Numerical Heat Transfer

(数值传热学)

Chapter 10 General Code for 2D Elliptical Fluid Flow and Heat Transfer (1)

Instructor： Fang, Wen-Zhen

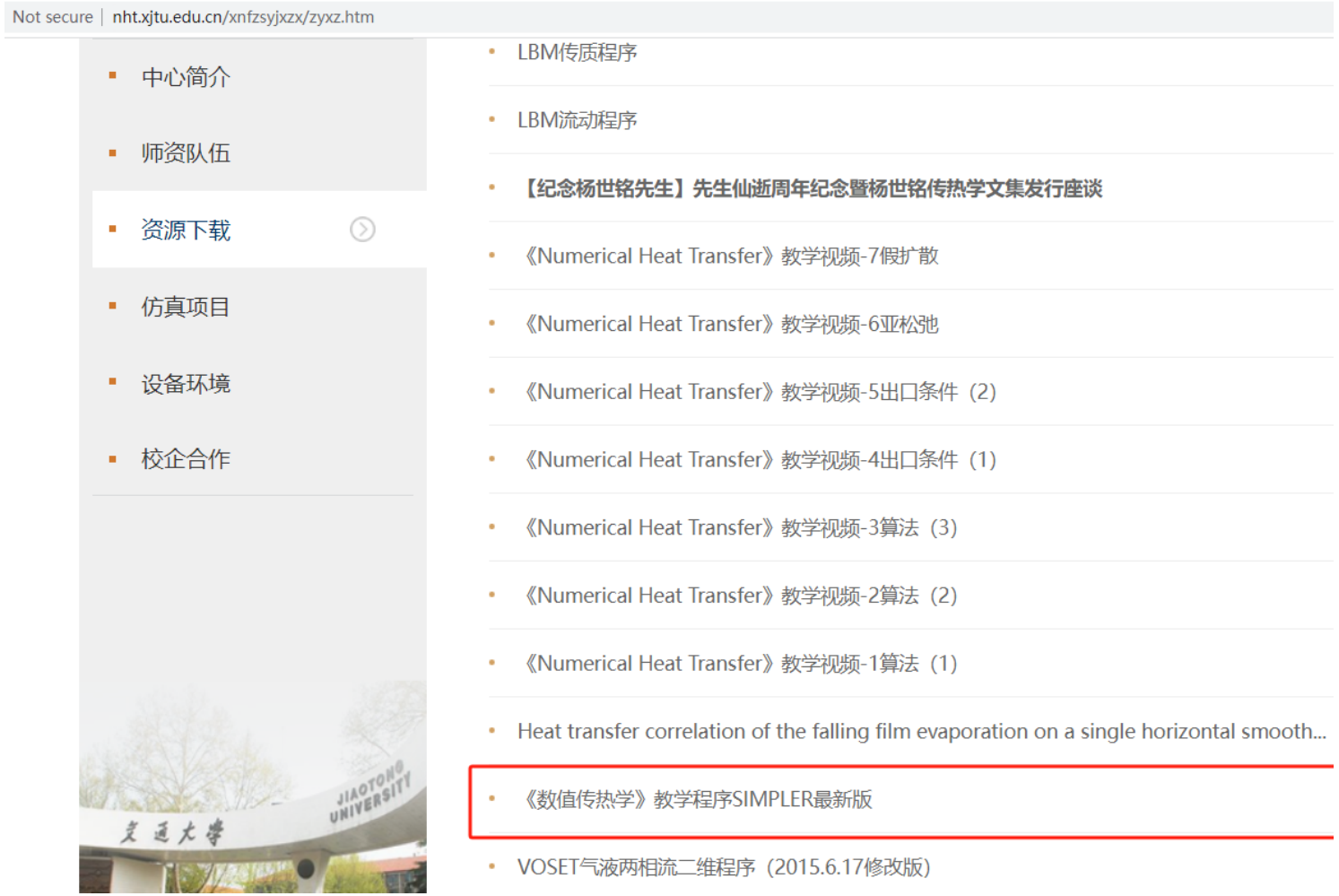
Email: fangwenzhen@xjtu.edu.cn

Key Laboratory of Thermo-Fluid Science & Engineering
Xi'an Jiaotong University

2024-Nov-25

Website to Download SIMPLER Teaching Code

http://nht.xjtu.edu.cn/xnfzsyjxzx/zyxz.htm



8 examples:

- EXAM1
- EXAM2
- EXAM3
- EXAM4
- EXAM5
- EXAM6
- EXAM7
- EXAM8

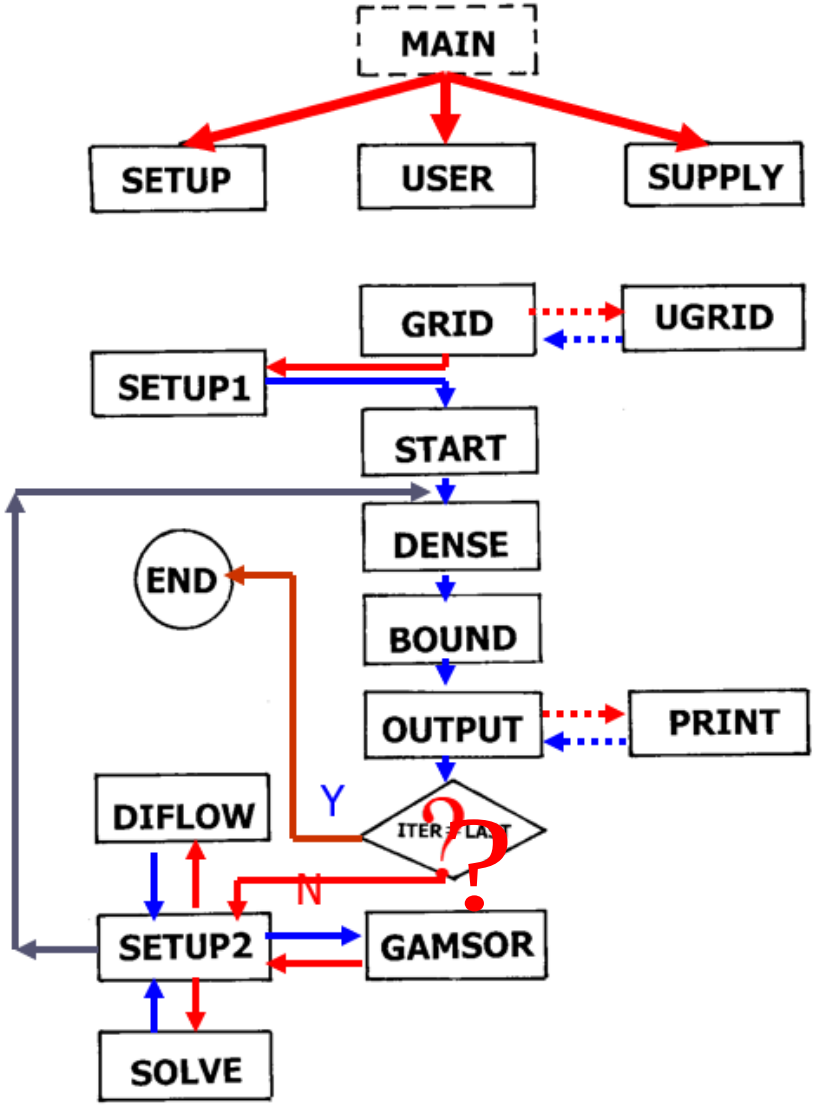
Exam 1:

