334	0.289	0.181	---	---	---	---
	Iteration Numbers	IDEAL	499	356	334	---	---	---	---
		SIMPLER	3171	2623	3911	---	---	---	---
		Ratio	0.157	0.136	0.085	---	---	---	---
	N1, N2 used in IDEAL		4, 4	4, 4	4, 4	---	---	---	---

Grid Numbers	Comparison terms		*Re* 100	400	1000	3200	5000	7500	10000
130 × 130	Time(s)	IDEAL	C	C	C	C	C	---	---
		SIMPLER	D	D	D	D	D	---	---
		Ratio	0	0	0	0	0	---	---
	Iteration Numbers	IDEAL	C	C	C	C	C	---	---
		SIMPLER	D	D	D	D	D	---	---
		Ratio	0	0	0	0	0	---	---
	N1, N2 used in IDEAL		5, 5	5, 5	5, 5	5, 5	5, 5	---	---

Grid Numbers	Comparison terms		*Re* 100	400	1000	3200	5000	7500	10000
260 × 260	Time(s)	IDEAL	C	C	C	C	C	C	C
		SIMPLER	D	D	D	D	D	D	D
		Ratio	0	0	0	0	0	0	0
	Iteration Numbers	IDEAL	C	C	C	C	C	C	C
		SIMPLER	D	D	D	D	D	D	D
		Ratio	0	0	0	0	0	0	0
	N1, N2 used in IDEAL		10,10	10,10	10,10	10,10	10,10	10, 10	10, 10



(a) **(b)**

Fig. Predicted velocity distributions for *Re*=1000 and gird numbers=52×52

***(a)* *U* component distribution along *X*=0.5;**

***(b)* *V* component distribution along *Y*=0.5.**

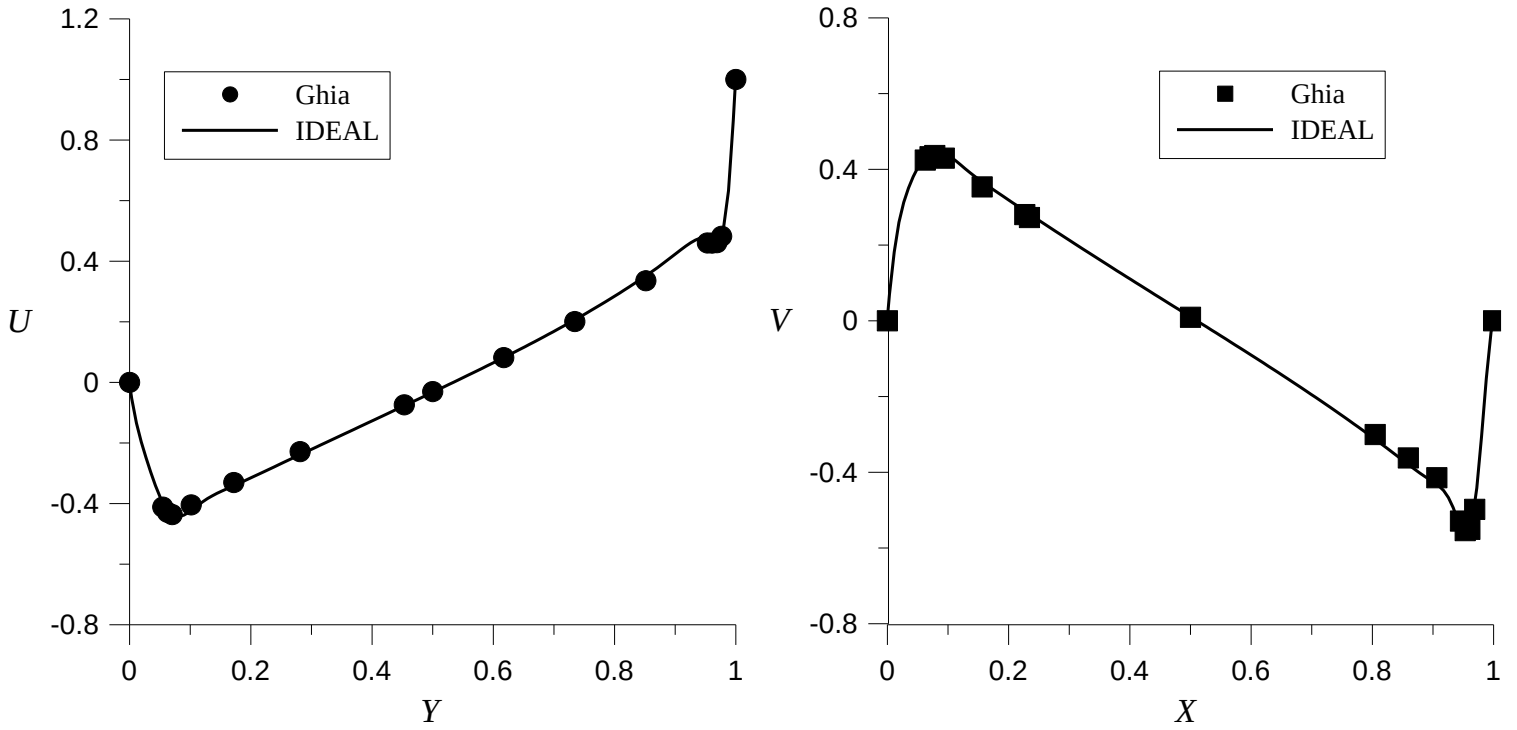


Fig. Predicted velocity distributions for *Re*=5000 and gird numbers=130×130

***(a) U* component distribution along *X*=0.5;**

***(b) V* component distribution along *Y*=0.5.**

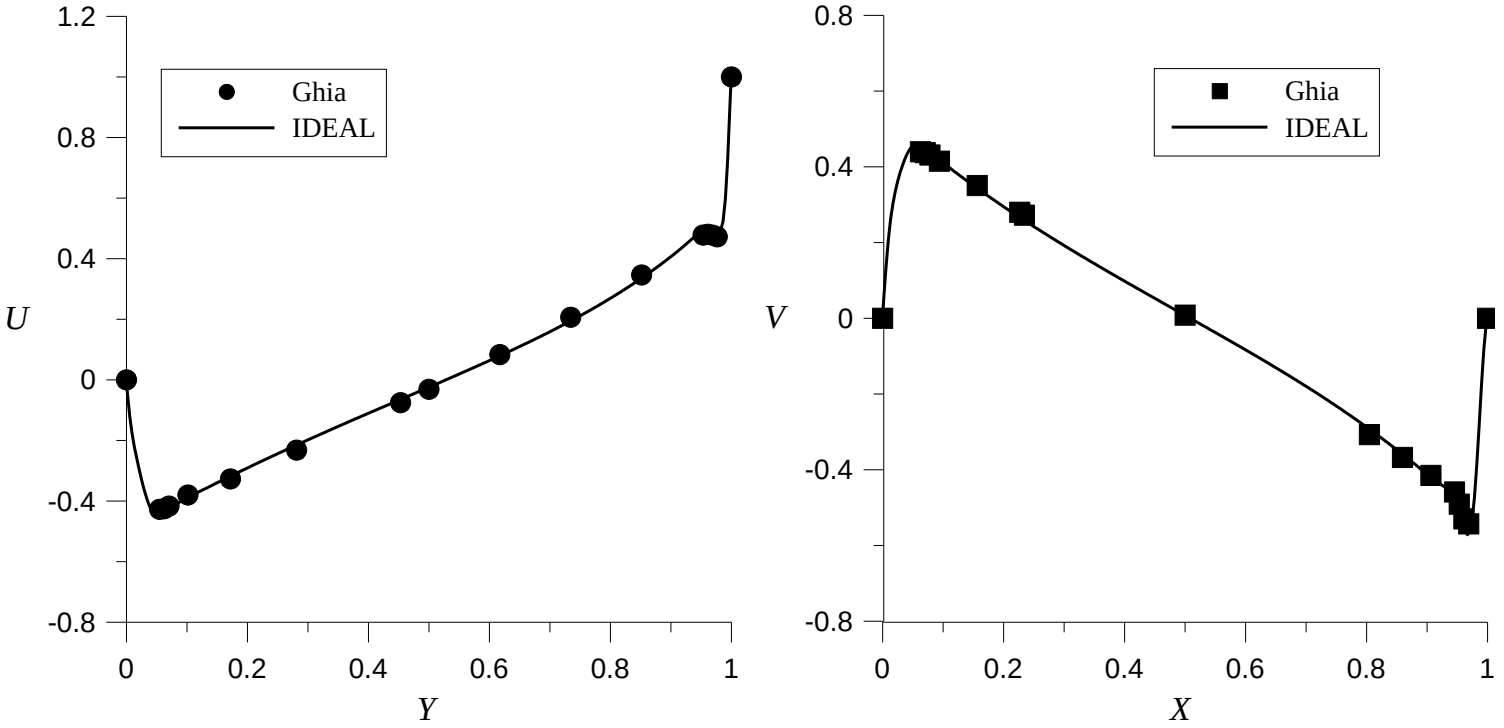


Fig. Predicted velocity distributions for *Re*=10000 and gird numbers=260×260
(a) *U* component distribution along *X*=0.5;
(b) *V* component distribution along *Y*=0.5.

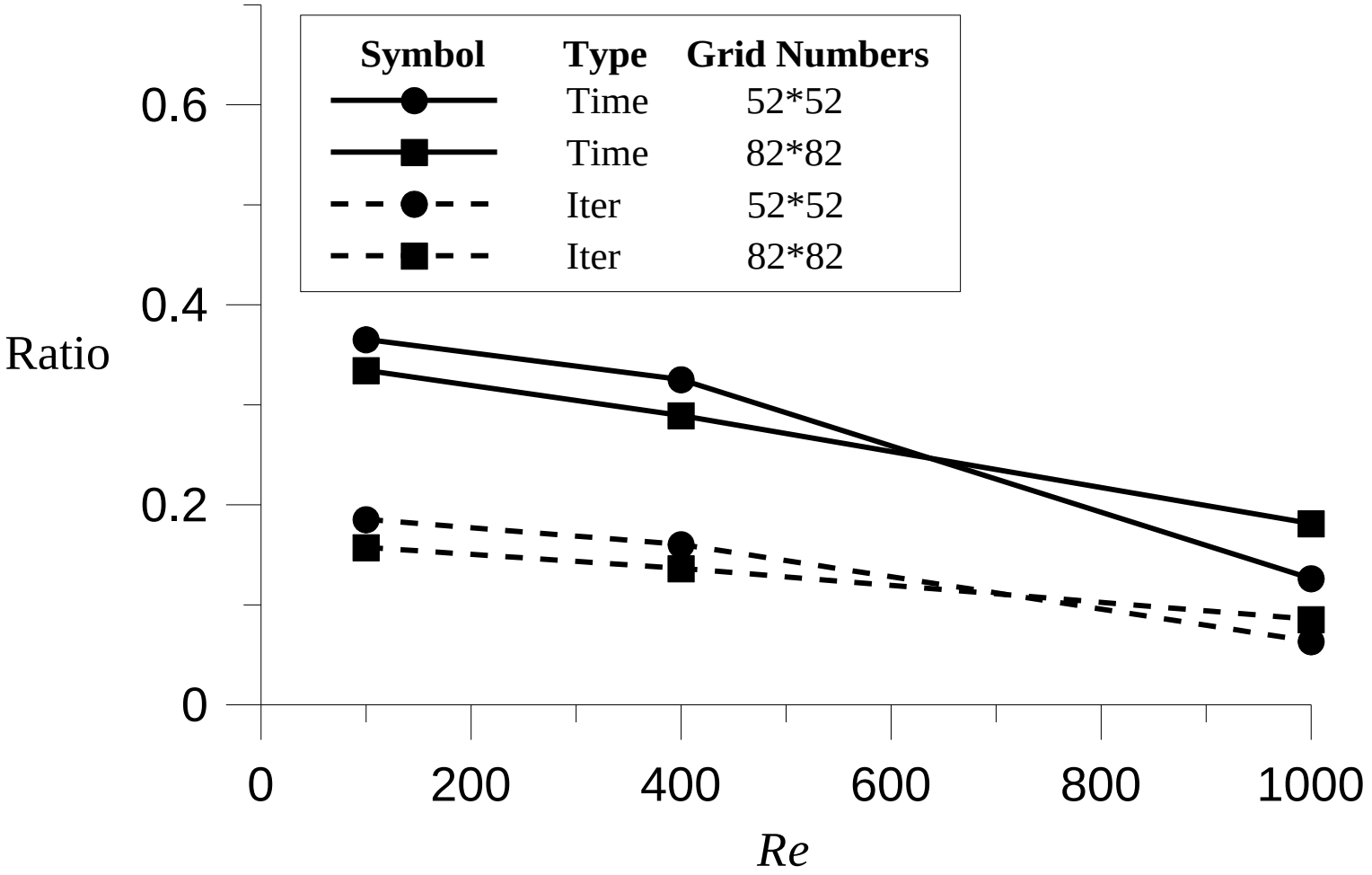


Fig. Ratios of CPU time and iteration numbers

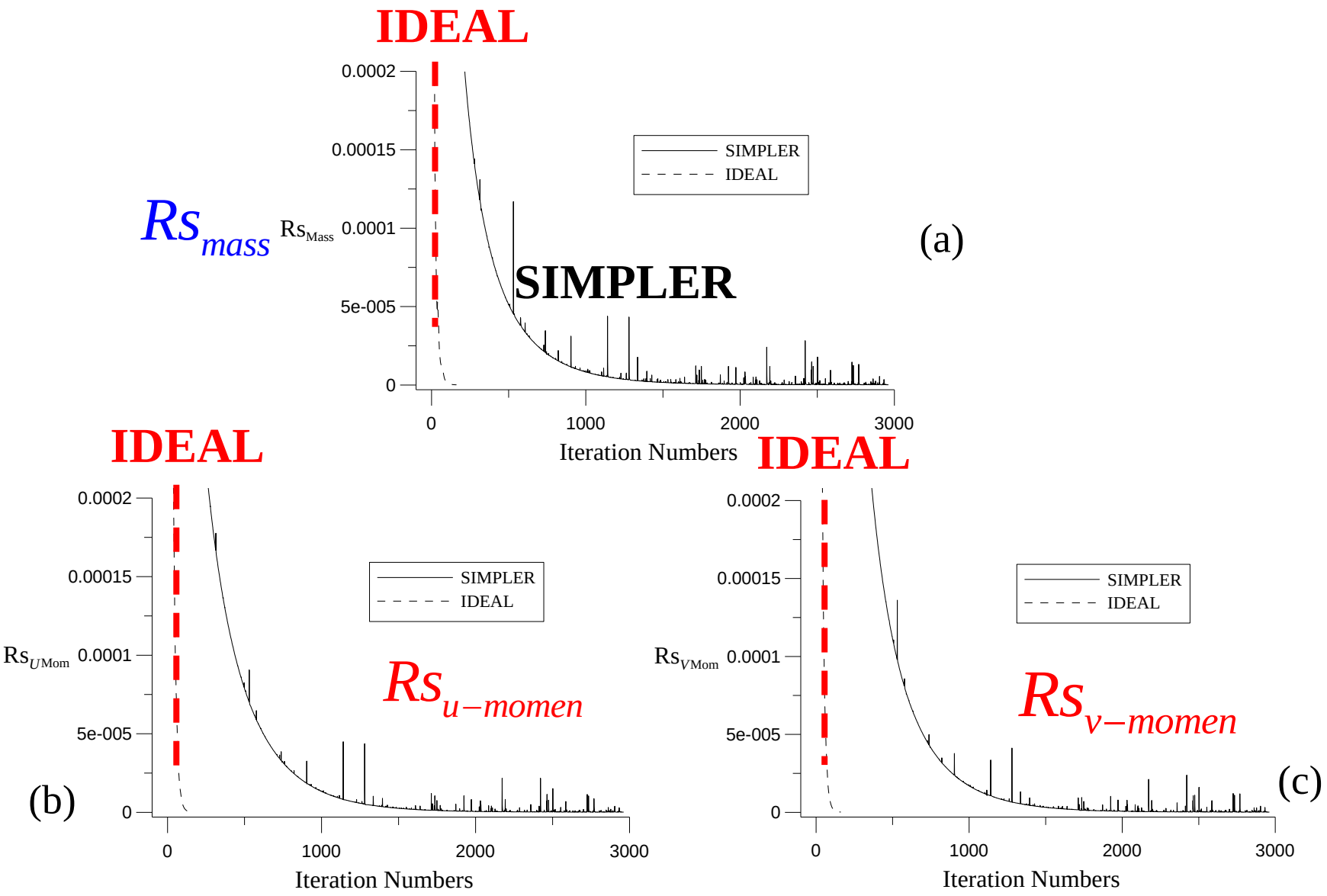
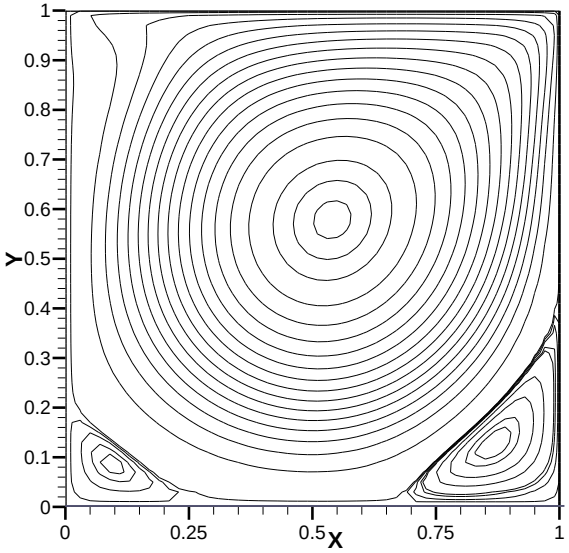
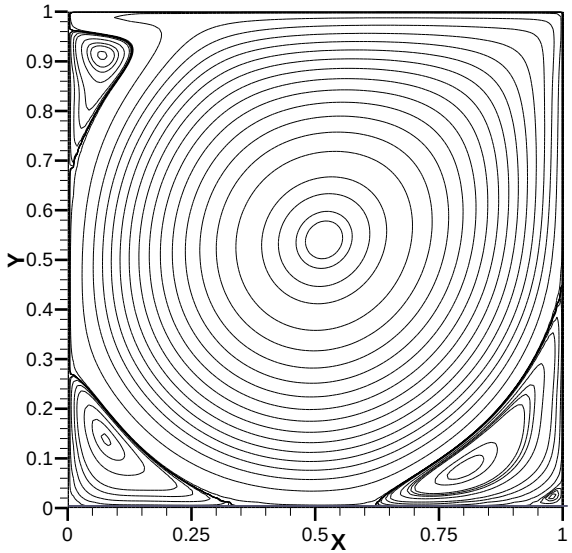