- Main95.f
- ONE.F

Exam 2:

- Main95.f
- TWO.F

Structure of Teaching Code



Teaching Code

Main Program

No need to change

Main95.f

USER

Varied to deal with a specific problem

ONE.F

TWO.F

Chapter 10 General Code for 2D Elliptical Fluid Flow and Heat Transfer Problems

10.1 Format Improvement of General Governing Equation

10.2 Numerical Methods Adopted and Discretization Equations

10.3 Code Structure and Module Functions

10.4 Grid System

10.5 Techniques Adopted in the Code

10.6 Methods of Application and Explanation of MAIN Program

Main95.f

10.1 Format Improvement of the General Governing Equation

The G.E. we just learned in the previous chapters is:

$$\frac{\partial(\rho\phi)}{\partial t}+div(\rho\phi\vec{U})=div(\Gamma_{\phi}^{*}grad\phi)+S_{\phi}^{*} \qquad (1)$$

Equation	ρ	ϕ	Γ_{ϕ}^{*}	S_{ϕ}^{*}
Continuity equation	ρ	1	0	0
Momentum eqn. (x direction)	ρ	u	μ	$\rho f_x-\frac{\partial p}{\partial x}$
Momentum eqn. (y direction)	ρ	v	μ	$\rho f_y-\frac{\partial p}{\partial y}$
Energy equation	ρ	T	λ/c_p	S_T/c_p

$$\frac{\partial(\rho T)}{\partial t}+\frac{\partial(\rho uT)}{\partial x}+\frac{\partial(\rho vT)}{\partial y}=\frac{\partial}{\partial x}\left(\frac{\lambda}{c_p}\frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(\frac{\lambda}{c_p}\frac{\partial T}{\partial y}\right)+\frac{S_T}{c_p}$$

When the fluid heat capacity is not constant, numerical results may not satisfy the conservation condition.

10.1.1 Analysis for energy equation

A correct energy equation is:

$$\frac{\partial(\rho c_p T)}{\partial t}+\frac{\partial(\rho c_p u T)}{\partial x}+\frac{\partial(\rho c_p v T)}{\partial y}=\frac{\partial}{\partial x}\left(\lambda\frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(\lambda\frac{\partial T}{\partial y}\right)+S_T. \quad (2)$$

Rewrite into the previous format of G.E.:

$$\frac{\partial(\rho T)}{\partial t}+\frac{\partial(\rho u T)}{\partial x}+\frac{\partial(\rho v T)}{\partial y}=\frac{\partial}{\partial x}\left(\frac{\lambda}{c_p}\frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(\frac{\lambda}{c_p}\frac{\partial T}{\partial y}\right)+\frac{S_T}{c_p}$$

$$-\frac{1}{c_p^2}\left[\rho c_p T\frac{\partial c_p}{\partial t}+\left(\rho c_p u T-\lambda\frac{\partial T}{\partial x}\right)\frac{\partial c_p}{\partial x}+\left(\rho c_p v T-\lambda\frac{\partial T}{\partial y}\right)\frac{\partial c_p}{\partial y}\right] \quad (3)$$

This term was neglected in the previous G.E.

When it can be neglected?

(1) c_p is constant;

(2) Or following three terms simultaneously equal zero:

$$\rho c_p u T - \lambda \frac{\partial T}{\partial x} = 0, \quad \rho c_p v T - \lambda \frac{\partial T}{\partial y} = 0 \quad \text{and} \quad \rho c_p T \frac{\partial c_p}{\partial t} = 0 \tag{4}$$

(3) Or the sum of the three terms equal zero!

Obviously, such chances are quite limited! Hence, simulation results based on previous G.E. inherently include some errors!

10.1.2 Improved format of the general G.E.

The framework (框架) of the previous general G.E. is retained (保留), but the diffusion coefficient is resumed to (恢复到) its original value (λ) by introducing a nominal density (ρc_p) as follows:

$$\frac{\partial(\rho^* \phi)}{\partial t} + div(\rho^* \phi \vec{U}) = div(\Gamma_\phi grad\phi) + S_\phi^* \tag{5}$$

The new form of G.E. for different variables are:

Equation	ρ^*	ϕ	Γ_ϕ	$S_\phi{}^*$
Continuity equation	ρ	1	0	0
Momentum eqn. (x direction)	ρ	u	μ	$\rho f_x - \frac{\partial p}{\partial x}$
Momentum eqn. (y direction)	ρ	v	μ	$\rho f_y - \frac{\partial p}{\partial y}$
Energy equation	ρc_p	T	λ	S_T

That is, now we regard ρc_p in the energy equation as a general density:

$$\frac{\partial(\rho c_p T)}{\partial t}+\frac{\partial(\rho c_p u T)}{\partial x}+\frac{\partial(\rho c_p v T)}{\partial y}=\frac{\partial}{\partial x}\left(\lambda\frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(\lambda\frac{\partial T}{\partial y}\right)+S_T.$$

Such a treatment is much better than taking λ / c_p as a nominal diffusion coefficient and S_T / c_p as a nominal source term .

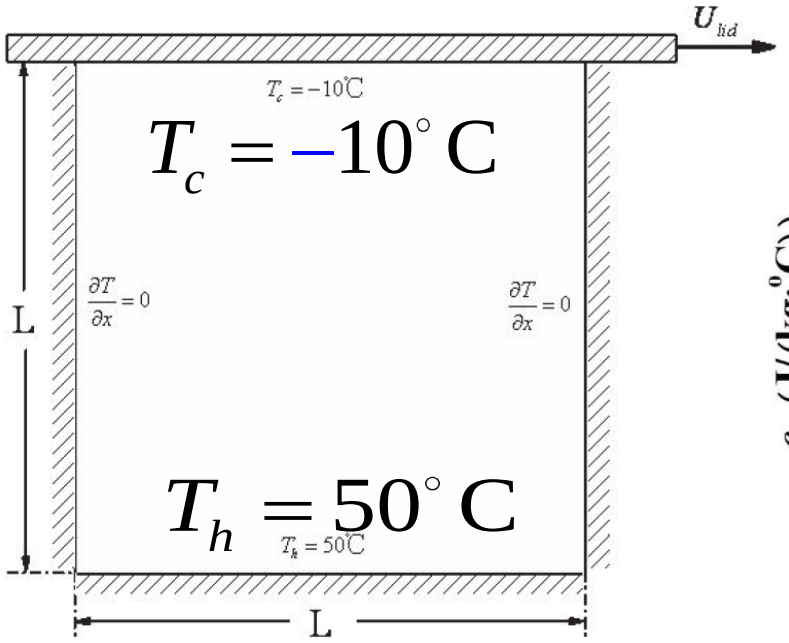
Li W, Yu B, Wang Y , et al. Communications in Computational Physics, 2012, 12(5): 1482-1494

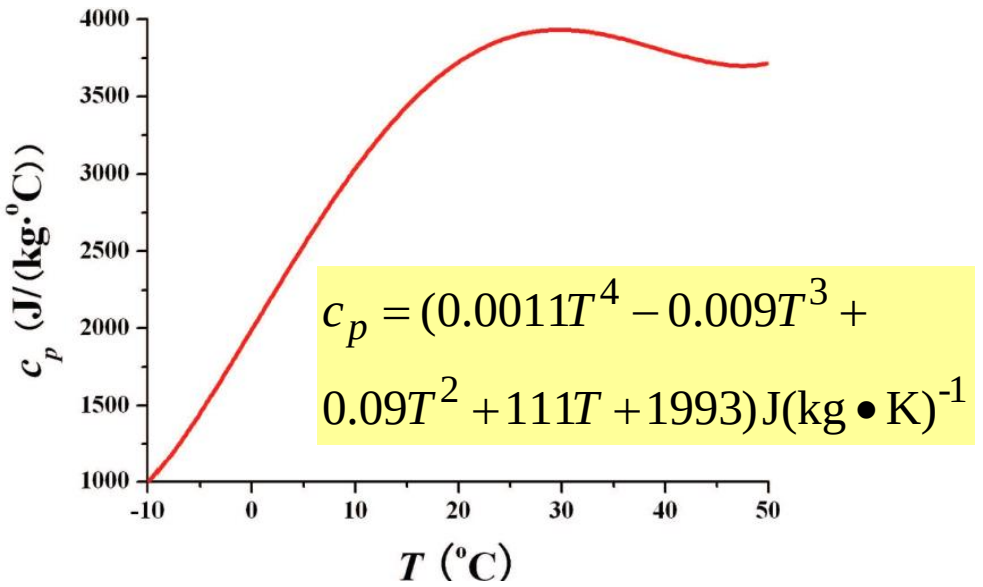
10.1.3 Comparisons of energy G.E.

(Old)
$$\frac{\partial(\rho T)}{\partial t}+\frac{\partial(\rho u T)}{\partial x}+\frac{\partial(\rho v T)}{\partial y}=\frac{\partial}{\partial x}\left(\frac{\lambda}{c_p}\frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(\frac{\lambda}{c_p}\frac{\partial T}{\partial y}\right)+\frac{S_T}{c_p} \quad (1)$$

(New)
$$\frac{\partial(\rho c_p T)}{\partial t}+\frac{\partial(\rho c_p u T)}{\partial x}+\frac{\partial(\rho c_p v T)}{\partial y}=\frac{\partial}{\partial x}\left(\lambda\frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(\lambda\frac{\partial T}{\partial y}\right)+S_T. \quad (5)$$

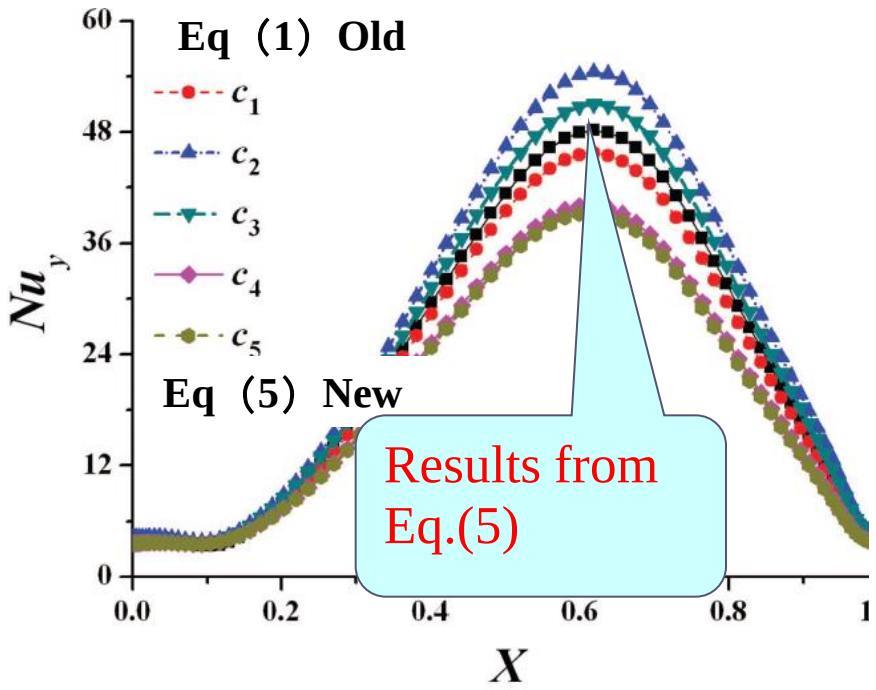
1）Example 1 ---Natural convection in enclosure with a moving lid and variable fluid heat capacity



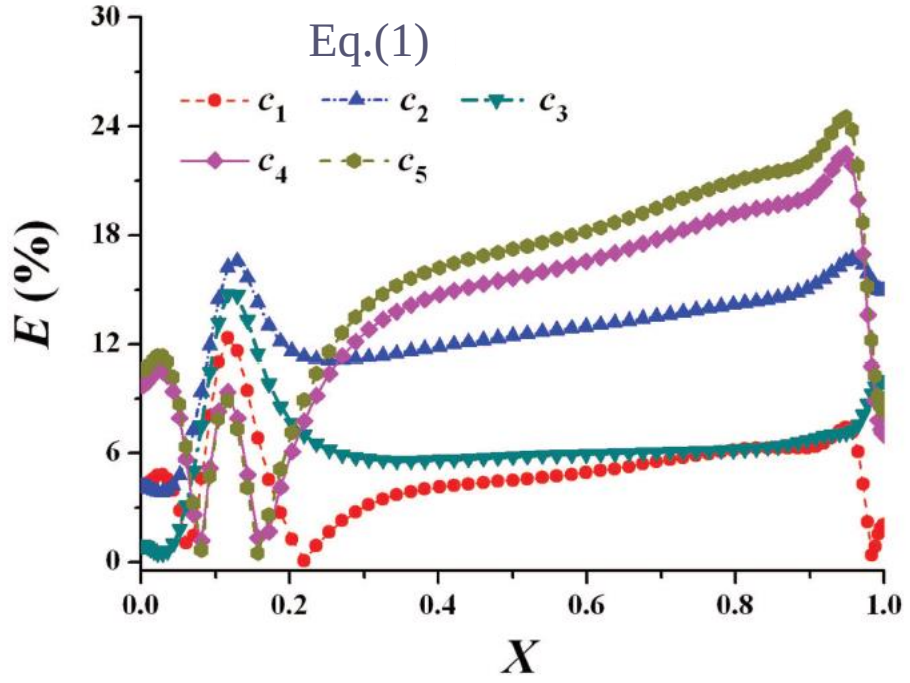


For the new G.E. of Eq.5, fluid heat capacity can be determined by the local temperature (correct one).