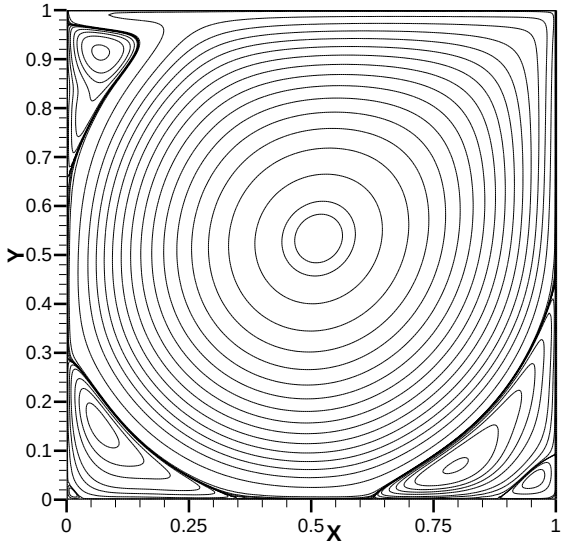
Fig. Convergence history at Re=1000, 52×52



(a) *Re*=1000, 52×52;



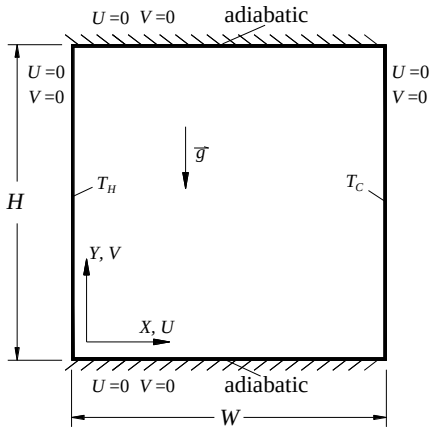
(b) *Re*=5000, 130×130;

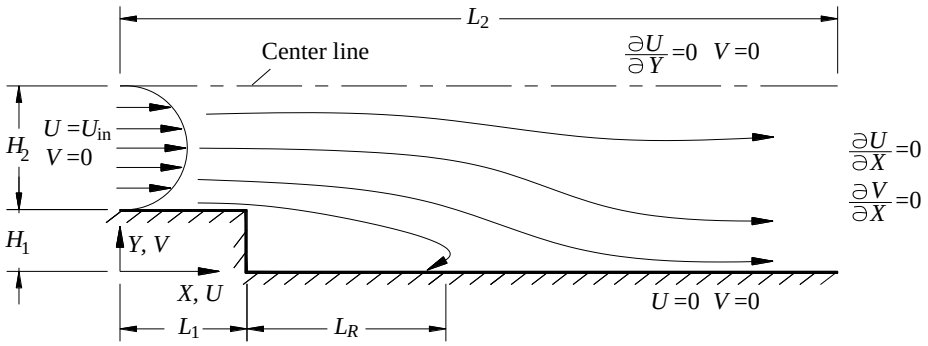


(c) *Re*=7500, 260×260;

Fig. Predicted flow fields at *Re*=1000 to 7500

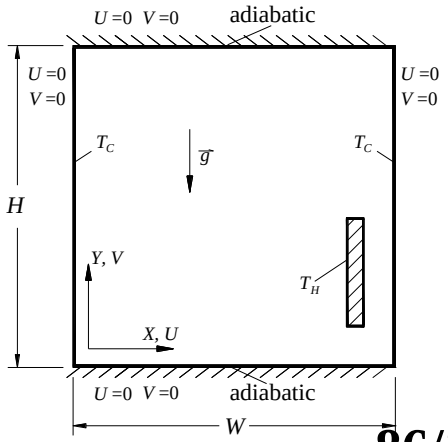
Problem 2: 2D natural convection in a square cavity



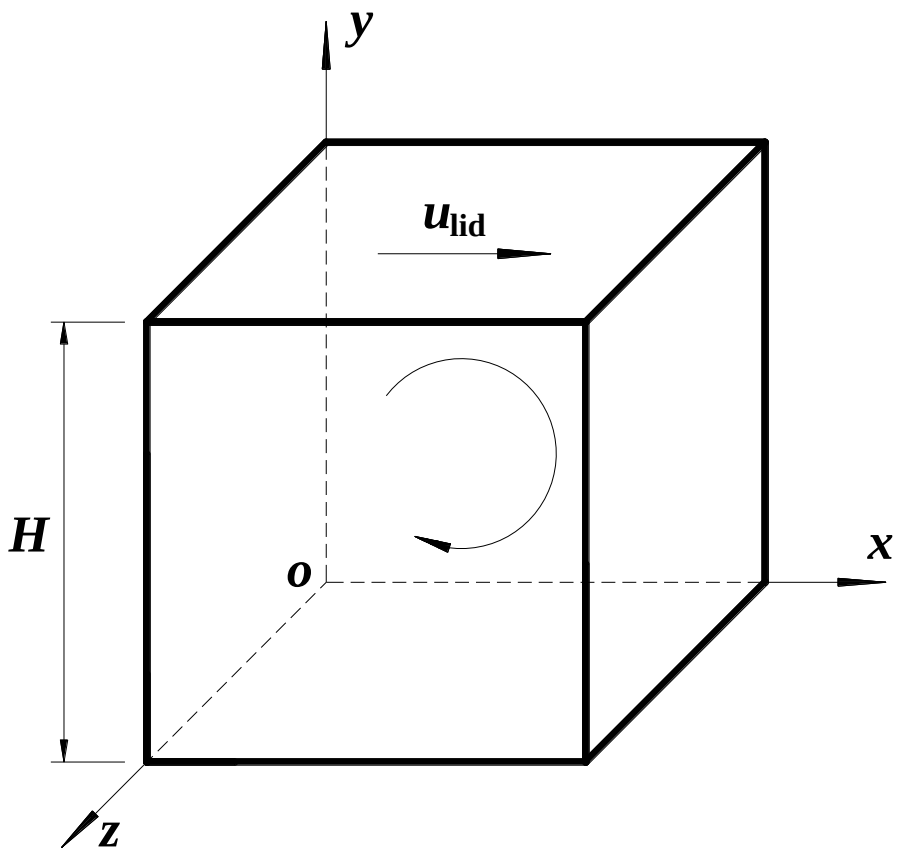


Problem 3: 2D laminar fluid flow over a rectangular backward-facing Step

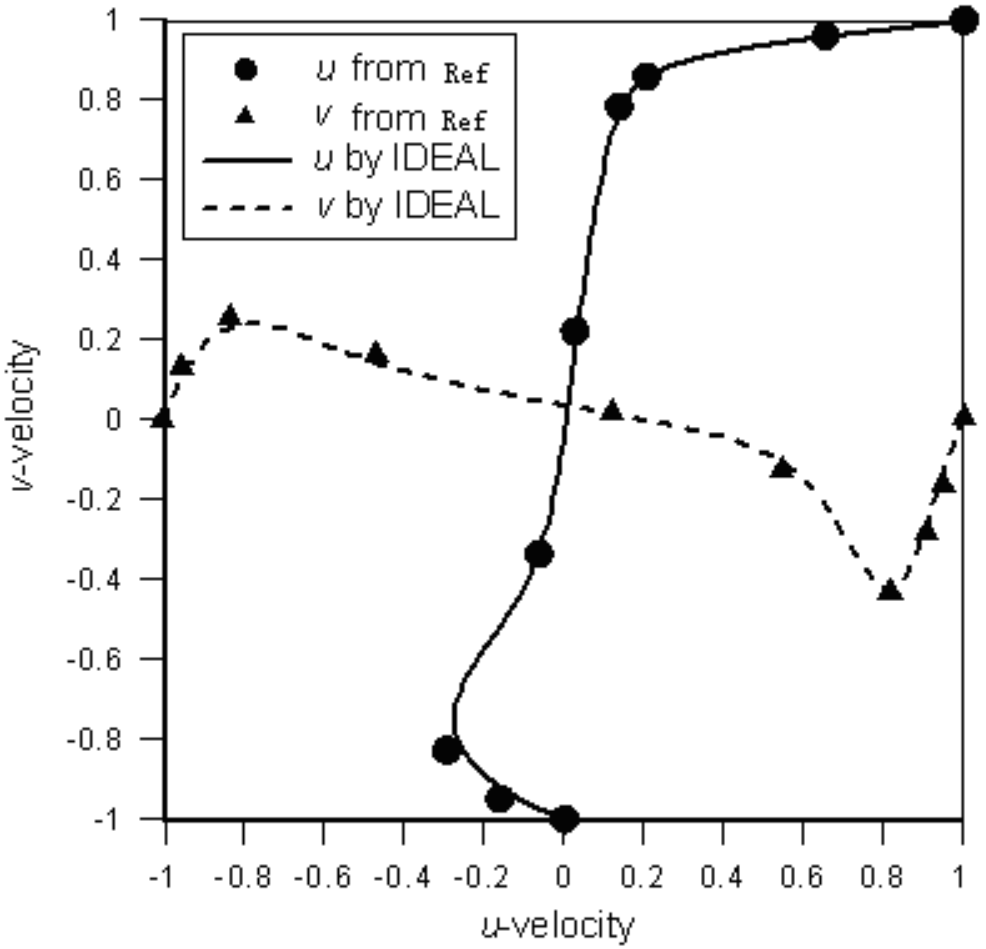
Problem 4: 2D natural convection in a square cavity with an internal isolated vertical plate



Problems 5: 3D lid-driven cavity flow

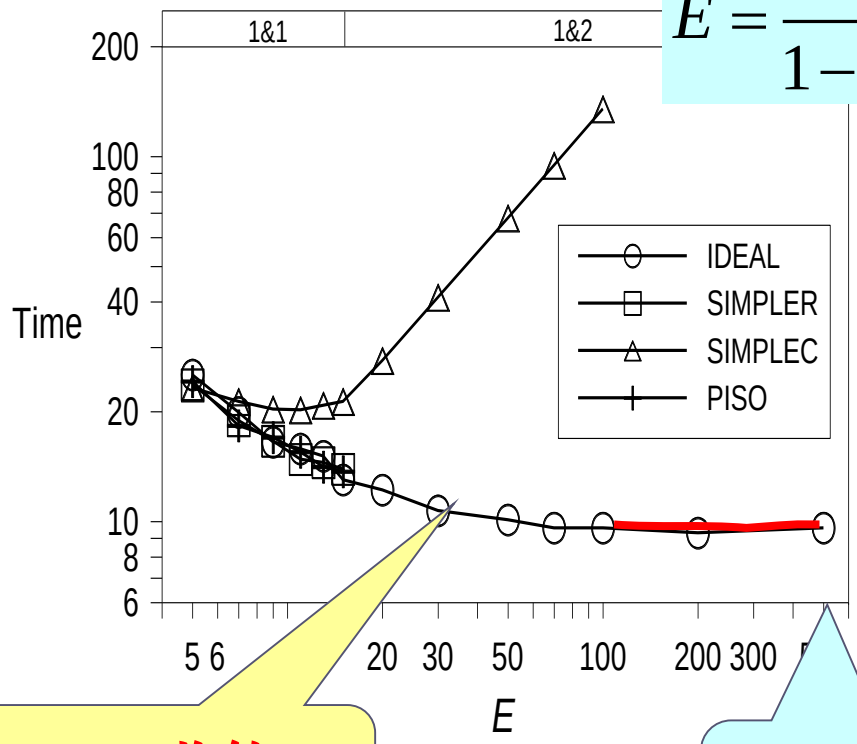


Flow configuration of lid-driven cavity flow in a cubic cavity

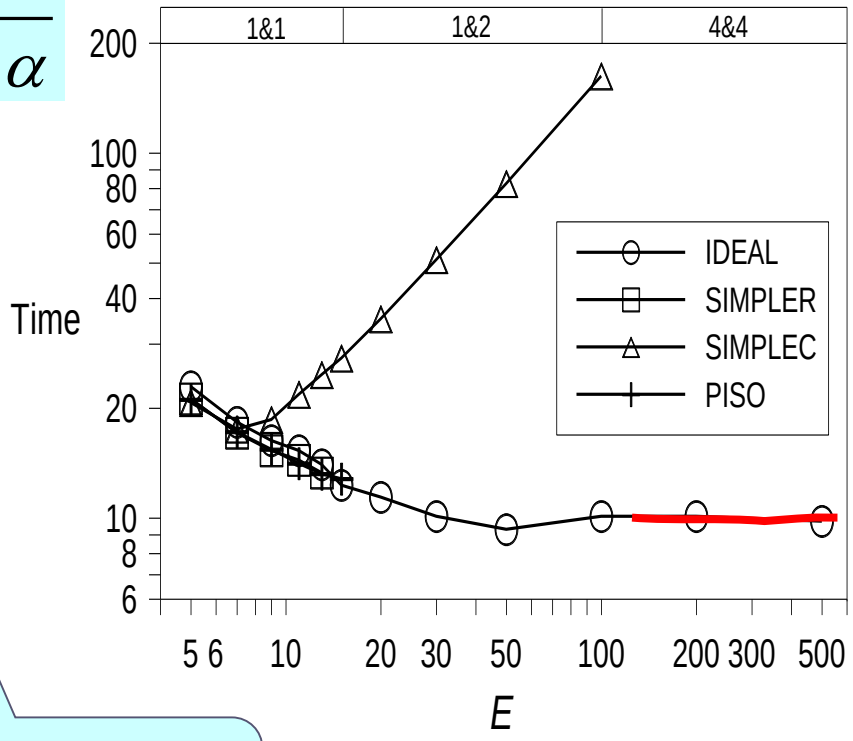


Comparison of velocity profiles *u* and *v* along the central axes on plane *z*=0.5*H* for *Re*=1000

$$E = \frac{\alpha}{1-\alpha}$$



(a) *Re*=100

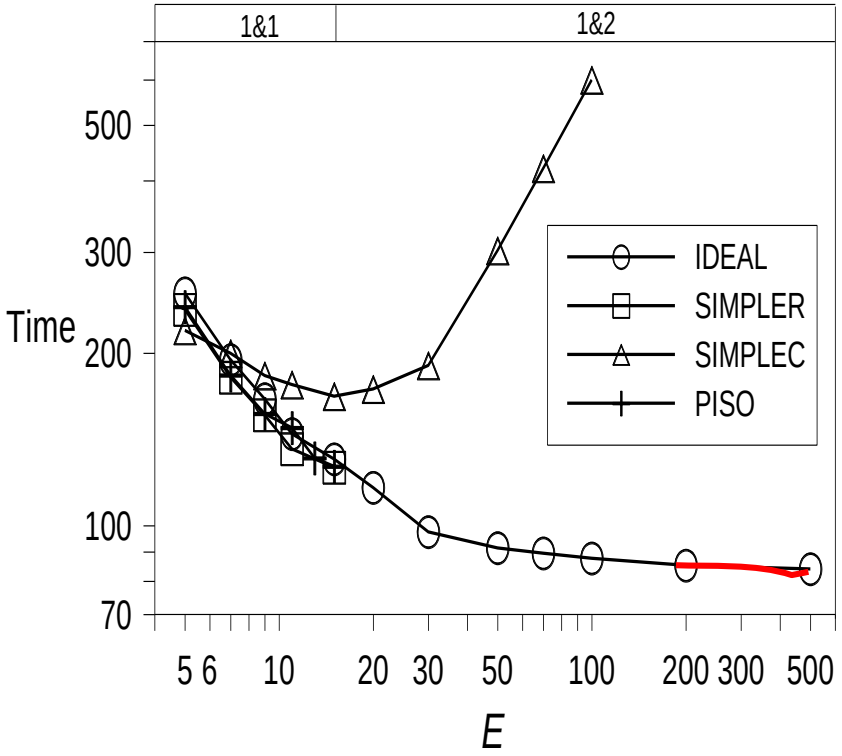


(b) *Re*=300

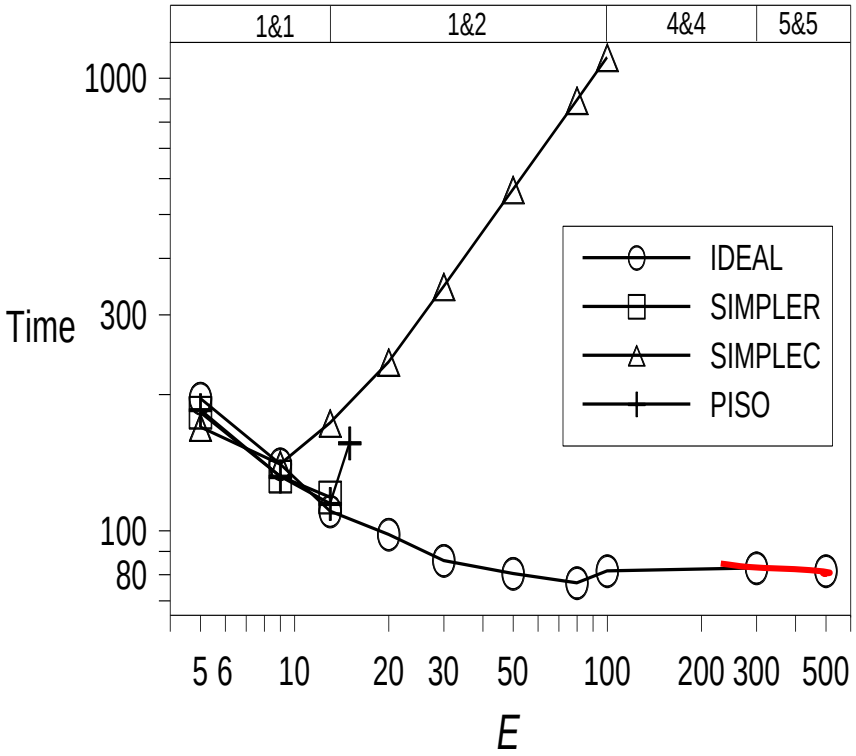
IDEAL 收敛特性曲线

亚松弛因子等于0.998

Comparison of computation time and robustness of IDEAL, SIMPLER, SIMPLEC and PISO algorithms for (a) *Re*=100 and (b) *Re*=300 with grid number=32×32×32

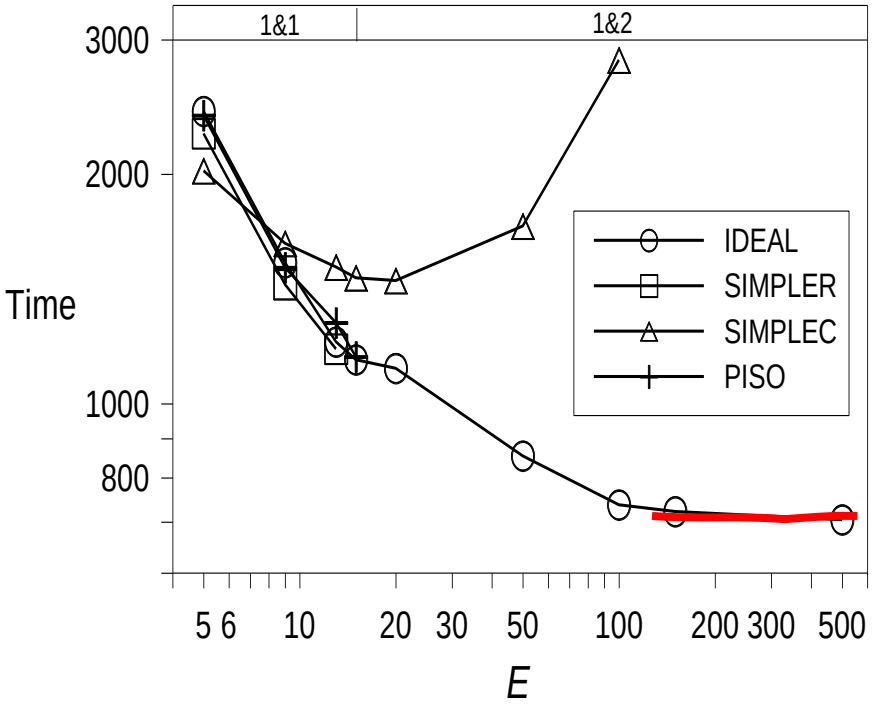


(a) *Re*=100

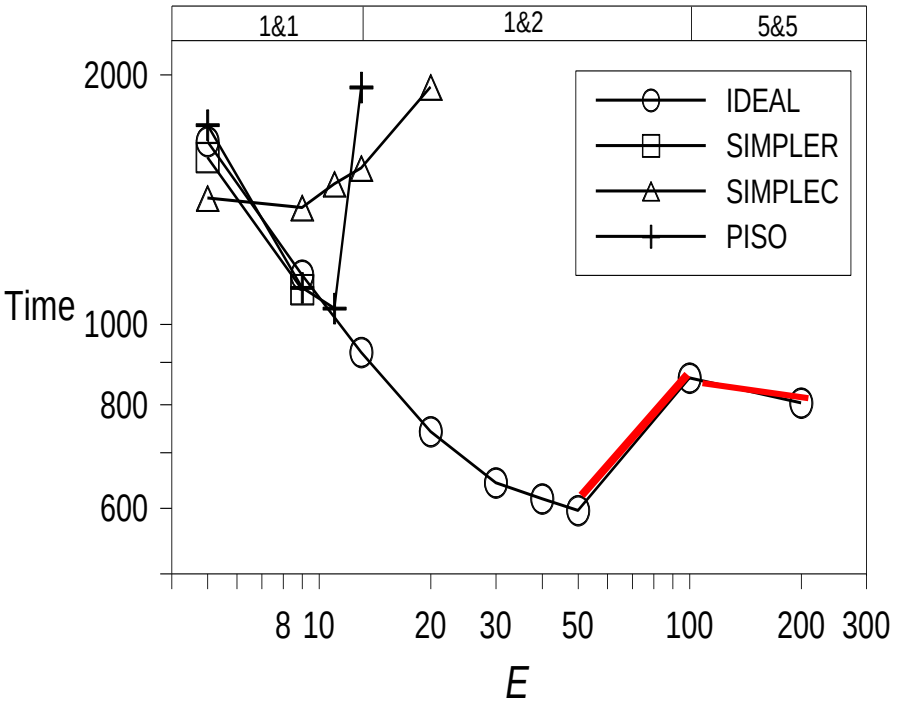


(b) *Re*=500

Comparison of computation time and robustness of an ideal algorithm, SIMPLER, SIMPLEC and PISO algorithms for (a) Re=100 and (b) Re=500 with grid number=52×52×52



(a) *Re*=100



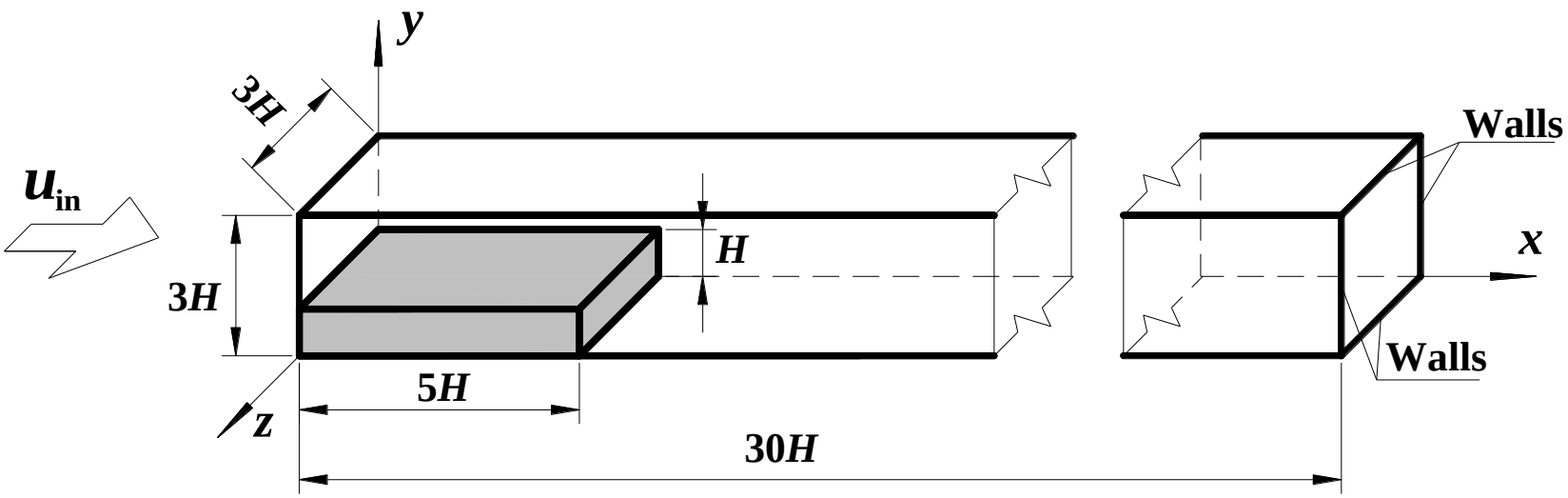
(b) *Re*=1000

Comparison of computation time and robustness of an ideal algorithm, SIMPLER, SIMPLEC and PISO algorithms for (a)Re=100 and (b)Re=1000 with grid number=82×82×82

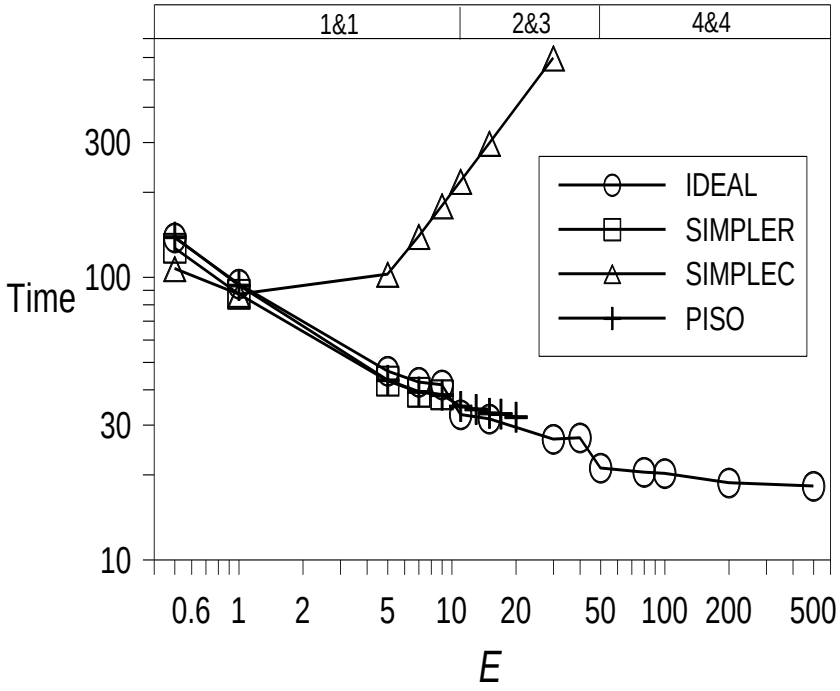
Reduced ratio of computation time of IDEAL algorithm over SIMPLER, SIMPLEC and PISO algorithms at their own optimal time step multiples

Grid number	32×32×32		52×52×52		82×82×82	
Re	100	300	100	500	100	1000
Reducing ratio over SIMPLER	33.1%	30.1%	33.5%	35.3%	40.3%	45.9%
Reducing ratio over SIMPLEC	54.0%	46.9%	50.0%	45.3%	51.4%	56.9%
Reducing ratio over PISO	32.1%	27.3%	33.8%	33.2%	38.8%	43.0%

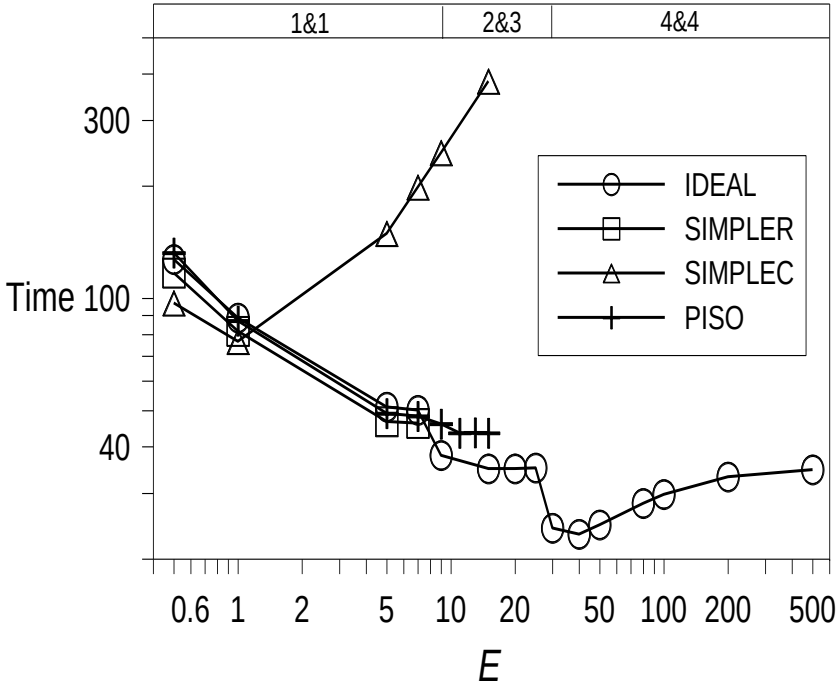
Problem 6: 3D flow over a back-ward step



Flow configuration of laminar fluid flow over a 3-D backward-facing step

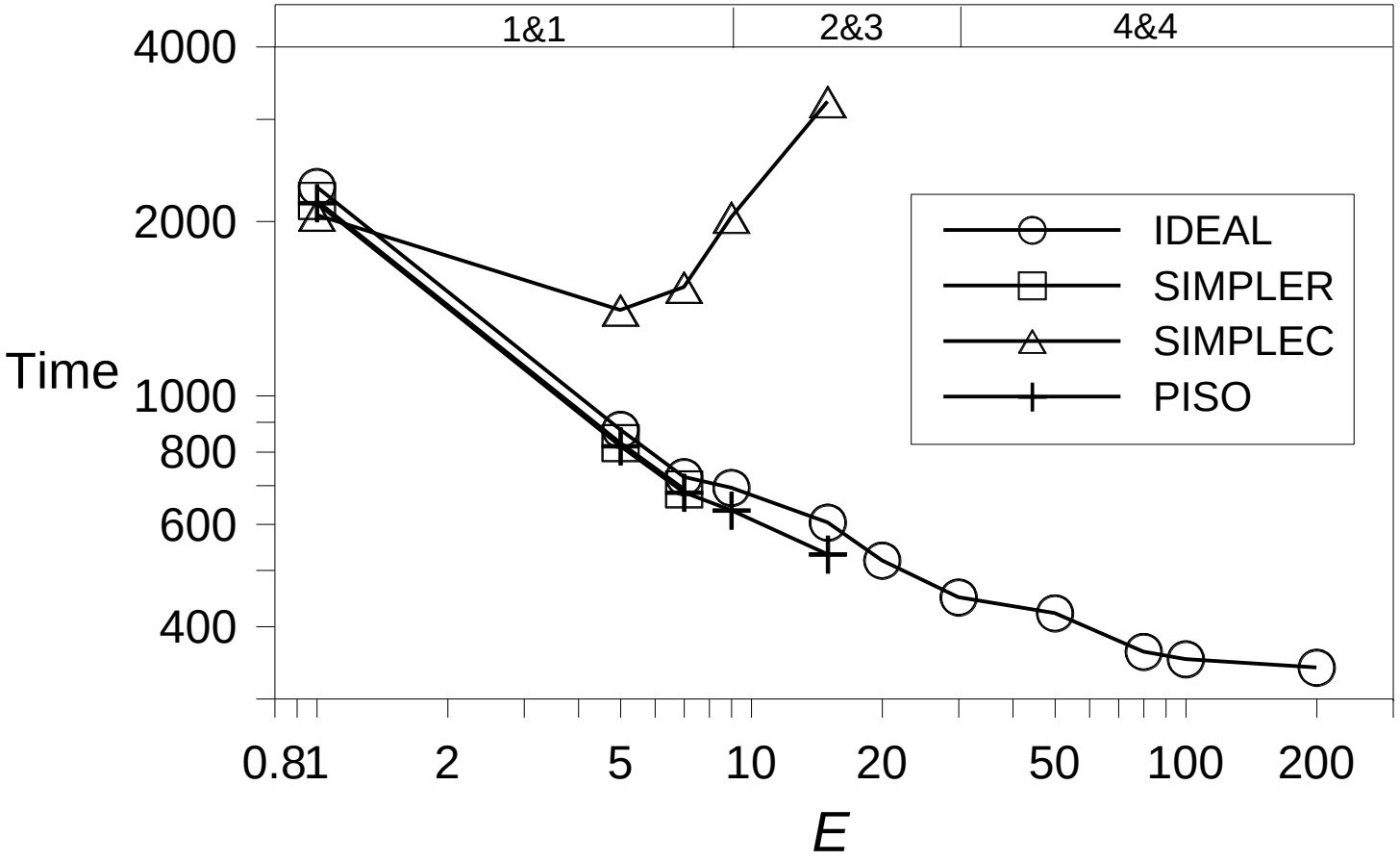


(a) *Re*=100



(b) *Re*=300

Comparison of computation time and robustness of IDEAL, SIMPLER, SIMPLEC and PISO algorithms for (a) Re=100 and (b) Re=300 with grid number=80×20×20

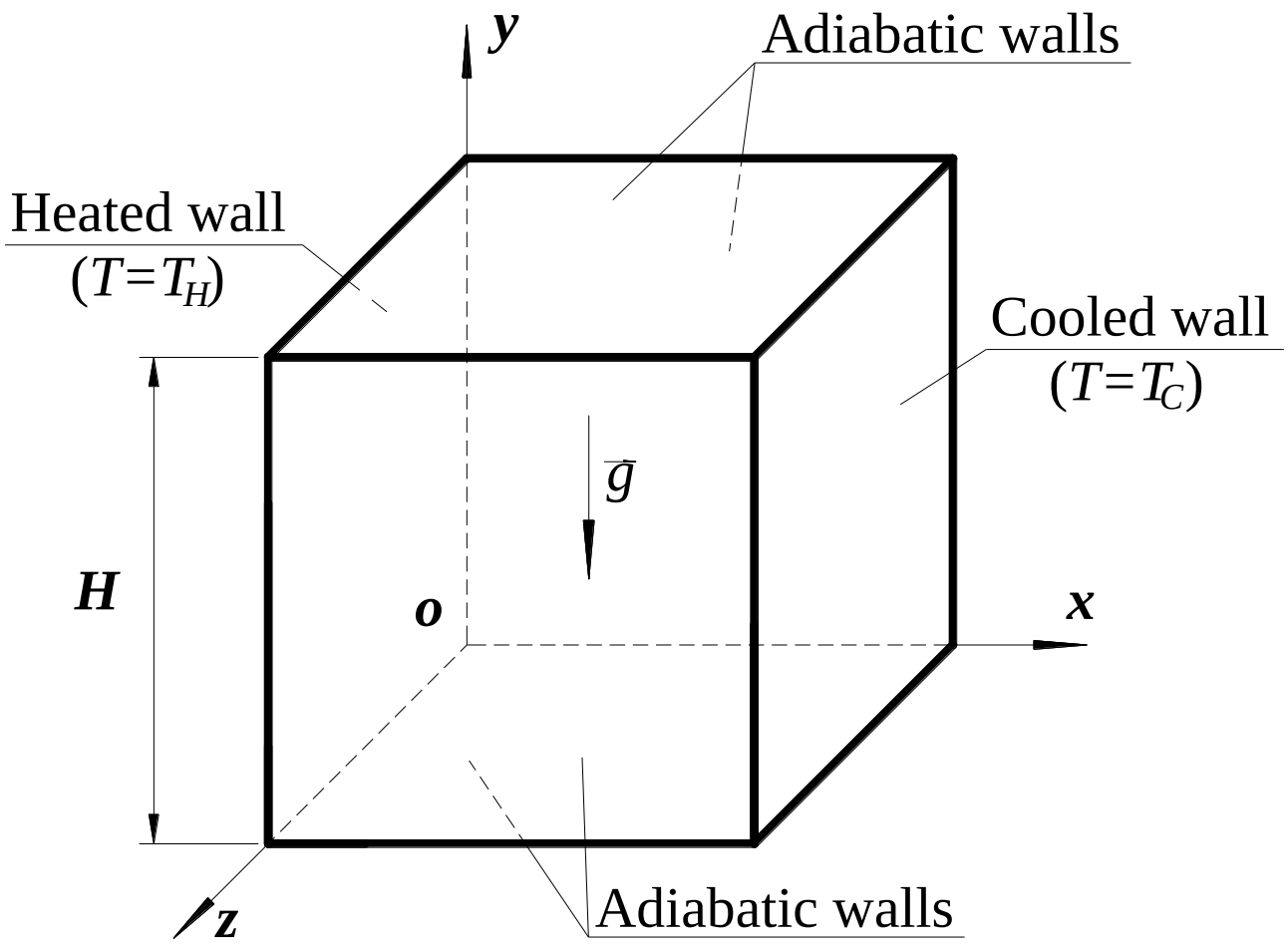


Comparison of computation time and robustness of IDEAL, SIMPLER, SIMPLEC and PISO algorithms for Re=100 with grid number=160×41×41