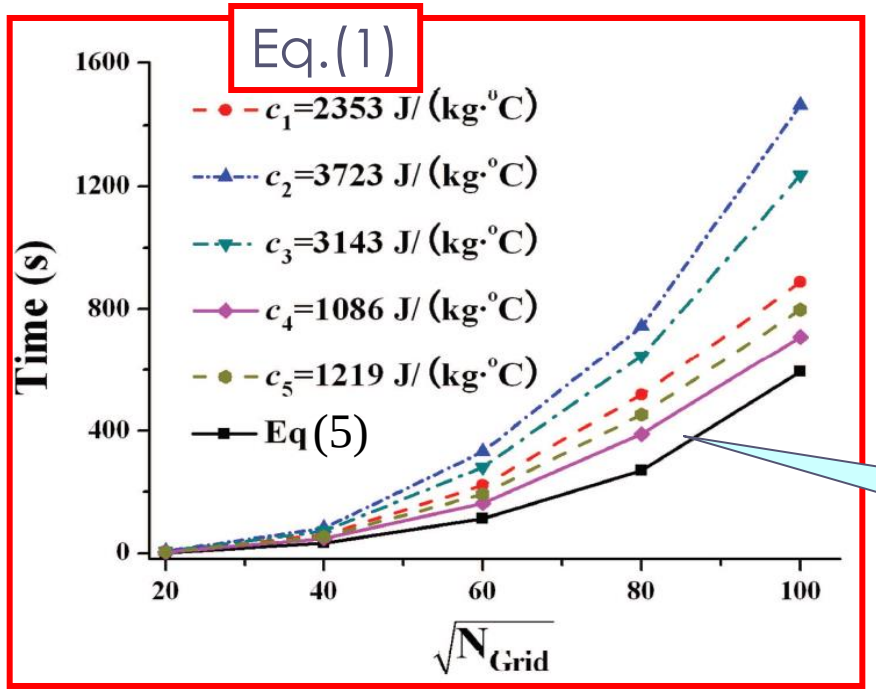
For original G.E. of Eq.1, five average c_p are used, which lead to large deviations with the correct one.



(a) Nusselt number



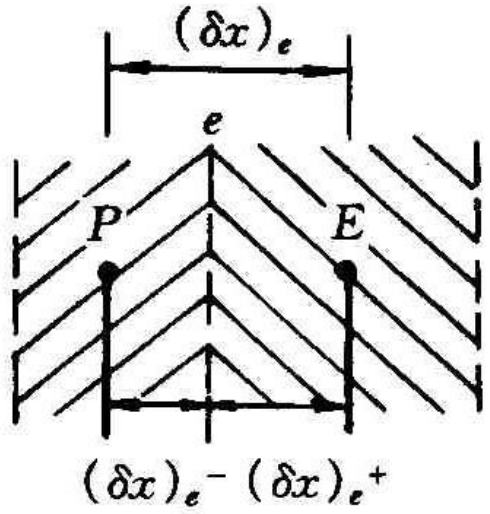
(b) Relative deviation



The convergence rate of the new G.E. is significantly higher than that of the previous format.

Results from Eq.(5)

2）Example 2 ---Conjugated problems(耦合问题)



$$\frac{(\delta x)_e}{\lambda_e} = \frac{(\delta x)_{e^+}}{\lambda_E} + \frac{(\delta x)_{e^-}}{\lambda_P}$$

Harmonic mean

Harmonic mean should be used to ensure continuity of heat flux. In the original G.E., diffusivity is nominal, $\Gamma = \lambda / c_p$; Thus for conjugated problem where fluid and solid are coupled, we should assume c_p are the same for the fluid and solid:

Harmonic mean for conjugated problem

$\Gamma = \lambda$

$$\frac{(\delta x)_e}{\lambda_e} = \frac{(\delta x)_{e^+}}{\lambda_E} + \frac{(\delta x)_{e^-}}{\lambda_P}$$

$$\frac{(\delta x)_e}{\Gamma_e} = \frac{(\delta x)_{e^+}}{\Gamma_E} + \frac{(\delta x)_{e^-}}{\Gamma_P}$$

$\Gamma = \lambda / c_p$

$$\frac{(\delta x)_e}{\lambda / c_{pf}} = \frac{(\delta x)_{e^+}}{\lambda_E / c_{pf}} + \frac{(\delta x)_{e^-}}{\lambda_P / c_{pf}}$$

For steady problems, such treatment is OK, while for transient problem, it does not work. For the new G.E., the diffusivity $\Gamma = \lambda$, so there is no such problem at all.

10.2 Numerical Methods Adopted in Teaching Code and Discretization Equations

10.2.1 Major numerical methods adopted

1.Primitive variable method (原始变量法): Dependent variables are u, v, p ;

2.Practice B of domain discretization: first determine the interfaces positions then the node positions;

3.Finite volume method (FVM) for discretization: to ensure conservation;

4. Staggered (交错) grids: three grid systems for u, v and p;

5.Power-law(乘方格式) for convection-diffusion discretization:

But easy to be replaced by CD, FUD or HS; For higher-order schemes, adopting defer correction(延迟修正) method;

6. Linearization for source term:

$$S = S_C + S_P \phi_P \,,\; S_P \le 0$$

7. Harmonic mean for interface diffusivity:

$$\frac{(\delta x)_e}{\lambda_e} = \frac{(\delta x)_{e^+}}{\lambda_E} + \frac{(\delta x)_{e^-}}{\lambda_P}$$

8. **Fully implicit** for transient problems: space derivatives are determined by the end instant of a time step;

9. Boundary conditions (BCs) treated as 1st kind: adopting additional source term method(ASTM) for 2nd and 3rd kinds BCs

10. SIMPLER algorithm to treat the coupling between velocity and pressure

$$\frac{\partial(\rho u)}{\partial t}+div(\rho u\vec{U})=div(\mu grad\phi)-\frac{\partial p}{\partial x}$$ (No Eq. for Pressure)

11. Iterative method for solving discretized equations:

$$a_P\phi_P=a_E\phi_E+a_W\phi_W+a_N\phi_N+a_S\phi_S+b$$

1) Iterative method for solving algebraic equations (inner iteration);

2) Iterative method for nonlinear Eqs. (outer iteration);

12. Alternative Direction Iteration (ADI) with block correction to solve algebraic equations.

10.2.2 Discretized equation of three kinds of variables

1. General scalar variable (标量) ϕ

For all other scalar variables (including T, etc)

$$\frac{\partial(\rho^{*}\phi)}{\partial t}+div(\rho^{*}\phi\vec{U})=div(\Gamma_{\phi}grad\phi)+S_{\phi}^{*}$$

$$a_P\phi_P=a_E\phi_E+a_W\phi_W+a_N\phi_N+a_S\phi_S+b$$

$$a_E=D_eA(|P_{\Delta e}|)+[\![-F_e,0 \qquad a_W=D_wA(|P_{\Delta w}|)+[\![F_w,0$$

$$a_P=a_E+a_W+a_N+a_S+a_P^0-S_P\Delta V$$

$$a_P^0=\frac{\rho\Delta V}{\Delta t} \qquad b=S_c\Delta V+a_P\phi_P^0$$

Power-law scheme: $A(|P_\Delta|)=|0,(1-0.1|P_\Delta|^5)|$

$$P_{\Delta e}=\frac{F_e}{D_e}=\frac{(\rho^* uA)_e}{(\Gamma A/\delta x)_e}=\frac{\rho_e^* u_e(\delta x)_e}{\Gamma_e}=\left(\frac{\rho^* u\delta x}{\Gamma}\right)_e$$

For temperature: $\Gamma_\phi=\lambda$ $\rho^*=\rho c_p$

2. Pressure & pressure correction equation

The derivation is based on mass conservation $\partial(\rho u_i)/\partial x_i=0$

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

(1) For SIMPLER, substituting $u_e=u_e+(A_e/a_e)(p_P-p_E)$

into discretized mass conservation equation:

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

$$b=(\rho A\tilde{u})_w-(\rho A\tilde{u})_e+(\rho A\tilde{v})_s-(\rho A\tilde{v})_n \qquad a_E=(\rho Ad)_e, a_P=\sum a_{nb}$$

(2) Substituting $u_e = u_e^* + \frac{A_e}{a_e}\left(p_P' - p_E'\right)$ into mass conservation equation:

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

Except b term, a_P and $a_{E,W,N,S}$ are the same as p-equation.

$$b = \left(\rho A u^*\right)_w - \left(\rho A u^*\right)_e + \left(\rho A v^*\right)_s - \left(\rho A v^*\right)_n$$

Remarks: The adopted mass conservation equation does not include $\partial\rho/\partial t$ and velocity correction neglects density effect. Thus, this code only applicable to incompressible flow.