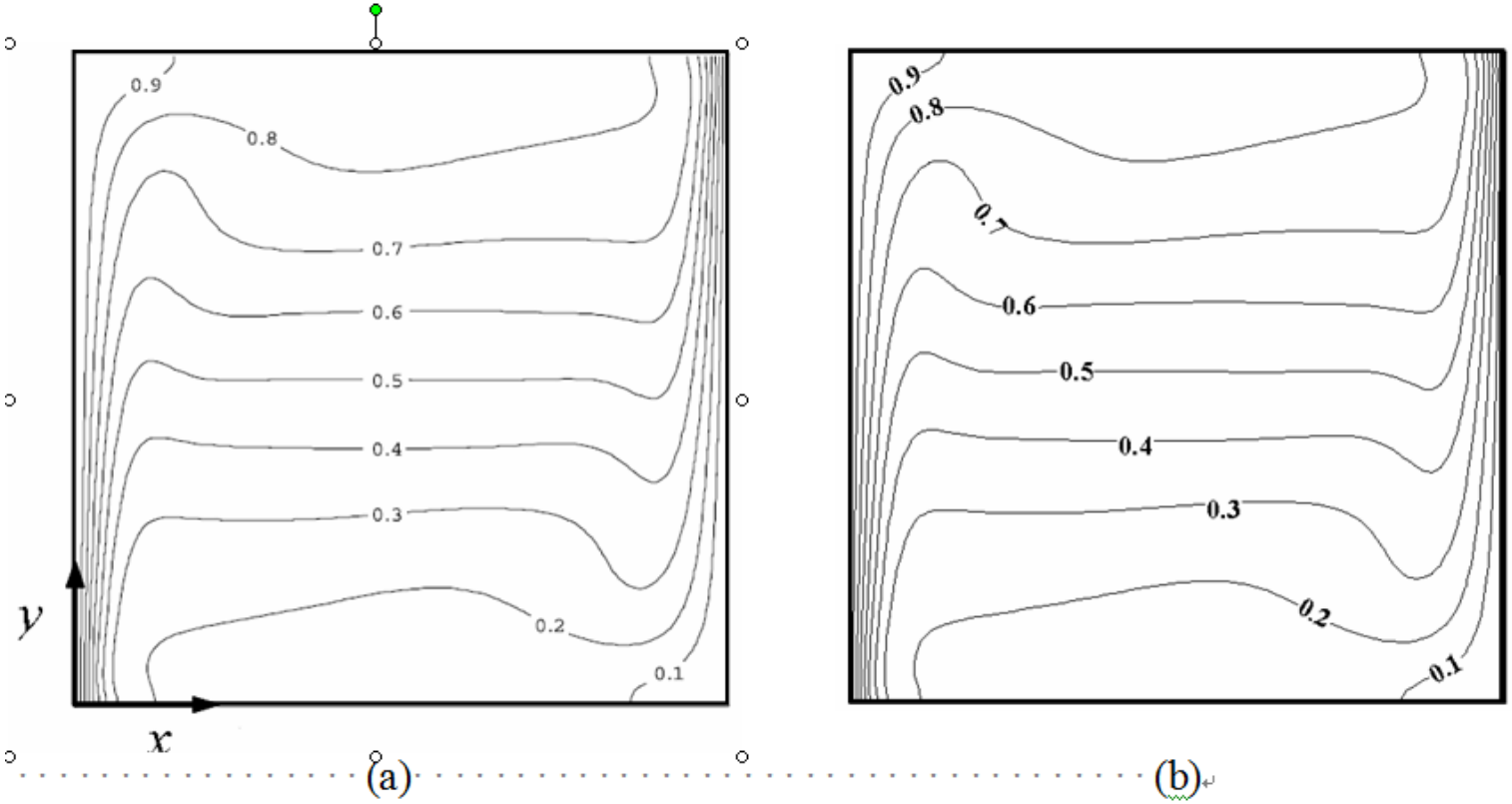
Reducing ratio of computation time of IDEAL algorithm over SIMPLER, SIMPLEC and PISO algorithms at their own optimal time step multiples, i.e., optimal underrelaxation factor

Grid number	80×20×20		160×41×41
Re	100	300	100
Reducing ratio over SIMPLER	52.6%	49.6%	50.8%
Reducing ratio over SIMPLEC	79.1%	69.6%	75.8%
Reducing ratio over PISO	43.1%	46.5%	36.3%

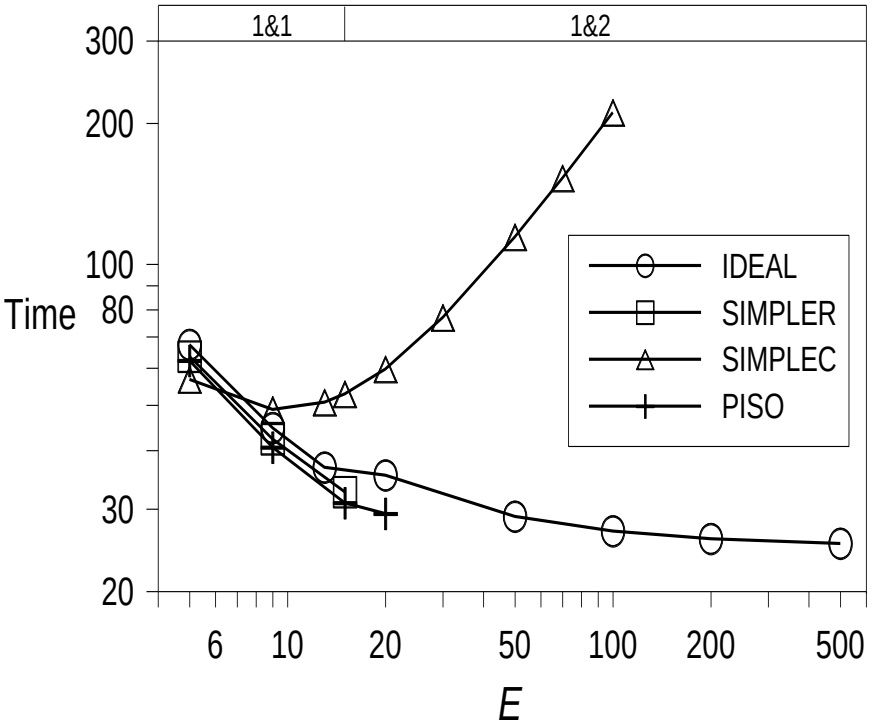
Problem 7: 3D natural convection in enclosure

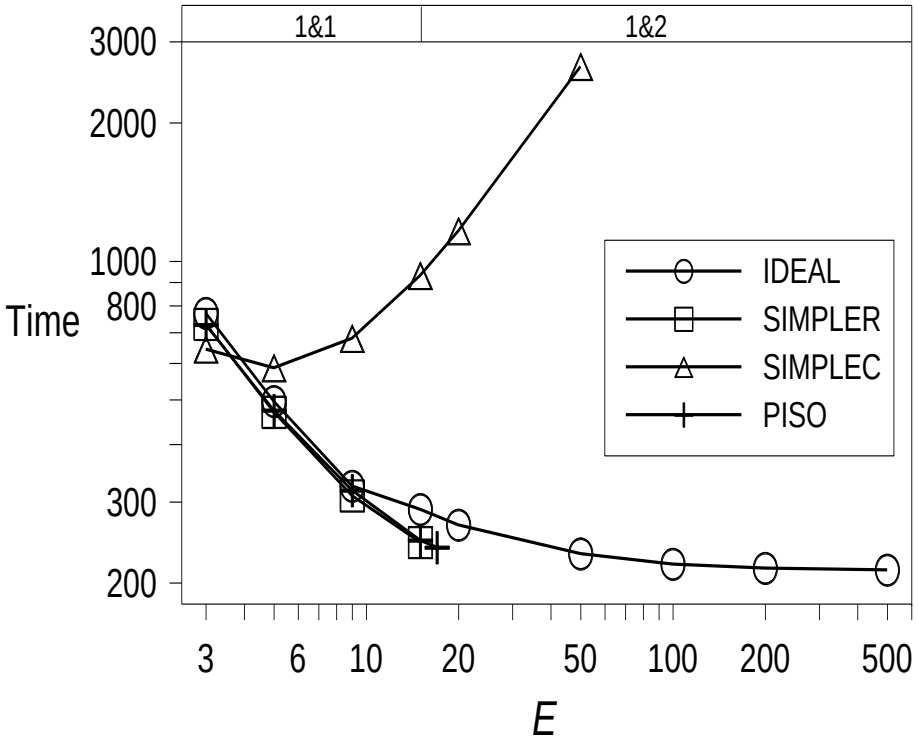


Flow configuration of natural convection in a cubic cavity.



Temperatures at the plane z=0.5H for Ra=10^6, obtained (a) from reference , (b) by IDEAL algorithm

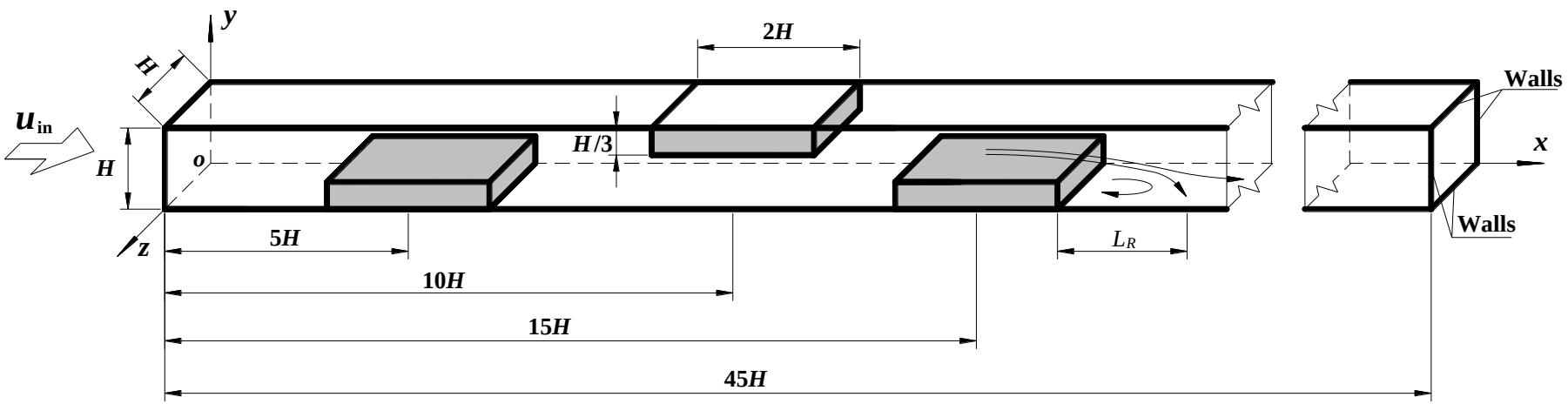




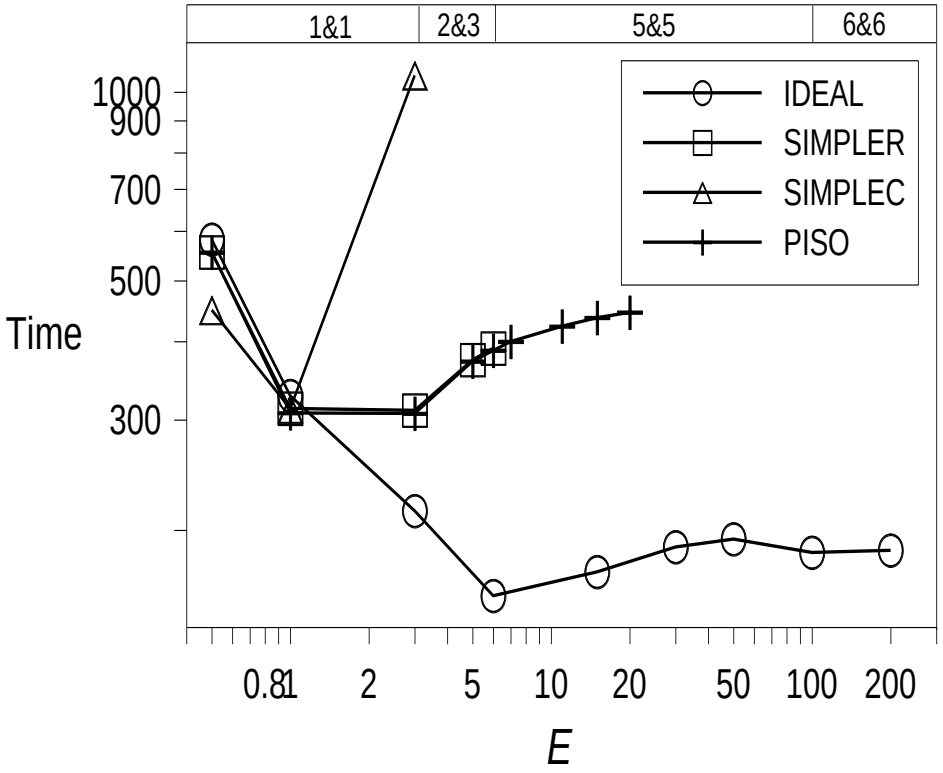
Comparison of computation time and robustness for $Ra=10^4$ with grid number=30×30×30

Comparison of computation time and robustness for $Ra=10^5$ with grid number=50×50×50

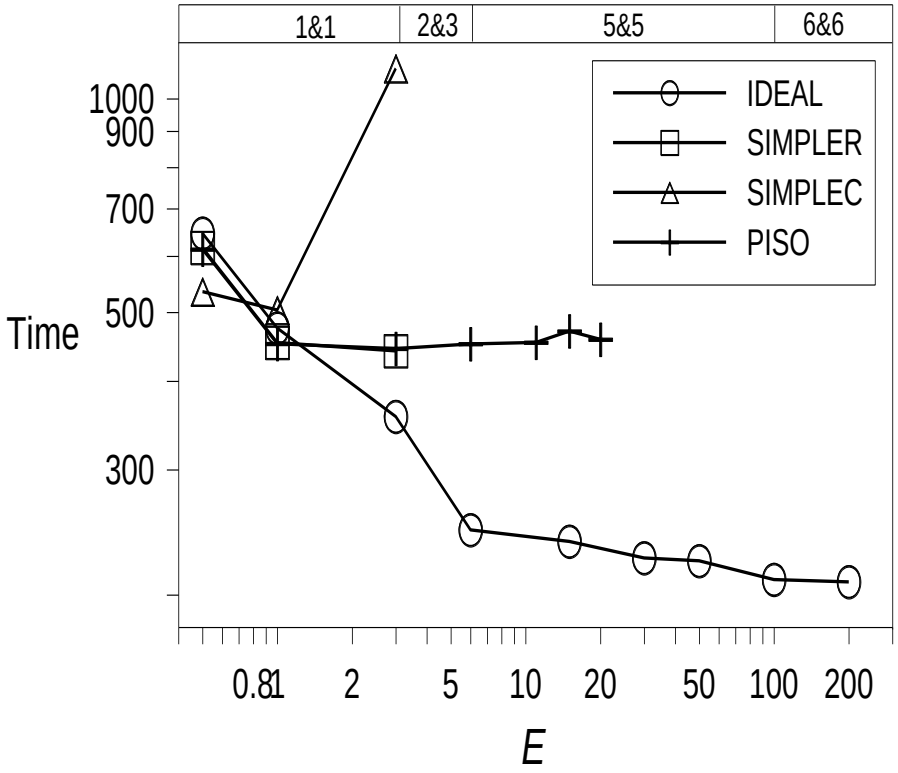
Problem 8: 3D flow over a back-ward step with complex structure



Flow configuration of laminar fluid flow through a 3-D duct with complicated structure

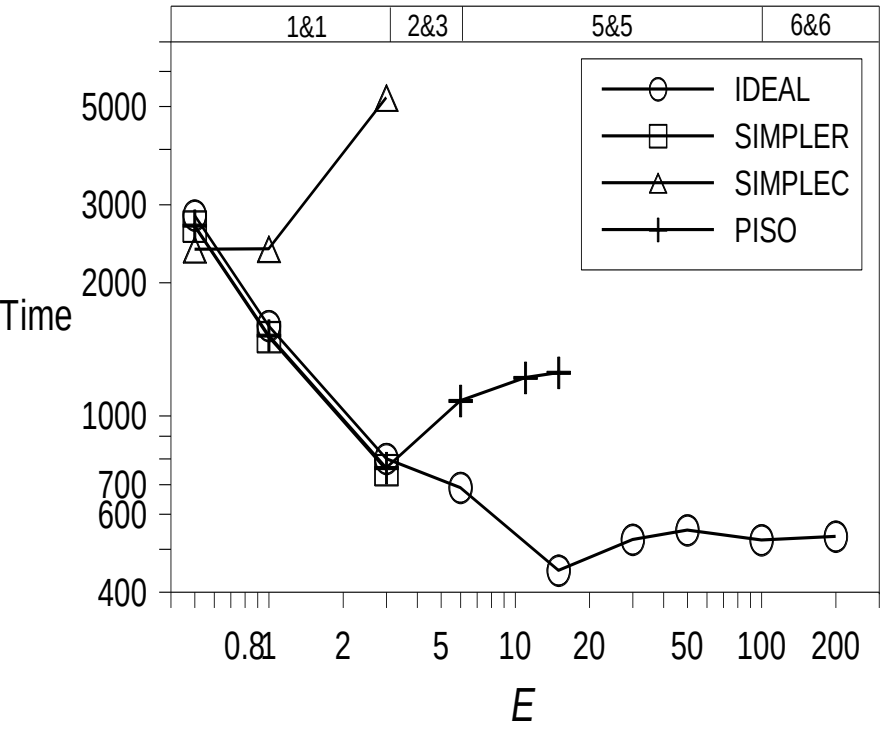


(a) *Re*=100

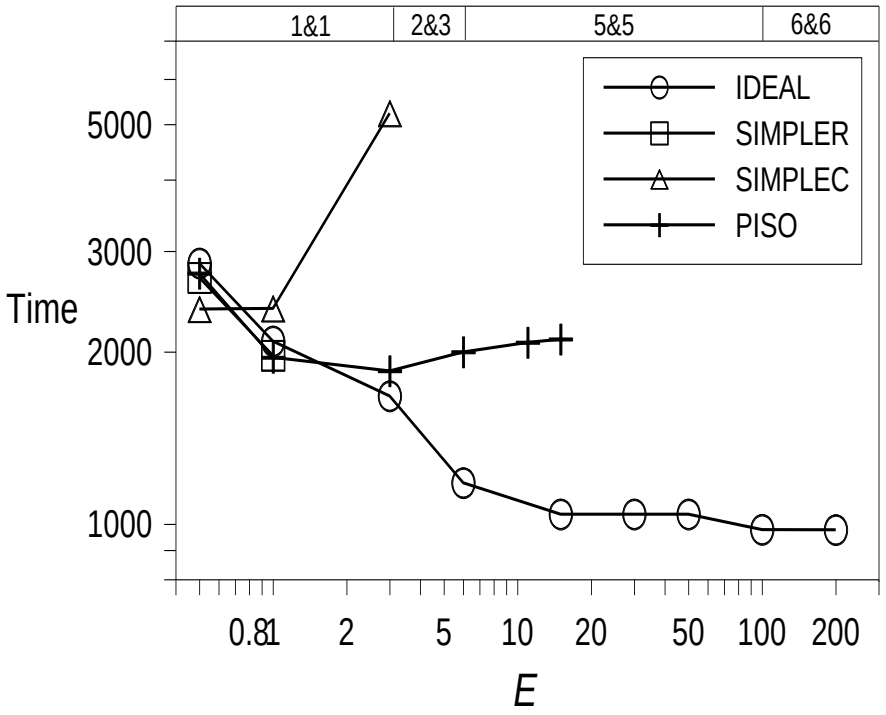


(b) *Re*=300

Comparison of computation time and robustness with grid number=150×20×20



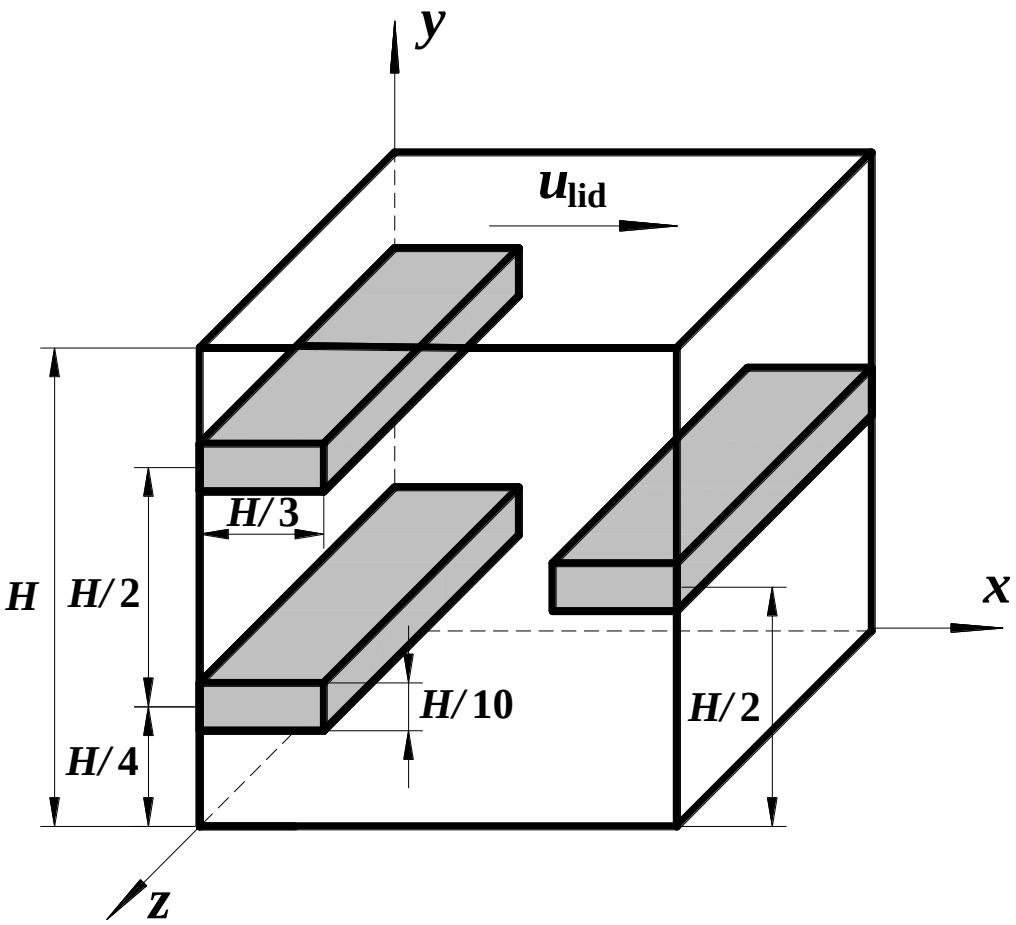
(a) *Re*=100



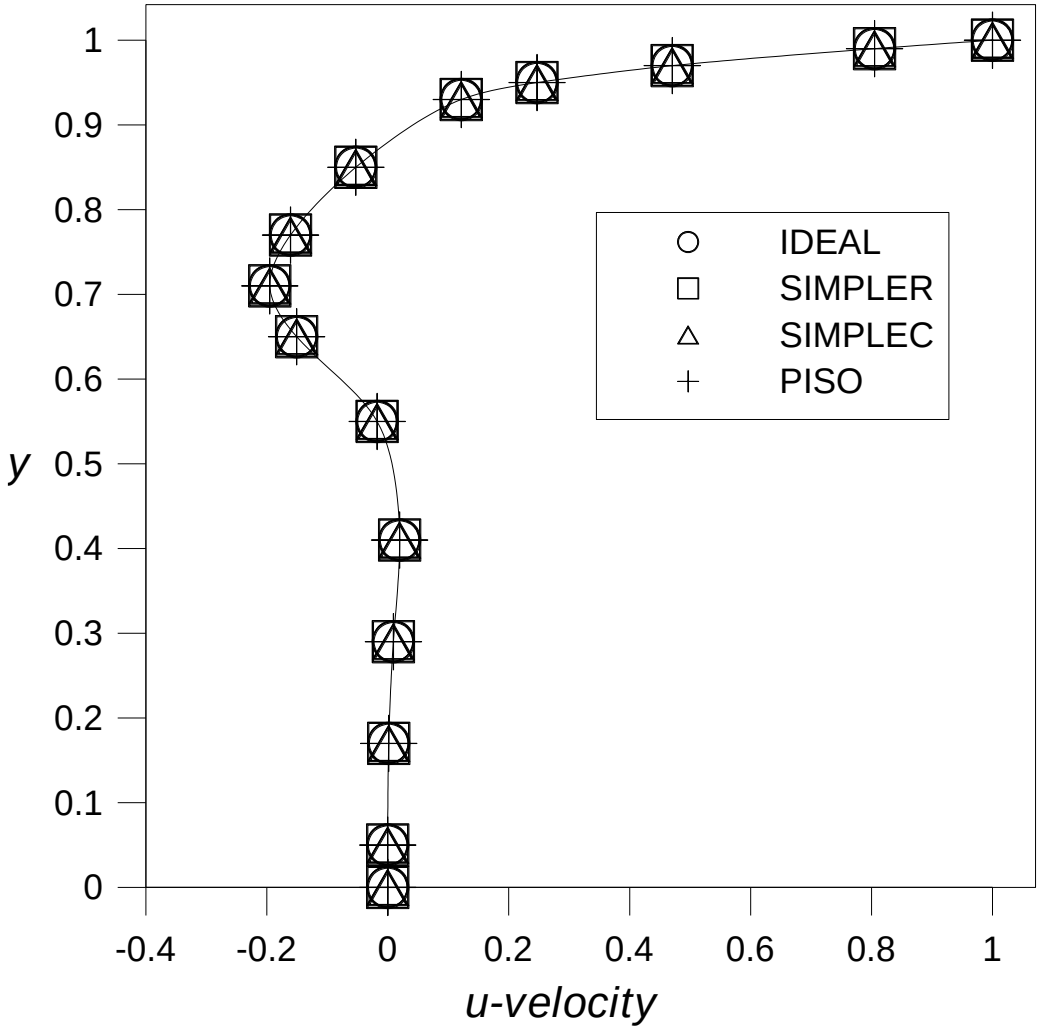
(b) *Re*=500

Comparison of computation time and robustness with grid number=190×29×29

Problem 9: 3D lid-driven flow in a complex cavity

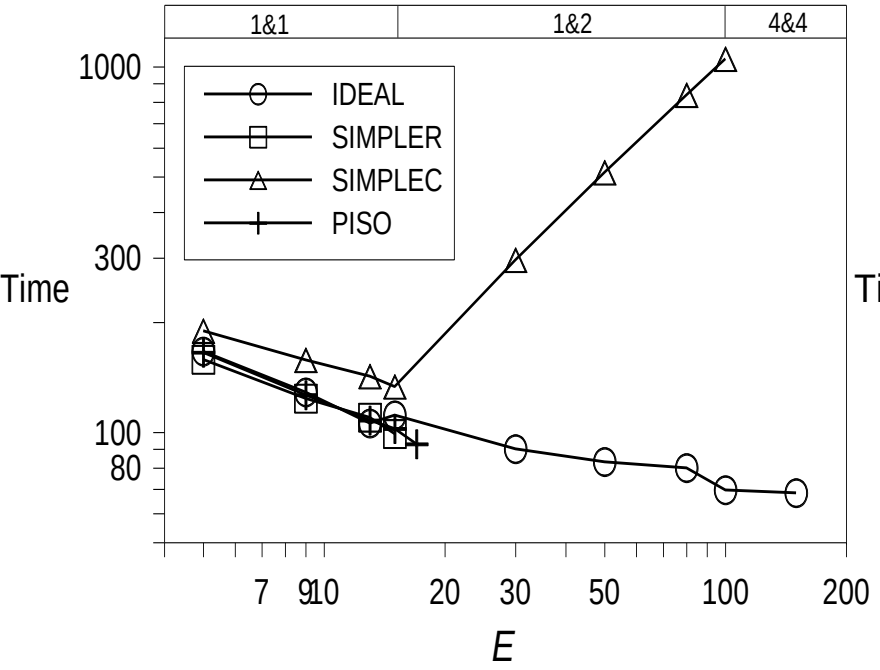


Flow configuration of lid-driven cavity flow in a cubic cavity with complicated structure

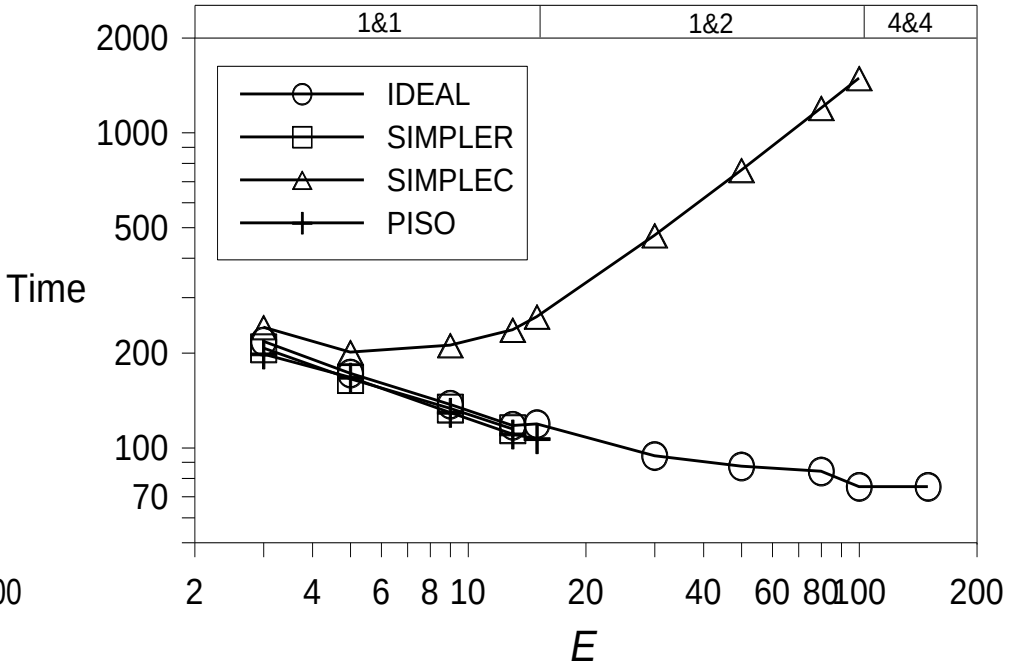


各种算法收敛时的结果应该相同---算法比健壮性；格式比数值解的精度。

Comparison of velocity profiles *u* along the central axes *y* on plane *z*=0.5*H* for *Re*=500.

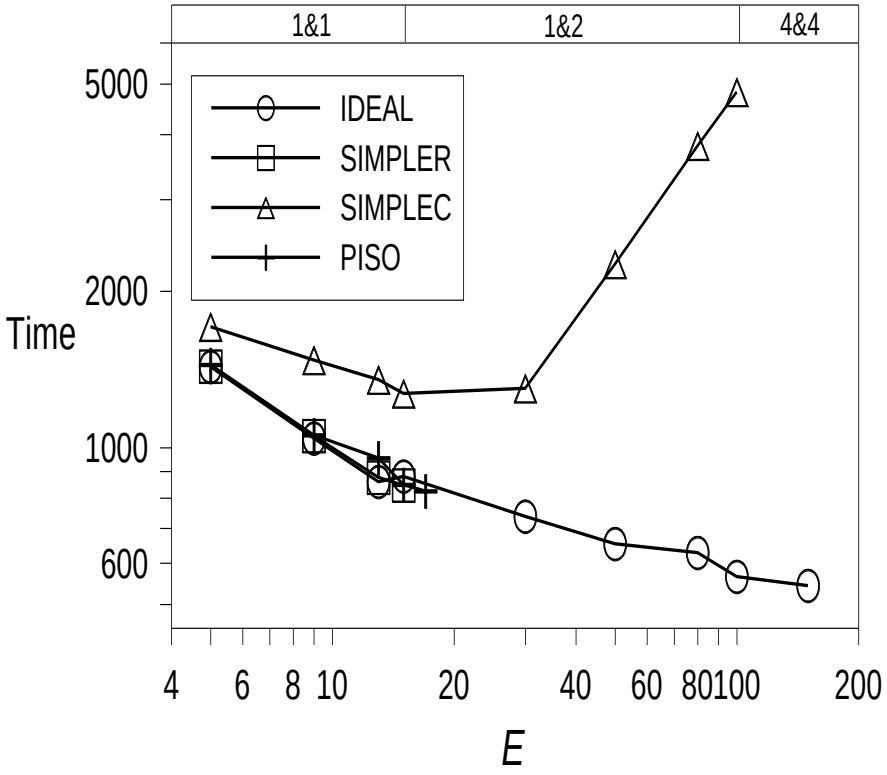


(a) *Re*=100

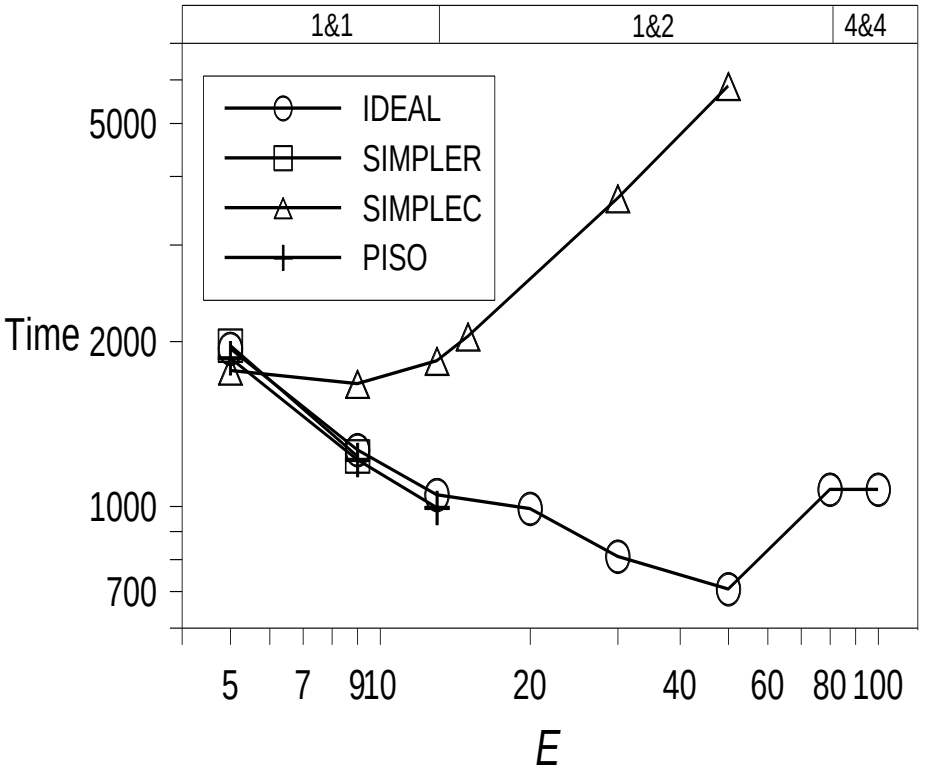


(b) *Re*=500

Comparison of computation time and robustness with grid number=52×52×52



(a) *Re*=100



(b) *Re*=800

Comparison of computation time and robustness with grid number=82×82×82

Extension of the IDEAL algorithm to 3D, body-fitted coordinates and compressible flows have been completed, and some improvements in convergence and robustness are made.