3. Momentum equation (taking u as example)

Governing equation:

$$\frac{\partial(\rho u)}{\partial t}+div(\rho\vec{u}u)=div(\mu gradu)-\frac{\partial p}{\partial x_i}+S_u$$

Discretized equation:

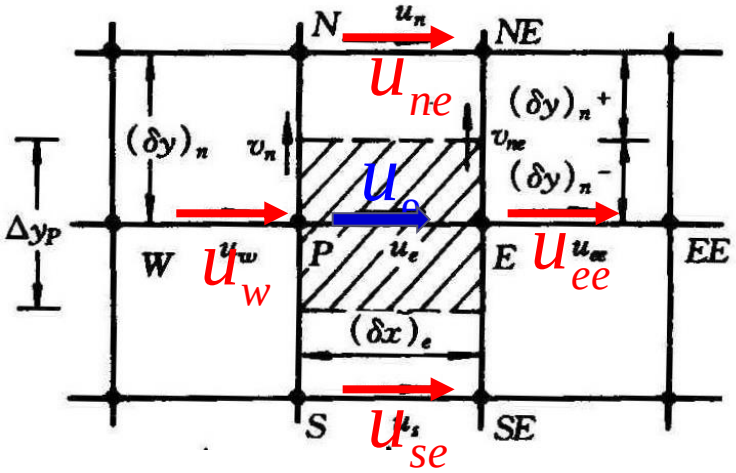
$$a_e u_e = a_{ee}u_{ee}+a_w u_w+a_{ne}u_{ne}+a_{se}u_{se}+b+A_e(p_P-p_E)$$

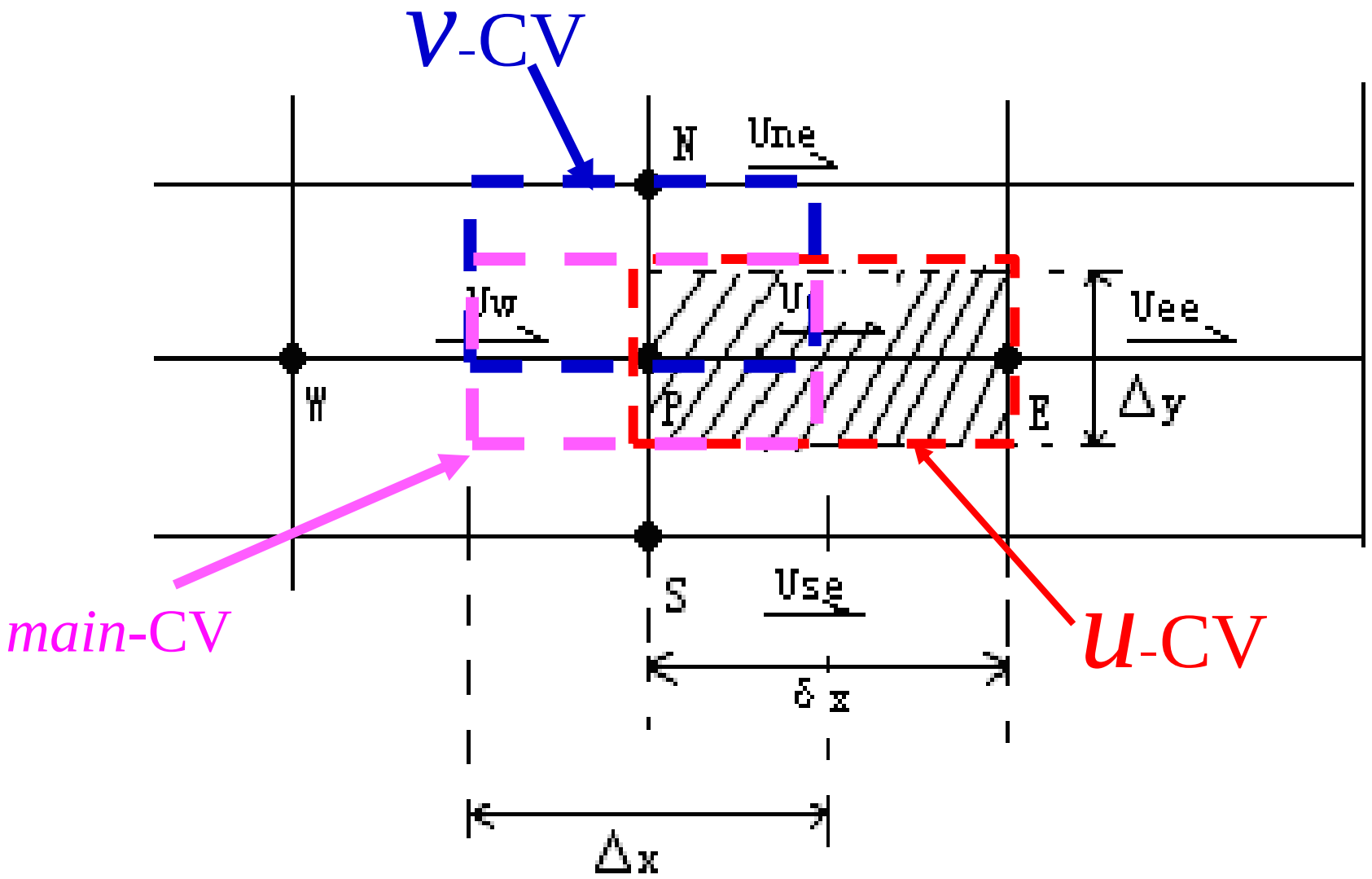
Volume of CV:

For general ϕ scalar variable $\Delta V=\Delta x\cdot\Delta y$

For u: $\Delta V=\delta x\cdot\Delta y$

For v: $\Delta V=\Delta x\cdot\delta y$




v-CV
N
Vne
Vw
Vee
W
P
E
Δy
S
Vse
main-CV
u-CV
δx
Δx

10.2.3 Implementation of under-relaxation

For the solution of non-linear Eqs., changes of variables between two subsequent iterations should not be too large to ensure convergence. Under-relaxation can control the speed of this change:

$$a_P\phi_P = a_E\phi_E + a_W\phi_W + a_N\phi_N + a_S\phi_S + b$$

In the code, the under-relaxation is implemented during the solution procedure:

$$\phi = \phi_P^0 + \alpha\left[\frac{\sum a_{nb}\phi_{nb} + b}{a_P} - \phi_P^0\right] \quad (0<\alpha<1),\ \text{relaxation factor}$$

$$\phi = \phi_P^0 + \alpha[\frac{\sum a_{nb}\phi_{nb} + b}{a_P} - \phi_P^0]$$

Thus:

$$(\frac{a_P}{\alpha})\phi_P = \sum a_{nb}\phi_{nb} + b + (1-\alpha)\frac{a_P}{\alpha}\phi_P^0$$

New $a_P, a_P^{'}$

New b, $b^{'}$

Finally following equation is sent to the solver:

$$a_P{}'\phi_P = \sum a_{nb}\phi_{nb} + b'$$

the obtained solution has been underrelaxed

10.3 Code Structure and Module Functions

10.3.1 Major features of general code

10.3.2 Entire structure of the code

10.3.3 Basic function of major modules

10.3.4 Functions and limitations of the code

10. 3 Code Structure and Module Functions

10.3.1 Major features of general code (通用程序)

General codes may be classified into two categories: One is commercial codes, the other is developed by ourselves.