Sun DL, Qu Z G, He Y L, Tao WQ. An efficient segregated algorithm for incompressible fluid flow and heat transfer problems-IDEAL (Inner doubly iterative efficient algorithm for linked equation) Part I:Mathematical formulation and solution. **Numerical Heat Transfer, Part B,** 2008,53(1);1-17

推荐阅读(9)

Sun DL, Qu Z G, He Y L, Tao WQ. An efficient segregated algorithm for incompressible fluid flow and heat transfer problems-IDEAL (Inner doubly iterative efficient algorithm for linked equation) Part II:Application examples. **Numerical Heat Transfer, Part B,** 2008,53(1);18-38

Jin-Ping Wang, Jian-Fei Zhang, Zhi-Guo Qu & Wen-Quan Tao. Adaptive inner iteration processes in pressure based method for viscous compressible flows
Numerical Heat Transfer, Part B, 2018, 74(3): 603–622

第7 章 压力与速度耦合算法研究进展

7.6 非结构化网格上流场的求解

7.6.1 非结构化网格上求解流场的三个选择

7.6.2 积分形式的对流扩散方程及其离散

7.6.3 对流项的离散格式

7.6.4 扩散项和源项的离散

7.6.5 离散方程的最终形式

7.6.6 压力修正方程的导出

7.6.7 非结构化网格上SIMPLE算法的实施步骤

7.6 非结构化网格上流场的求解

7.6.1 非结构化网格上求解流场的三个选择

1. 作为求解变量速度的选择

直角坐标分量；逆变分量；协变分量

2. 节点位置选择：单元顶点；单元中心

3. 速度与压力的相对位置：交叉网格；同位网格

7.6.2 积分形式的对流扩散方程及其离散

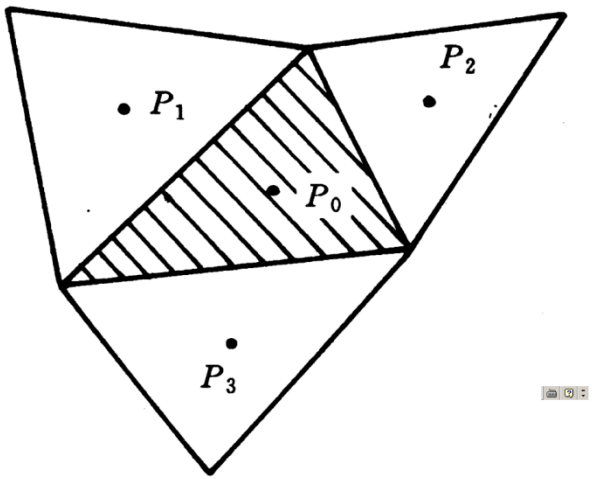
采用积分形式，将二阶导数降低到一阶导数,以减轻导数离散的难度：

对稳态问题，积分形式对流-扩散方程：

$$\int_A (\rho \vec{u}\phi - \Gamma_\phi \nabla \phi) \bullet d\vec{A} = \int_V S_\phi dV$$

将计算区域 V 用有限个三角形单元来离散，对其中任意一个三角形：

$$\sum_{j=1}^{3} \int_{A_j} (\rho \vec{u}\phi - \Gamma_\phi \nabla \phi) \bullet d\vec{A} = \int_V S_\phi dV$$



上式可简写为：

$$\sum_{j=1}^{3} (C_j + D_j) = \int_{V_{P0}} S_\phi dV$$

C_j, D_j 为界面上对流，扩散项的离散表达式。

7.6.3 对流项的离散格式

$$C_j = \int_{A_j} \rho\vec{u}\phi \bullet d\vec{A}_j \cong (\rho\vec{u}\phi)_j \bullet \vec{A}_j = (\rho\vec{u} \bullet \vec{A}_j)\phi_j = F_j\phi_j$$

F_j 为界面流量，流出为正，流入为负。两个问题：

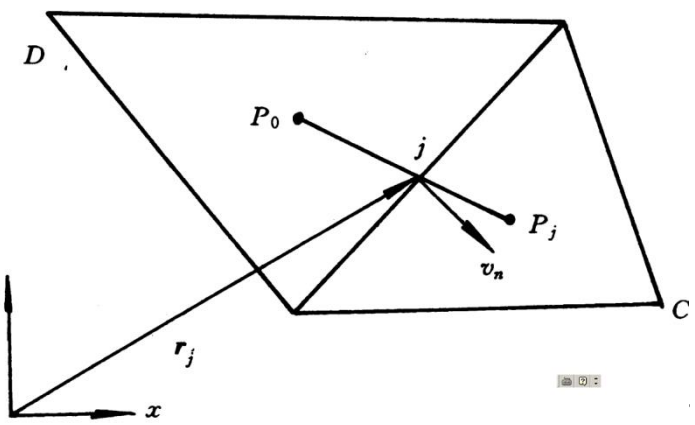
1）如何计算界面上的 ϕ_j ——格式问题；

2）如何计算界面流量 F_j ——同位网格特有的问题。

1. 界面上 ϕ_j 的计算

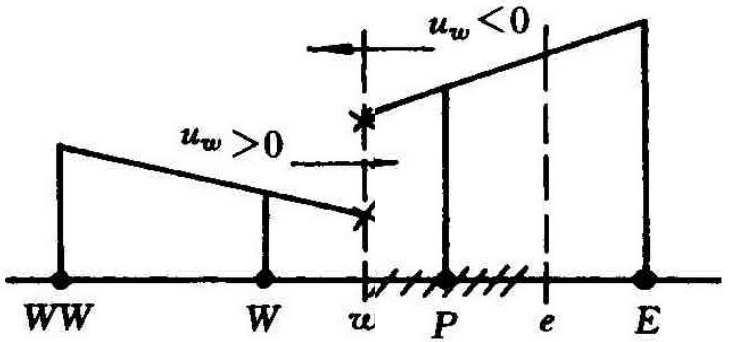
1）一阶迎风格式

$$\begin{cases} \phi_j = \phi_{P_0}, F_j > 0; \\ \phi_j = \phi_{P_j}, F_j < 0 \end{cases}$$



2）中心差分格式

ϕ_j 取为 ϕ_{P_j}与 ϕ_{P_0} 的算术平均值。



3）二阶迎风格式

在直角坐标系中，当u_w>0时有：

$$\phi_w = 1.5\phi_W - 0.5\phi_{WW} = \phi_W + 0.5(\phi_W - \phi_{WW}) =$$

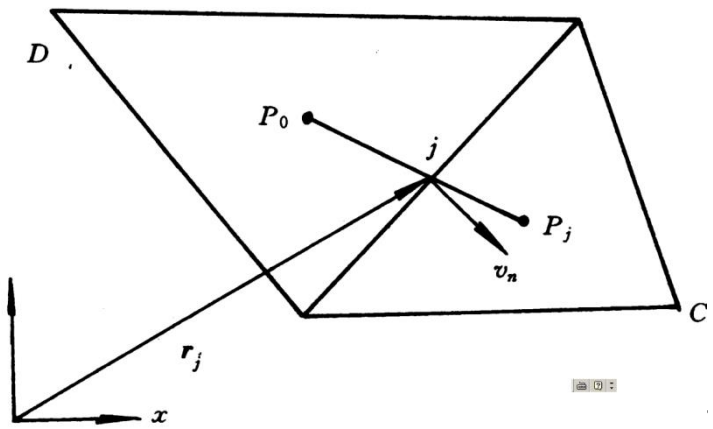
$$\phi_W + (\frac{\phi_W - \phi_{WW}}{\Delta x}) \times 0.5\Delta x$$

w界面与W节点间的距离

平均梯度

$$\phi_j = \phi_{P_0} + (\nabla\phi)_{P_0} \bullet (\vec{r}_j - \vec{r}_{P_0}), F_j > 0$$

$$\phi_j = \phi_{P_j} + (\nabla\phi)_{P_j} \bullet (\vec{r}_j - \vec{r}_{P_j}), F_j < 0$$

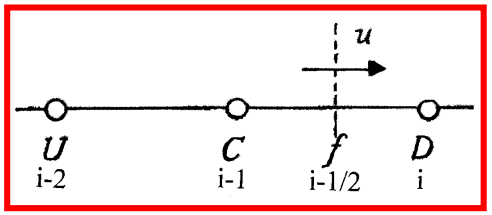


4）混合迎风格式

通过权因子 γ 将一阶与二阶迎风组合起来：

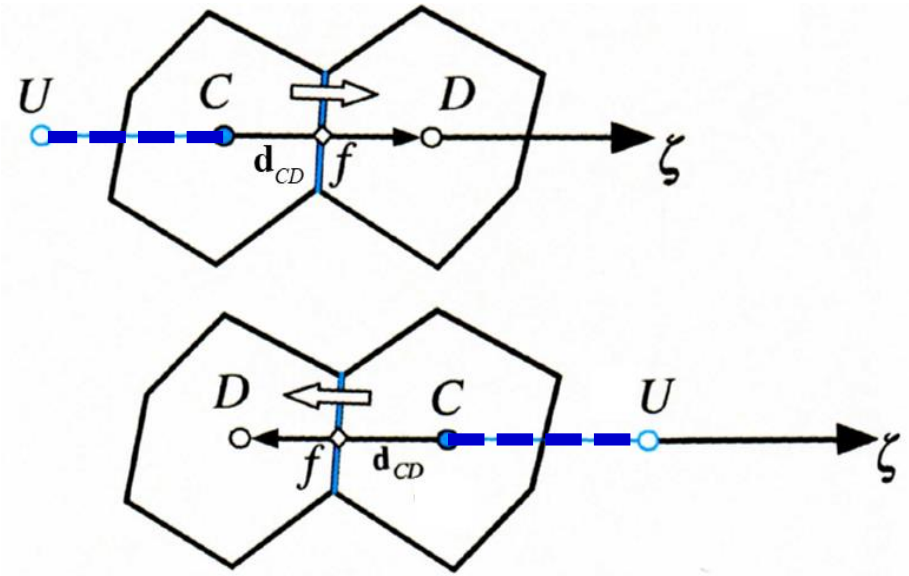
$$\phi_j = \begin{cases} \phi_{P_0} + \gamma(\nabla\phi)_{P_0} \bullet (\vec{r}_j - \vec{r}_{P_0}), F_j > 0 \\ \phi_{P_j} + \gamma(\nabla\phi)_{P_j} \bullet (\vec{r}_j - \vec{r}_{P_j}), F_j < 0 \end{cases} \quad (0 \le \gamma \le 1)$$

对非结构化网格，通过构筑一个虚拟的上游点 **U** 就可以采用第6章中的各种定义：



$$\phi_D - \phi_U = \nabla\phi_C \bullet \mathbf{d}_{UD}$$

$$\phi_U = \phi_D - 2\nabla\phi_C \bullet \mathbf{d}_{CD}$$



2. 如何计算 $(\nabla\phi)_{P_0}$

要确定计算单元中的 $(\nabla\phi)_{P_0}$ 就是要计算直角坐标系中的分量 $\frac{\partial\phi}{\partial x},\frac{\partial\phi}{\partial y}$ 的离散表达式。**一种简单的方法**：

1）将一阶导数的离散表达式看成是一阶导数在该单元中的平均值 (**FVM的基本思想**）：

$$\left(\frac{\partial\phi}{\partial x_i}\right)_{P_0}\cong\frac{\int_V\left(\frac{\partial\phi}{\partial x_i}\right)dV}{V_{P_0}}$$

$$(2D: i=1,2)$$

$$div(\vec{\phi})=\frac{\partial\phi_x}{\partial x}+\frac{\partial\phi_y}{\partial y}$$

其中 ϕ_x,ϕ_y 为 $\vec{\phi}$ 在两轴上的分量。

如果 $\vec{\phi}=\phi\vec{i}+0\bullet\vec{j}$ 则

2）矢量 $\vec{\phi}$ 的散度为：

$$div(\vec{\phi})=\frac{\partial\phi}{\partial x}$$

因此 $\frac{\partial \phi}{\partial x}$ 可以看成是矢量 $\vec{\phi} = \phi\vec{i}$ 的散度；

3）利用上述结果和Gauss降维定律，导数计算式中的体积分可以进一步简化：

Gauss降维定律

$$\int_V \left(\frac{\partial \phi}{\partial x}\right) dV = \oint_A (\phi\vec{i}) \bullet d\vec{A} = \oint_A \phi(\vec{i} \bullet \vec{n} dA) =$$

看成是 $\phi\vec{i}$ 的散度

矢量本身沿周界的积分

$$\sum_{j=1}^{N} \phi_j (\vec{i} \bullet \vec{n}_j A_j) = \sum_{j=1}^{N} \phi_j A_j^x$$

j 是界面的编号

A_j 界面在i方向的投影

j 界面在x方向的投影

故有 $$\int_V (\frac{\partial \phi}{\partial x}) dV = \sum_{j=1}^{3} \phi_j A_j^x \ (j = 1,2,3)$$ （x方向）

类似地 $\frac{\partial \phi}{\partial y}$ 可以看成是矢量 $\vec{\phi} = \phi \vec{j}$ 的散度；

$$\int_V (\frac{\partial \phi}{\partial y}) dV = \sum_{j=1}^{3} \phi_j A_j^y \ (j = 1,2,3)$$

梯度的两个分量为：

$$(\frac{\partial \phi}{\partial x_i})_{P_0} \cong \frac{\int_V (\frac{\partial \phi}{\partial x_i}) dV}{V_{P_0}} = \frac{\sum_{j=1}^{3} \phi_j A_j^i}{V_{P_0}} \qquad (i = 1,2)$$

3. 如何计算界面流速

这是保证不出现波形压力场的关键步骤。回顾结构化网格的同位网格上的处理方式：

1. 交叉网格成功的经验－引入1－δ 压差及 O(δ^2)

1－ δ 压差可有效地克服不合理的波形压力分布；

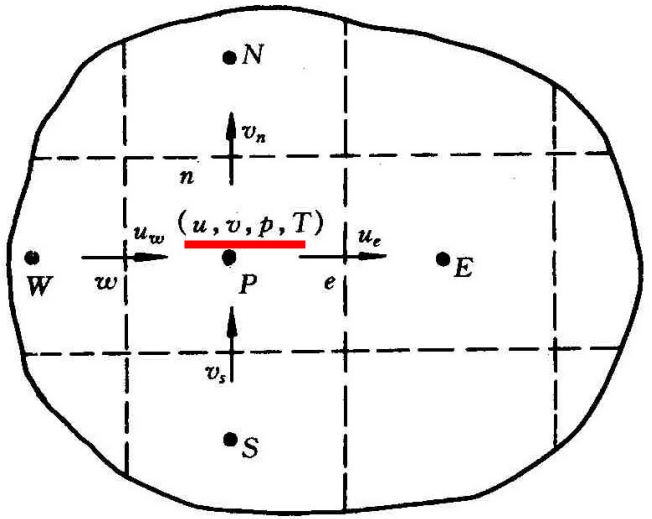
O (δ^2)保证压力Poisson方程的扩散特性。

2 . 在结构化的非交叉网格上控制方程的离散

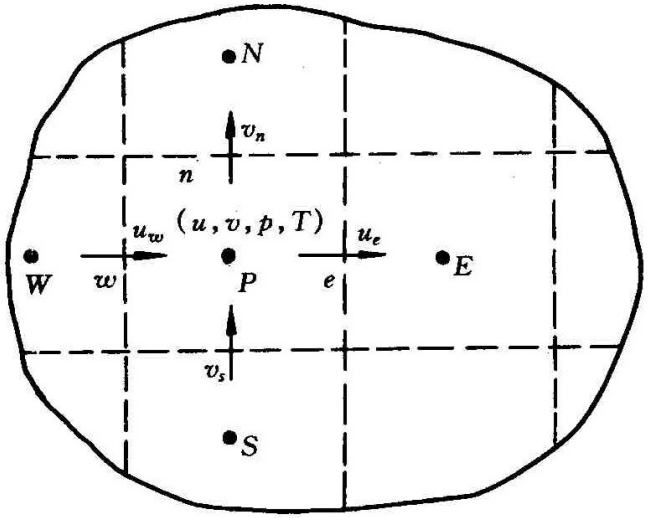
1) 动量守恒

$$a_P u_P = \sum a_{nb} u_{nb} + b + A_P (p_w - p_e)$$

$$u_P = \frac{\sum a_{nb} u_{nb} + b}{a_P} + \frac{A_P}{a_P}(p_w - p_e)$$



$$u_P = u_P - \frac{A_P}{a_P}(p_e - p_w)$$



类似地

$$u_E = u_E - (\frac{A_P}{a_P})_E (p_e - p_w)_E$$

对E点写出其东、西界面压差

2) 质量守恒

对主节点控制容积写出，需要用到未知的界面流速：

$$(\rho uA)_e - (\rho uA)_w + (\rho vA)_n - (\rho vA)_s = 0$$

3) 界面流速的确定

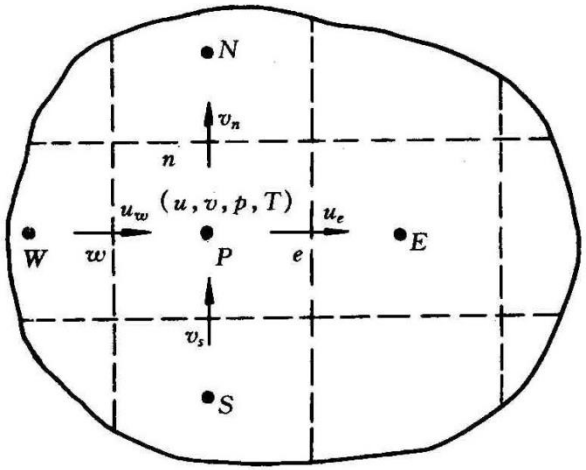
界面流速的确定为引入 $1-\delta$ 压差提供了途径；模仿 u_P, u_E 的计算式：

$$u_E = u_E - \left(\frac{A_P}{a_P}\right)_E (p_e - p_w)_E$$

类似地，试将界面流速表示为：

$$u_e = u_e - \left(\frac{A_P}{a_P}\right)_e (p_E - p_P)$$

E界面的东、西邻点压差



上式也是界面流速的动量方程。

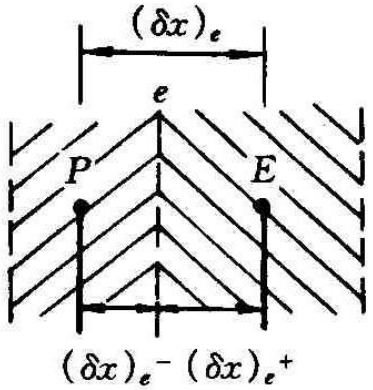
如果界面流速表示为 u_P, u_E 间的线性插值，则就失去了机会。Rhie-Chow引入了这种插值,称为**动量插值**(momentum interpolation method-MIM)

结构化的同位网格中MIM网格界面上 $u_e,(\frac{A_P}{a_P})_E$ 均采用线性插值计算公式

线性插值

$$u_e = u_P \frac{(\delta x)_{e^+}}{(\delta x)_e} + u_E \frac{(\delta x)_{e^-}}{(\delta x)_e}$$

$$(\frac{A_P}{a_P})_e = (\frac{A_P}{a_P})_P \frac{(\delta x)_{e^+}}{(\delta x)_e} + (\frac{A_P}{a_P})_E \frac{(\delta x)_{e^-}}{(\delta x)_e}$$



参考结构化网格上界面流速计算的基本思想——引入相邻两点间的压力差；但非结构化网格上几何关系复杂，不宜采用界面的假拟速度这样的参量。

目前文献中采用的一种确定界面流速的方法系参考结构化网格确定界面流速的下列思想：

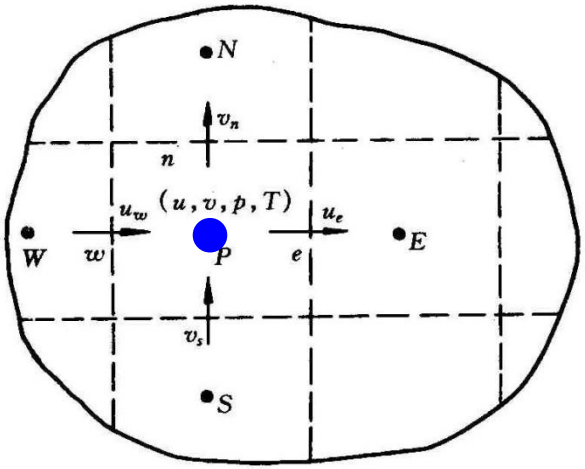
由结构化同位网格界面流速：

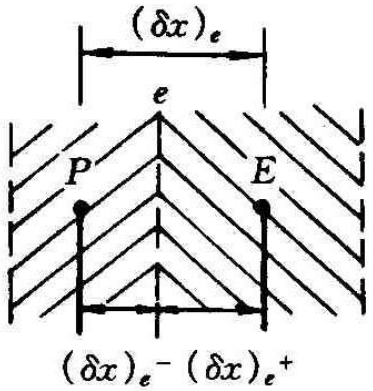
据 $u_E = u_E - (\frac{A_P}{a_P})_E (p_e - p_w)_E$

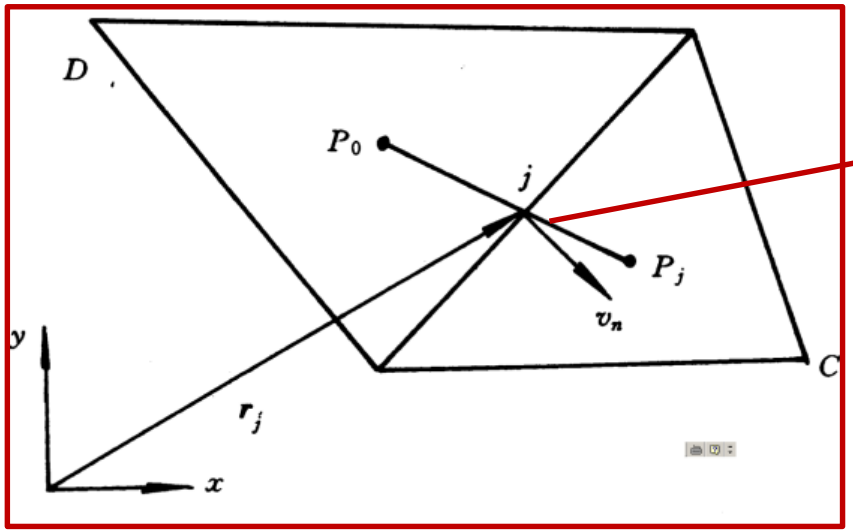
将界面流速表示为：

$$u_e = u_e - (\frac{A_P}{a_P})_e (p_E - p_P)$$

$$(\frac{A_P}{a_P})_e = (\frac{A_P}{a_P})_P \frac{(\delta x)_{e^+}}{(\delta x)_e} + (\frac{A_P}{a_P})_E \frac{(\delta x)_{e^-}}{(\delta x)_e}$$







$\vec{d}_j$

在 $\overrightarrow{A_j}$ 方向上的一个速度小量(包含相邻2点压差)

$$\overrightarrow{u_j}=\overrightarrow{u_{P_0 j}}+K_i\left\{\frac{1}{2}[(\nabla p)_{P_0}\bullet\frac{\overrightarrow{d_j}}{|\overrightarrow{d_j}|}+(\nabla p)_{P_j}\bullet\frac{\overrightarrow{d_j}}{|\overrightarrow{d_j}|}]-\frac{p_{P_j}-p_{P_0}}{|\overrightarrow{d_j}|}\right\}\frac{\overrightarrow{A_j}}{|\overrightarrow{A_j}|}$$

$\overrightarrow{u_{P_0}},\overrightarrow{u_{P_j}}$ 之间的线性插值。

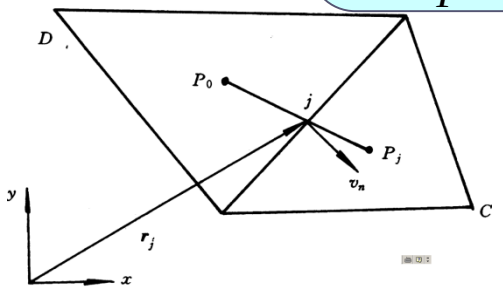
压差项前的系数，相当于：$(\frac{A_P}{a_P})_e$

d_j 方向单位矢量

P_0,P_j 单元的平均压力梯度在 d_j 方向的投影，其值与后者十分接近；在结构化均分网格中将不显含相邻两点间的压差。

保证相邻两点间的压差出现在界面流速中的附加项，以衰减波形压力场。

单位向量



类似于结构化网格上同位网格的动量插值：

$$(\frac{A_P}{a_P})_e=(\frac{A_P}{a_P})_P\frac{(\delta x)_{e^+}}{(\delta x)_e}+(\frac{A_P}{a_P})_E\frac{(\delta x)_{e^-}}{(\delta x)_e}\xrightarrow{\text{均分}}\frac{1}{2}[(\frac{A_P}{a_P})_P+(\frac{A_P}{a_P})_E]$$

在非结构化网格上该系数为

$$K_j=\frac{1}{2}[(\frac{V}{a_0^u})_{P_0}+(\frac{V}{a_0^u})_{P_j}]$$

为抵消压力梯度中的距离，由面积改为体积。

在结构化网格上

$$u_e=u_e-(\frac{A_P}{a_P})_e(p_E-p_P)$$

在非结构化网格上从量纲来看

$$\vec{u}_j=\vec{u}_{P_{0j}}+K_j\left\{C-\frac{p_{P_j}-p_{P_0}}{|\vec{d}_j|}\right\}\frac{\vec{A}_j}{|\vec{A}_j|}$$

分母中多了一个长度

7.6.4 扩散项和源项的离散

在非结构化网格中扩散项的离散也较复杂。

$$D_j = \int (-\Gamma_\phi \nabla \phi) \bullet d\vec{A} = -\int \Gamma_\phi \nabla \phi \bullet d\vec{A}$$

其其物理意义是通过界面*j*的扩散传递。将总传递分解为两部分：

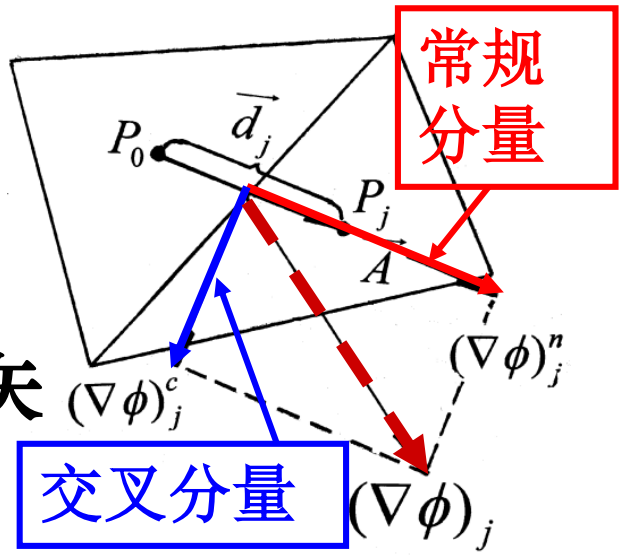
1）沿着P_0-P_j连线方向的扩散在界面*j*上的投影-----称为常规分量；

2）沿着与P_0-P_j连线的垂直方向的扩散在界面*j*上的投影------称为交叉分量。

由于常规(normal)分量并不垂直于界面，因此交叉分量也不平行于界面。

1. 常规分量与交叉分量

$$D_j = -\int \Gamma_\phi \nabla\phi \bullet d\vec{A} = D_j^n + D_j^c$$



2. 常规分量的计算

常规分量是指沿着P_0-P_j连线的分矢量在面积矢量 $\vec{A}$ 上的投影：

$$D_j^n = -\Gamma_\phi\left[\frac{\phi_{P_j}-\phi_{P_0}}{\left|\vec{d}_j\right|}\frac{\vec{d}_j}{\left|\vec{d}_j\right|}\right]\bullet\vec{A}_j = -\Gamma_\phi(\phi_{P_j}-\phi_{P_0})\frac{\vec{d}_j\bullet\vec{A}_j}{\left|\vec{d}_j\right|^2}$$

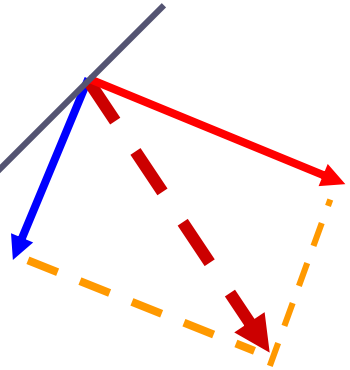
连线上变量的梯度

在面积 A_j 上的投影

3. 交叉分量的计算

交叉分量是指沿着垂直于P_0-P_j连线方向的分矢量在面积矢量 $\vec{A}_j$ 上的投影，

$$D_j^c = -\Gamma_\phi \left\{ \underline{(\nabla\phi)_j} - [(\nabla\phi)_j \bullet \frac{\vec{d}_j}{|\vec{d}_j|}] \frac{\vec{d}_j}{|\vec{d}_j|} \right\} \bullet \vec{A}_j \; A_j$$



总梯度与常规矢量的矢量差

总梯度在两节点连线方向的投影

在连线上变量的梯度—常规矢量

在面积Aj 上的投影

4. 源项的离散

采用源项线性化的方法： $S_\phi = S_C + S_P\phi_P$

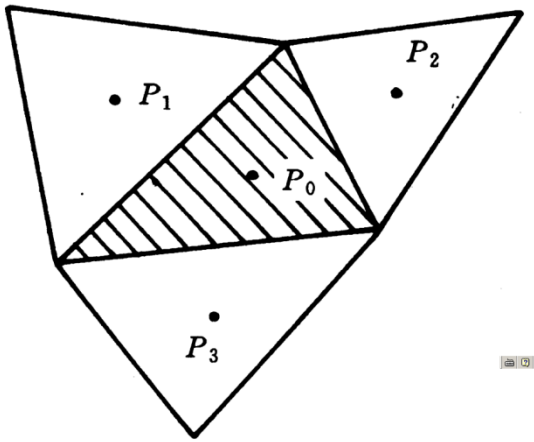
$$\int_{V_{P_0}} S_\phi dV = S_C V_{P_0} + S_P \phi_{P_0} V_{P_0}$$

非结构化网格上对流扩散方程的离散过程汇总

积分型控制方程

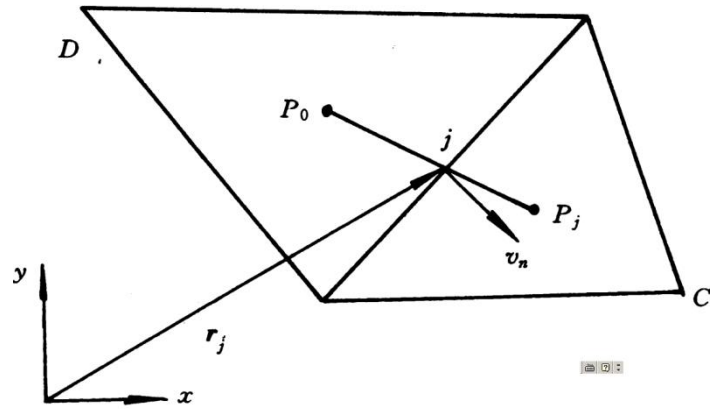
$$\int_A (\rho\vec{u}\phi - \Gamma_\phi \nabla\phi)\bullet d\vec{A} = \int_V S_\phi dV$$

$$\sum_{j=1}^{3}\int_{A_j} (\rho\vec{u}\phi - \Gamma_\phi \nabla\phi)\bullet d\vec{A} = \int_V S_\phi dV$$



方程离散

$$\sum_{j=1}^{3}(C_j + D_j) = \int_{V_{P_0}} S_\phi dV$$



对流通量

$$C_j = \int_{A_j} \rho\vec{u}\phi \bullet d\vec{A}_j = F_j\phi_j$$

界面插值

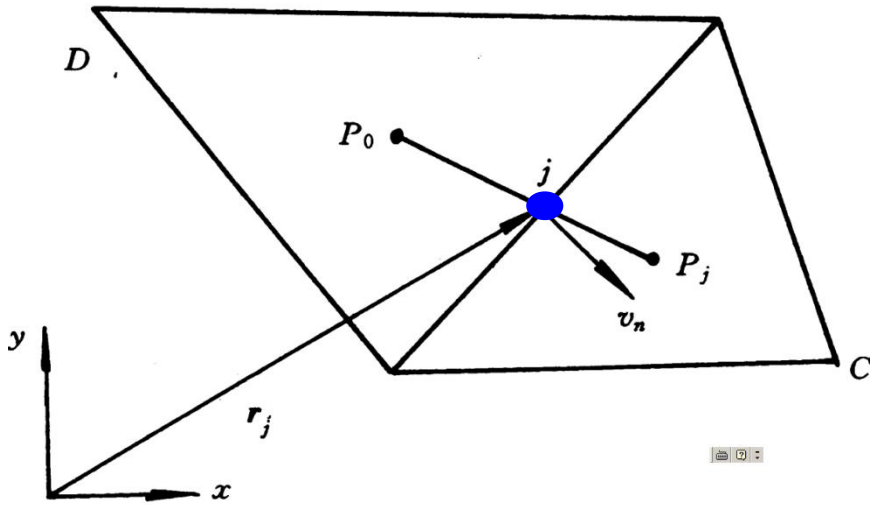
$$\phi_j = \begin{cases} \phi_{P_0} + \gamma(\nabla\phi)_{P_0}\bullet(\vec{r}_j - \vec{r}_{P_0}), F_j > 0 \\ \phi_{P_j} + \gamma(\nabla\phi)_{P_j}\bullet(\vec{r}_j - \vec{r}_{P_j}), F_j < 0 \end{cases} \quad (0 \le \gamma \le 1)$$

需要知道单元P_0的梯度

$$(\nabla\phi)_{P_o}=(\frac{\partial\phi}{\partial x})_{P_o}\vec{\mathbf{i}}+(\frac{\partial\phi}{\partial y})_{P_o}\vec{\mathbf{j}}$$

$$(\frac{\partial\phi}{\partial x_i})_{P_0}\cong\frac{\int_V(\frac{\partial\phi}{\partial x_i})dV}{V_{P_0}}=\frac{\sum_{j=1}^{3}\phi_j A_j^i}{V_{P_0}}\quad(i=1,2)$$

需要知道单元P_0的界面流速



界面流速

在 $\vec{A}_j$ 方向上的一个速度小量(包含相邻2点压差)

$$\vec{u}_j = \vec{u}_{P_0 i} + K_j \left\{ \frac{1}{2}\left[(\nabla p)_{P_0} \bullet \frac{\vec{d}_j}{|\vec{d}_j|} + (\nabla p)_{P_j} \bullet \frac{\vec{d}_j}{|\vec{d}_j|}\right] - \frac{p_{P_j} - p_{P_0}}{|\vec{d}_j|} \right\} \frac{\vec{A}_j}{|\vec{A}_j|}$$

$\vec{u}_{P_0}, \vec{u}_{P_j}$ 之间的线性插值。

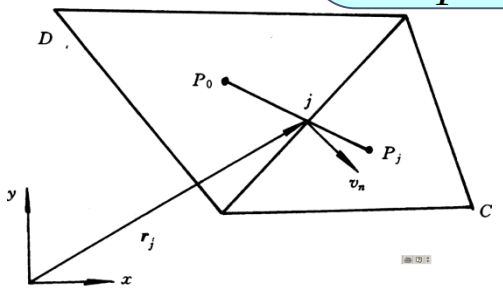
压差项前的系数，相当于：$(\frac{A_P}{a_P})_e$

d_j 方向单位矢量

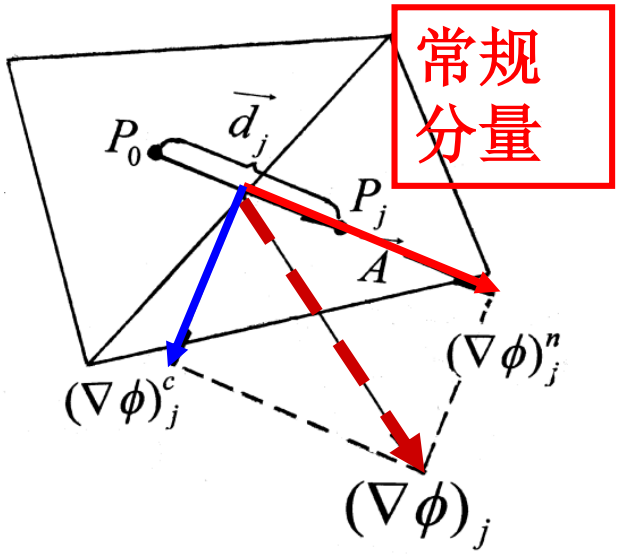
P_0, P_j 单元的平均压力梯度在 d_j 方向的投影，其值与后者十分接近；在结构化均分网格中将不显含相邻两点间的压差。

保证相邻两点间的压差出现在界面流速中的附加项，以衰减波形压力场。

单位向量



扩散通量的常规分量在A_j上的投影



$$D_j^n = -\Gamma_\phi(\phi_{P_j} - \phi_{P_0})\frac{\vec{d}_j \bullet \vec{A}_j}{\left|\vec{d}_j\right|^2}$$