1.Introduction to commercial codes(商用软件)

Commercial software has a good generality **(通用性)**

Most widely adopted ones include:

PHEONICS, COMSOL, FLUENT, CFX , STAR-CD etc.

Except COMSOL, which adopts FEM, the rest adopts the finite volume method (FVM).

2. **The codes developed by ourselves should also possess some generalities.**

Following techniques are often used:

(1) Adopting module structure

The so called module (模块) is a set of statements, which possess input, output and can implement some functions; Subroutine（子程序）in FORTRAN is a kind of module.

Module structure has a good readability (可读性), and is convenient for maintenance (维护).

(2) The relationship between different modules should be loose (松宽的), so that changes in one module will not affect other modules.

(3) The code is usually **divided into two parts**: unchanged part and user;

For the application of the code, the unchanged part is kept as is (保持不变); It is the **blackbox** for the users. The user part is closely related to the problem to be solved.

user part

(4) Discretization, scheme and solution procedure should belong to three different modules.

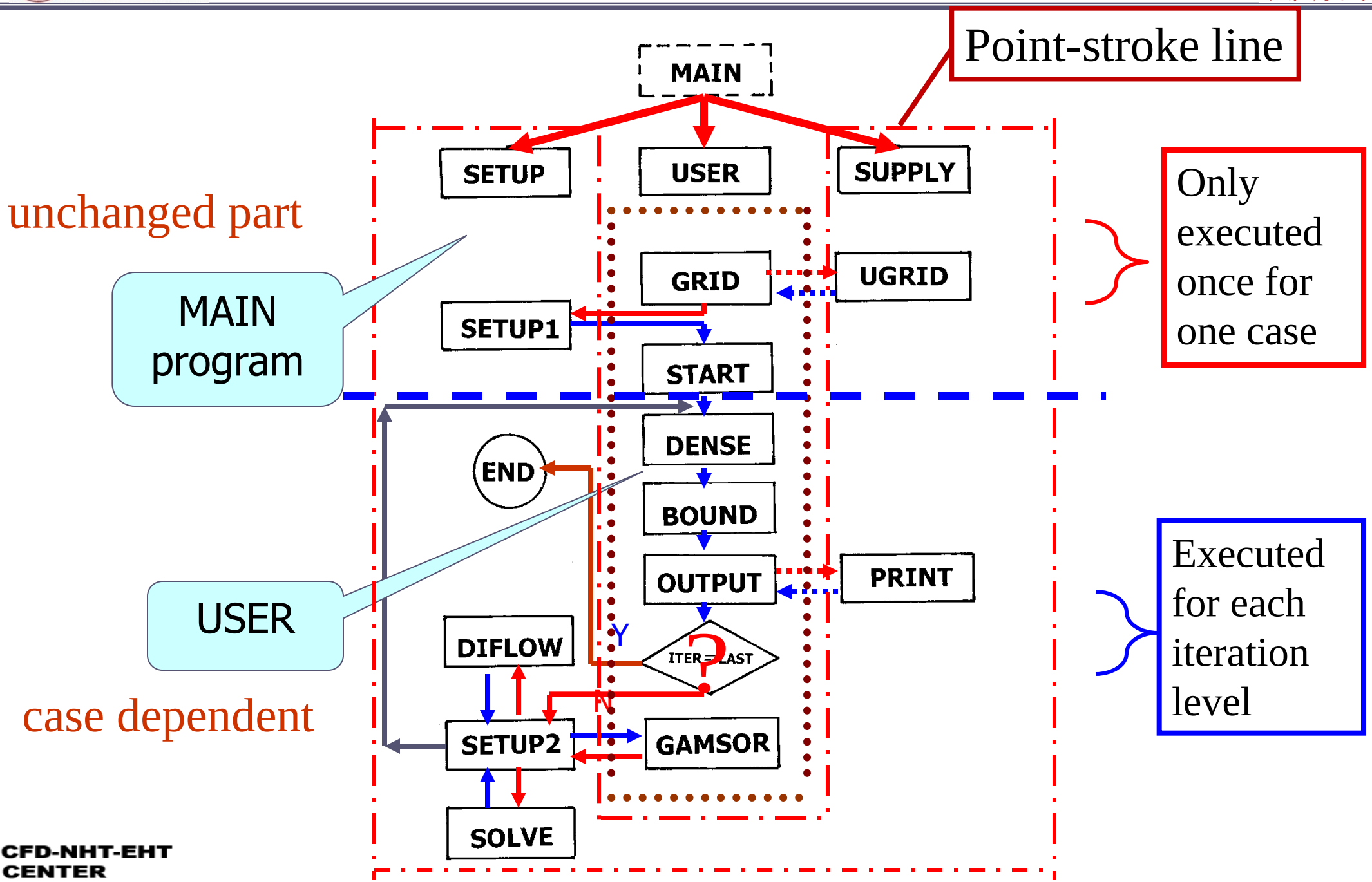
Such a structure is convenient to the studies for the algorithm, scheme and solution methods of the algebraic equations.

(5) For common variables, default values（预置值）should be set up；

(6) A certain pre-process (前处理）and post-process (后处理）functions should be provided.

10.3.2 The entire structure of the teaching code

Our teaching code belong to this category. It divides into two parts: **Main Program (主程序)** and **User （用户）.**


Point-stroke line
MAIN
SETUP
USER
SUPPLY
unchanged part
GRID
UGRID
SETUP1
START
MAIN program
Only executed once for one case
DENSE
END
BOUND
OUTPUT
PRINT
USER
DIFLOW
Y
ITER=LAST
?
N
SETUP2
GAMSOR
SOLVE
case dependent
Executed for each iteration level

10.3.3 Key modules of the teaching code

1．“MODULE”

The ”MODULE” is a terminology (术语) which is for being quoted (被引用) in FORTRAN 90/95 .