扩散通量的交叉分量在A_j上的投影

$$D_j^c = -\Gamma_\phi\left\{(\nabla\phi)_j - [(\nabla\phi)_j \bullet \frac{\vec{d}_j}{\left|\vec{d}_j\right|}]\frac{\vec{d}_j}{\left|\vec{d}_j\right|}\right\} \bullet \vec{A}_j$$

C_j的紧凑表达方式（兼顾流速的正负两个方向）

$$C_j = F_j\phi_j = \max(F_j,0)[\phi_{P_0} + \gamma(\nabla\phi)_{P_0} \bullet (\vec{r}_j - \vec{r}_{P_0})]$$

$$- \max(-F_j,0)[\phi_{P_j} + \gamma(\nabla\phi)_{P_j} \bullet (\vec{r}_j - \vec{r}_{P_j})]$$

$$= [\max(F_j,0)\phi_{P_0} - \max(-F_j,0)\phi_{P_j}]$$

对流项的一阶迎风部分，进入求解项

$$+\max(F_j,0)\,\gamma\,(\nabla\phi)_{P_0} \bullet (\vec{r}_j - \vec{r}_{P_0}) - \max(-F_j,0)\,\gamma\,(\nabla\phi)_{P_j} \bullet (\vec{r}_j - \vec{r}_{P_j})$$

对流项的其余部分，进入源项

源项的离散 $S_\phi = S_C + S_P\phi_P \quad \int_{V_{P_0}} S_\phi dV = S_C V_{P_0} + S_P\phi_{P_0}V_{P_0}$

7.6.5 离散方程的最终形式

$$\sum_j (C_j + D_j) = \int_V S_\phi dV$$

对流项离散的一阶迎风部分进入直接求解，其余部分进入源项（延迟修正）；

扩散项中的常规分量进入直接求解部分，交叉分量进入源项（显式处理）；

将含 ϕ_{P_0} 项置于等号前，含 ϕ_{P_j} 项置于等号后.得:

$$\sum_{j=1}^{3}\left\{[\max(F_j,0)+\frac{\Gamma_\phi}{|d_j|^2}(\vec{d_j}\bullet\vec{A_j})-S_P V_{P_0}\right\}\phi_{P_0} = \cdots$$

主节点系数

$$\sum_{j=1}^{3}\left\{[\max(-F_j,0)+\frac{\Gamma_\phi}{|d_j|^2}(\vec{d_j}\bullet\vec{A_j})\right\}\phi_{P_j}+S_C V_{P_0}+$$

源项常数部分

邻点系数

$$\sum_{j=1}^{3}\gamma[(\nabla\phi)_{P_0}\max(F_j,0)(\vec{r_{P_0}}-\vec{r_j})+\sum_{j=1}^{3}\gamma[(\nabla\phi)_{P_0}\max(-F_j,0)(\vec{r_j}-\vec{r_{P_j}})]$$

对流部分的延迟修正

$$+\sum_{j=1}^{3}\Gamma_\phi[(\nabla\phi)_j-(\nabla\phi)_j\bullet\frac{\vec{d_j}}{|\vec{d_j}|}\frac{\vec{d_j}}{|\vec{d_j}|}]\bullet\vec{A_j}$$

交叉扩散部分显式处理

简写之：

$$a_{P_0}\phi_{P_0} = \sum_{j=1}^{3} a_{P_j}\phi_{P_j} + b$$

$$a_{P_0} = \sum_{j=1}^{3} a_{P_j} - S_P V_{P_0}$$

要求质量守恒成立

$b =$ 源项常数部分 + 对流项延迟修正部分 +

交叉扩散部分

7.6.6 压力修正方程的导出

类似于结构化网格上SIMPLE算法的压力修正方程的推导过程，由界面流速的计算公式，

$$\vec{u}_j = \vec{u}_{P_0 j} + K_j \left\{ \frac{1}{2}[(\nabla p)_{P_0} \bullet \frac{\vec{d}_j}{|\vec{d}_j|} + (\nabla p)_{P_j} \bullet \frac{\vec{d}_j}{|\vec{d}_j|}] - \frac{p_{P_j} - p_{P_0}}{|\vec{d}_j|} \right\} \frac{\vec{A}_j}{|\vec{A}_j|}$$

可得:

$$\vec{u}_j' = -\frac{1}{2}[(\frac{V}{a_0^u})_{P_0} + (\frac{V}{a_0^u})_{P_j}] \frac{p_{P_j}' - p_{P_0}'}{|d_j|} \frac{\vec{A}_j}{|\vec{A}_j|} = -\overline{(\frac{V}{a_0^u})_{P_j}} \frac{p_{P_j}' - p_{P_0}'}{|d_j|} \frac{\vec{A}_j}{|\vec{A}_j|}$$

速度修正值公式

要求 $(\vec{u}_j^* + \vec{u}_j')$ **满足质量守恒，可导出压力修正值方程**

$$a_0^p p_{P_0}' = \sum_{j=1}^{3} a_{P_j}^p p_{P_j}' + b^p \quad \text{(齐次Neumann边界条件)}$$

$$a_j^p = \rho(\frac{V}{a_0^u})_{P_j} \frac{|\vec{A}_j|}{|\vec{d}_j|}, \qquad a_0^p = \sum_{j=1}^{3} a_j^p, \qquad b^p = \sum_{j=1}^{3} F_j^*$$

7.6.7 非结构化网格上SIMPLE算法的实施步骤

1. 假定一个速度场 $\overrightarrow{u^0}$ ，计算系数与源项;

2. 假定一个压力场 p^*;

3. 求解动量方程，得出临时速度场 $\overrightarrow{u^*}$;

4. 据 $\overrightarrow{u^*}$ 修正压力，获得修正分量p', 要求与p' 对应的 $\overrightarrow{u'}$ 使 $(\overrightarrow{u^*}+\overrightarrow{u'})$ 满足质量守恒，由此导出压力修正方程并求解之;

5. 获得p'后修正速度和压力;

界面流速 $$\vec{u}_j = \vec{u}_j^* - \left(\frac{\overline{V}}{a_0^u}\right)_{P_j} \frac{p'_{P_j} - p'_{P_0}}{|d_j|} \frac{\vec{A}_j}{|\vec{A}_j|}$$

节点压力 $$p_{P_0} = p_{P_0}^* + \alpha_p p'_{P_0}$$

节点流速 $$\vec{u}_{P_0} = \vec{u}_{P_0}^* - \frac{V}{a_0^u} \nabla p'_{P_0}$$

6. 以 $\vec{u}$ 以及p*+ $\alpha_p p$'开始下一层次的迭代计算。

非结构化网格上离散方程的求解难以应用结构化网格上行之有效的ADI－TDMA方法，一般采用G－S迭代法或者共轭梯度法。

7.7 SIMPLE系列算法向可压缩流的发展简介

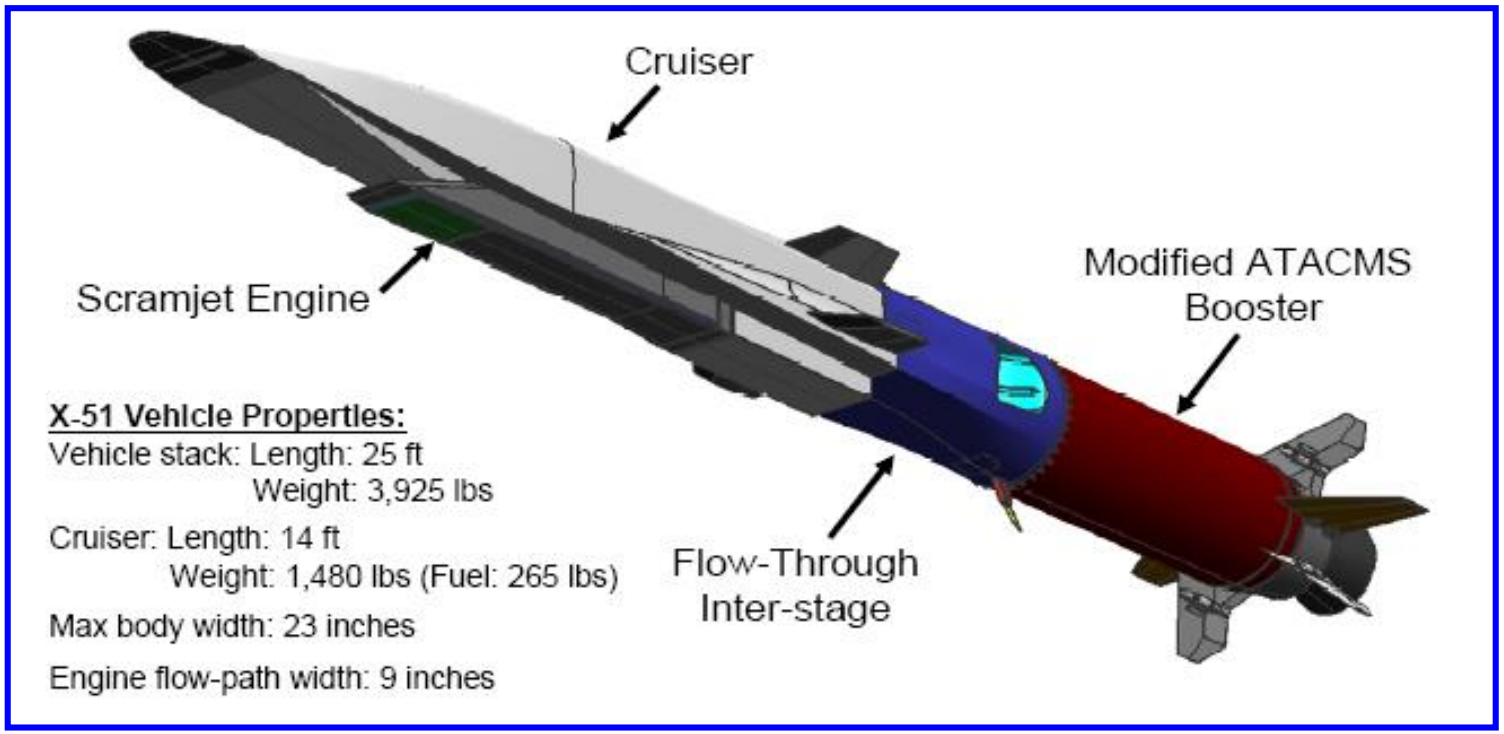
7.7.1 引言－发展全速算法的必要性

7.7.2 目前构建流场全速算法的两种途径

7.7.3 基于压力算法的可压缩流与不可压缩流算法的主要区别

7.7.1 引言－发展全速算法的必要性

美国的X-51A飞行器在2010年5月26日进行的第一次飞行速度达到了5Ma。



http://www.cnhuu.com/junshi/chinajq/200804/junshi_195112.html

高超声速飞行器从地面起飞到达一定高度的飞行中，其四周气体的流速经历了两种变化；

(1) 从不可压缩流动到高Ma数流动；

(2) 从连续介质流动到稀薄气体薄流动。

由于特别适用于稀薄气体流动模拟的DSMC计算十分费时，因此只有在必须使用DSMC的局部地区才使用，其余地区仍然采用连续介质的数值方法。

因此发展一种能同时进行不可压缩与可压缩流畅计算的所谓“全速”算法（all speeds)具有重要意义，引起了全世界的关注。

7.7.2 目前构建流场全速算法的两种途径

1.基于密度方法-从可压缩流动延拓到不可压缩流动

以密度为基本变量的方法以(u,v,w, ρ)为求解变量，最初是对欧拉方程求解提出的，后扩大到粘性流动，代数方程求解常采用直接解法。

基于密度的方法不适用用于马赫数很小的流动：在这些情况下密度的变化很小甚至不变，压力与密度之间的耦合变得很弱。

最近十余年随着航天航空事业的发展，将基于密度的方法向不可压缩推广引起了广泛关注。

2.基于压力的方法-从不可压缩流动延拓到可压缩流动

无论流动马赫数为多少，压力的变化总是存在的。因此以压力作为基本变量的算法有望可以适用于整个马赫数范围，气液固三相耦合特别适宜。

从1989年出现将基于压力的方法推广到可压缩流动至今，已经提出了十余种将基于压力的算法推广到全速流动的计算。

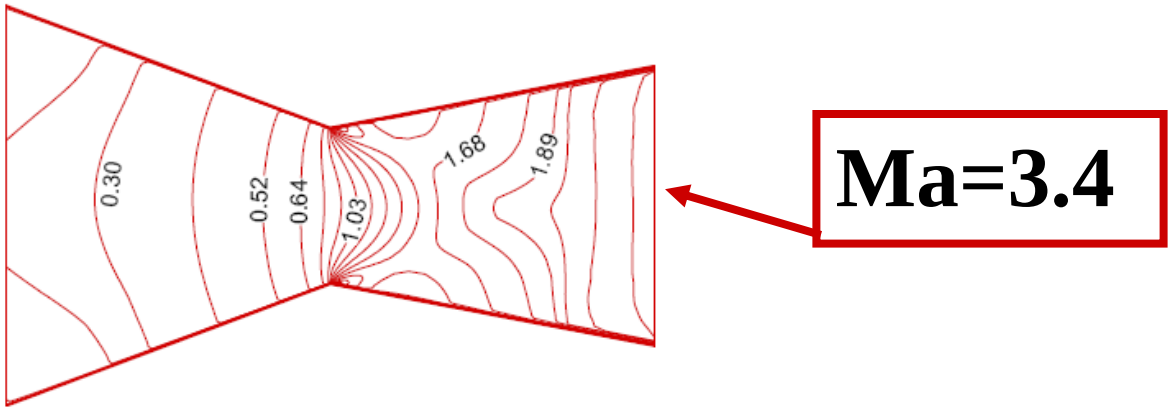
7.7.3 基于压力算法的可压缩流与不可压缩流算法的主要区别

(1) 修正速度时除了考虑压力修正的作用外还需考虑密度修正引起的变化；

(2) 对流项的离散格式与不可压缩流动有区别；

(3) 速度与压力边界条件的处理与不可压缩流不同。

本团队两篇博士论文（屈治国，2005；张宏伟，2006）喷管出口Ma分别达到3.4与6.7。

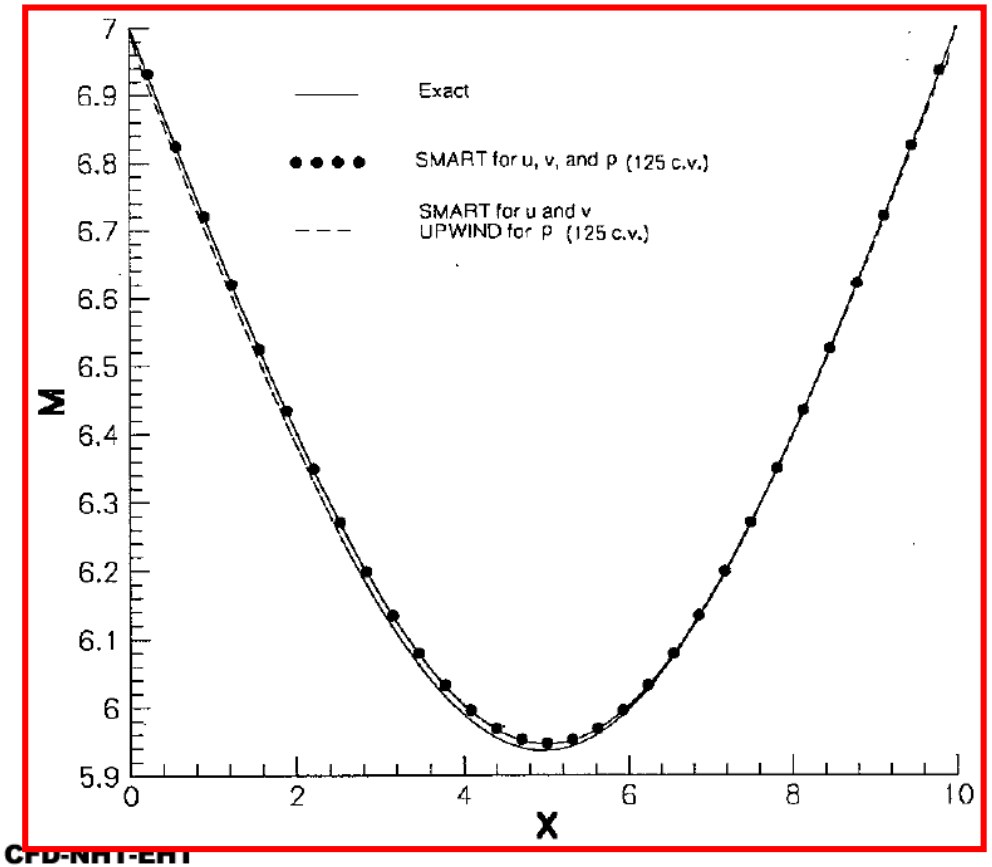


屈治国.流动传热问题先进算法及其在强化空气对流传热应用中的研究.西安交通大学,2005.

张宏伟．高压推力室流动与传热特性及液膜冷却的数值模拟研究．西安交通大学，2006.

在欧盟支持下Moukalled and Darwish将基于压力的算法计算到Ma=7，且获得很高的计算精度。

1）将规整变量公式（NVF）用于对流项的离散；2）将高分辨率格式用来进行密度插值。



缩放喷管无粘流动沿程马赫数分布对比，进口马赫数为7

Moukalled F and Darwish M. A high-resolution pressure-based algorithm for fluid flow at all speeds. A report submitted to European Office of Aerospace Research and Development (EOARD), SPC-99-4003.

将基于压力的算法拓宽到高超声速下的高效计算仍然是一个难题，当前的进展不大。基于密度的算法可能是个基本选择。

Karki KC, Patankar SV. Pressure based calculation procedure for viscous flows at all speeds in arbitrary configurations. AIAA J, 1989, 27(9):1167-1174

Shyy W, Chen M H, Sun C S. Pressure based multigrid algorithm for flow at all speeds. AIAA J, 1992, 30,2660-2669

Shyy W, Chen M H, Sun C S. Pressure based multigrid algorithm for flow at all speeds. AIAA J, 1992, 30,2660-2669

Demirdzic I, Leilek I, Peric M. A collocated finite volume method for predicting flows at all speeds. Int J Numer Methods Heat Fluids. 1993, 16: 1029 -1050

Date A W. Solutions of Navier-Stokes equations on non-staggered grids of all speeds. Numerical Heat Transfer, Part B,1998, 33:451-467

同舟共济
渡彼岸!

People in the same boat help each other to cross to the other bank, where….