MODULE structure is as follows：

```
MODULE module_name
●●●●●●
●●●●●●
●●●●●●
END MODULE
```

Its major function is to be quoted (被引用） by other units of the program----When it is quoted, all the quoting units share the same variables and memory.

```
MODULE START_L
 PARAMETER(NI=100,NJ=200,NIJ=NI,NFMAX=10,NFX4=NFMAX+4)
**********************************************************
CHARACTER*8 TITLE(NFX4)
LOGICAL LSOLVE(NFX4),LPRINT(NFX4),LBLK(NFX4),LSTOP
REAL*8,DIMENSION(NI,NJ,NFX4)::F
REAL*8,DIMENSION(NI,NJ,6)::COF,COFU,COFV,COFP
REAL*8,DIMENSION(NI,NJ)::P,RHO,GAM,CP,CON,AIP,AIM,AJP,
```

Its main function is to declare the type of data (声明数据类型).

When a MODULE is going to be used:

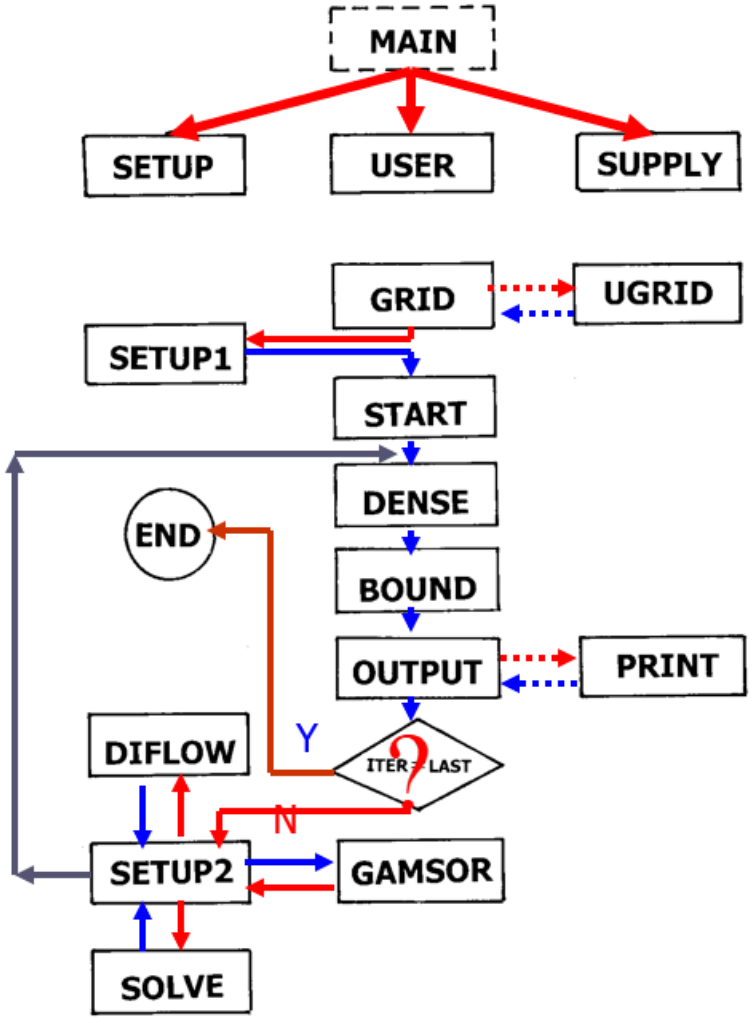
USE module_ name

2．MAIN （different form Main Program）

(1) Set up entire flow chart;

(2) Judge whether terminating the execution of the code.

```
C*************************************
      PROGRAM MAIN
      USE START_L
      IMPLICIT NONE
C*************************************
      OPEN(8,FILE='RESULT.TXT')
      CALL GRID
      CALL SETUP1
      CALL START
      DO WHILE(.NOT.LSTOP)
      CALL DENSE
      CALL BOUND
      CALL OUTPUT
      CALL SETUP2
      ENDDO
      CALL OUTPUT
      CALL CLOSE(8)
      STOP
      END
```

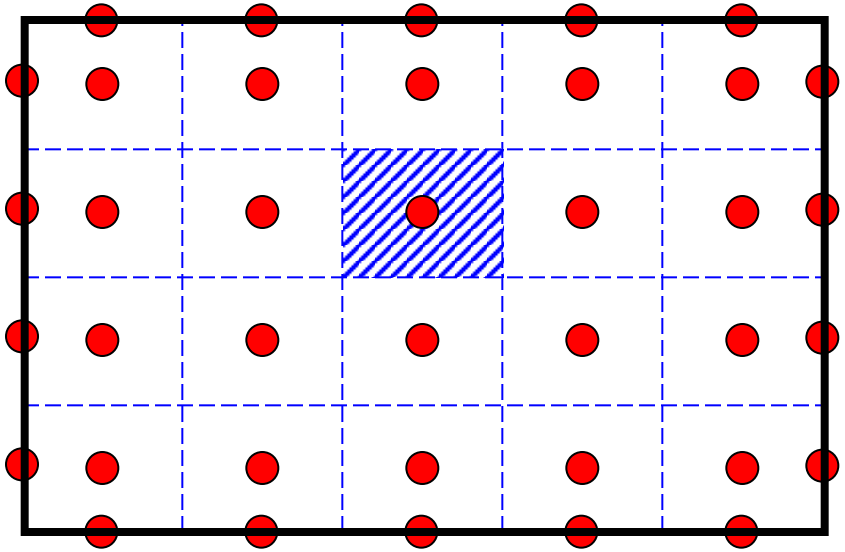


3．GRID—Grid generation

(1) **Select coordinate**: MODE＝1, 2, 3 stands for *x*-*y*, *r*-*x* and *r*-θ

(2) Set up length in x, y direction by XL, YL, respectively;

(3) Set up number of nodes in x and y directions by L1, M1, respectively;

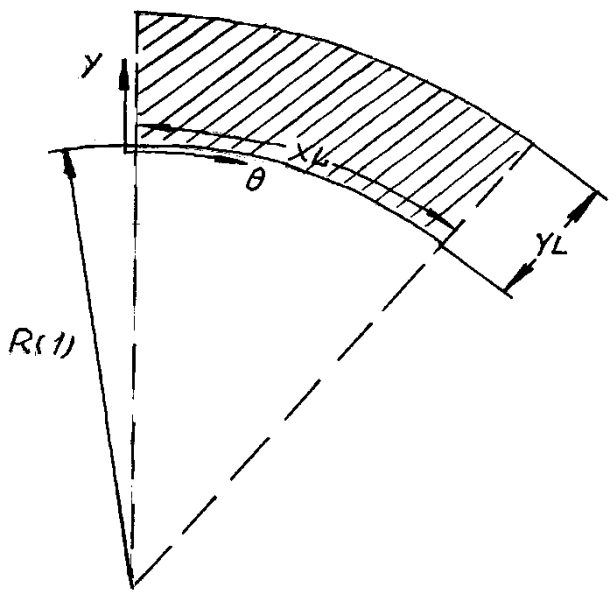


Practice B

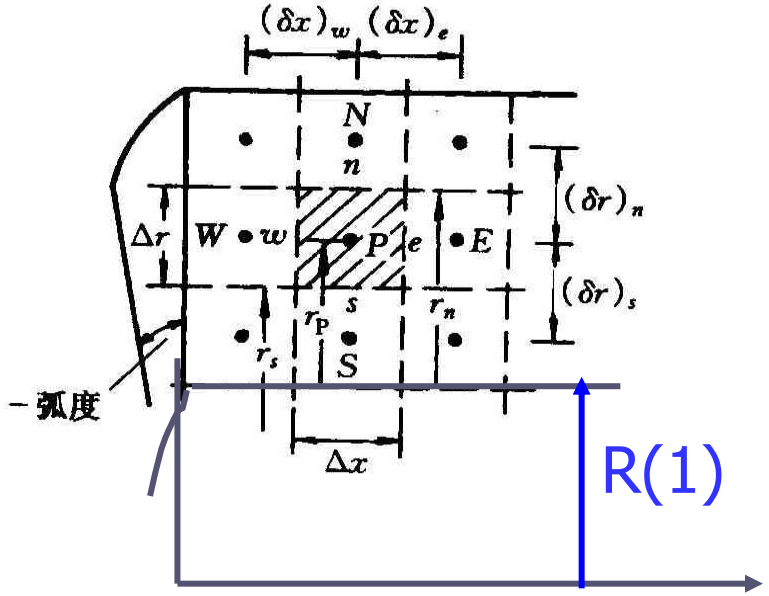
(4) Call UGRID to set up interface positions of the main CV in x,y direction, respectively, by

XU(i), YV(j), i=2, L1, j=2,M1 (Practice B);

(5) Set up the starting radius R(1) for MODE≠1



Polar coordinate



Cylindrical symmetric coordinate

4．UGRID

Generate interface positions according to pre-specified XL, YL and L1, M1.

5．SETUP 1

Main function: Set up 1-D arrays of geometric parameters which remain unchanged during iteration:

(1) Set up node positions by X(i), Y(j), i=1··L1, j=1··M1,

(2) Generate **width of main CV** by XCV(i), YCV(j),
i=2···L2, j=2···M2;

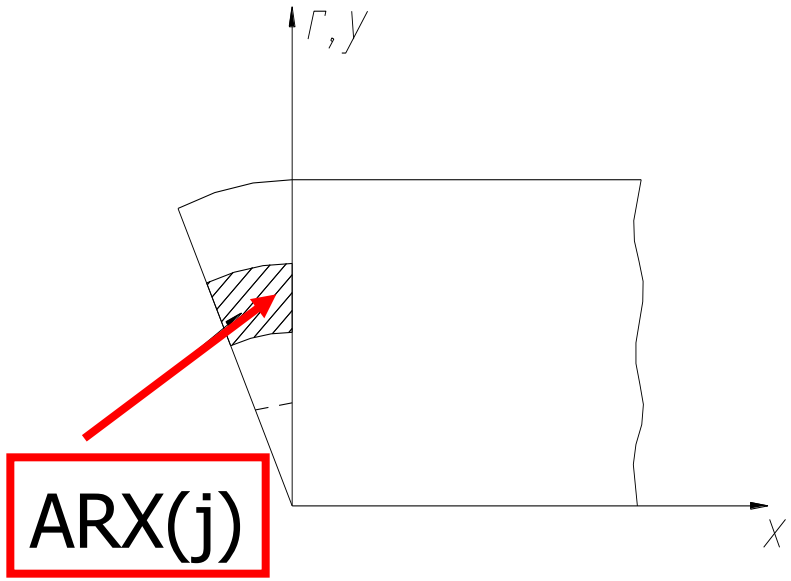
(3) Determine distance between two neighboring nodes
by XDIF(i), YDIF(j), i=2···L1, j=2···M1,
XDIF(i)=X(i)−X(i-1),
YDIF(j)=Y(j)−Y(j-1).

(4) Generate width of u, v CVs, respectively, by:

XCVS(i), i=3…L2, YCVS(j), j=3…M2;

(5) Set up radius R(j) in Y direction and scaling factor SX(j) in X direction;

(6) Calculate surface area normal to X direction: ARX(j)



(7) Set up XCVI(i), XCVIP(i), which correspond to $(\delta x)_{e^-}$, $(\delta x)_{e^+}$, respectively;

(8) Establish interpolation functions, such as FX(i), FXM(i), etc.

6．START

(1) Specify initial values for unsteady problems；

(2) Assume initial values of iteration for steady-state problems； Give boundary conditions which do not change during iteration.

Above four modules (GRID,UGRID,SETUP1,START) are executed only once during simulation of one case.

7．DENSE

Specify fluid density； For a constant-density problem, it can be empty(空块), but should be kept as is. For energy equation，

the definition of the nominal density is conducted in the Main Program. If the actual density is a function of temperature, it should be defined in this module.

8. BOUND

Set up boundary conditions for all variables.

9. OUTPUT

(1) For every outer iteration, output some representative results for observation of convergence;

(2) Compute some special 2nd quantities: h, q, Nu, f ,etc.;

(3) Call PRINT, output 2-D fields.

10．PRINT

Output simulation results.

11．SETUP2 （Key module of Main Program）

（1）Call **GAMSOR** to determine Γ_ϕ, S_P, S_C ;

（2）Call **DIFLOW** to determine $A(|P_\Delta|)$ for the scheme;

$$a_E = D_e A(|P_{\Delta e}|) + [\![-F_e, 0$$

（3）Set up the discretized coefficients denoted by

$AIP(I,J), AIM(I,J)\ AJM(i,j), AJP(i,j),\ AP(i,j)$

$CON(i,j)$---a_E, a_W, a_S, a_N, a_P and b in the lecture

（4）Call SOLVE to solve algebraic equations;

（5）Update indicator: ITER＝ITER+1.

12．GAMSOR

$$\frac{\partial(\rho^*\Phi)}{\partial t}+div(\rho^*\vec{u}\Phi)=div(\Gamma_\Phi grad\Phi)+S_\phi^*$$

(1) Determine Γ_ϕ for different variables:

$$u\ ,v-\mu(\text{or}\,\eta)\ ;T-\lambda$$

(2) Store source terms S_P, S_C of different variables into correspondent $AP(i,j)$,$CON(i,j)$, respectively.

(3) Set up **additional** source terms for boundary CVs with 2nd or 3rd condition

$CON(I,J)$ ← Original data + $CON(I,J)$ $S_{c,ad}$

$AP(I,J)$ ← Original data + $AP(I,J)$ $S_{p,ad}$

Add accumulated !

13．DIFLOW

$$a_E = D_e A(|P_{\Delta e}|) + [\![-F_e, 0$$

Determine $D_e A(|P_\Delta|)$ based on D and F.

14．SOLVE

Adopt ADI line iteration + block correction to solve algebraic equations;

Inner iteration is controlled by a variable NTIMES(NF), usually within 1 to 6;

Outer iteration is controlled by indicator ITER, which may reach $10^3 - 10^5$, depending on cases.

10.3.4 Functions and limitations of the code

Three functions (功能):

(1) It can solve the incompressible fluid flow and heat transfer problems in three 2D coordinates;

(2) It can solve 10 dependent variables consecutively (连续地) and print out 14 variables consecutively;

(3) It can solve both dimensional and dimensionless governing equations.

Limitations (限制)

(1) It is not convenient to simulate transient problems with high non-linearity;

(2) It is not convenient to simulate problems with irregular domain;

(3) It can not simulate compressible fluid flow.

10. 4 Grid System

10.4.1 Regulations for three coordinates

10.4.2 Numbering system for geometric parameters and variables

10.4.3 Composite picture of coordinates

10.4.4 Explanations of pressure field simulation

10. 4 Grid System

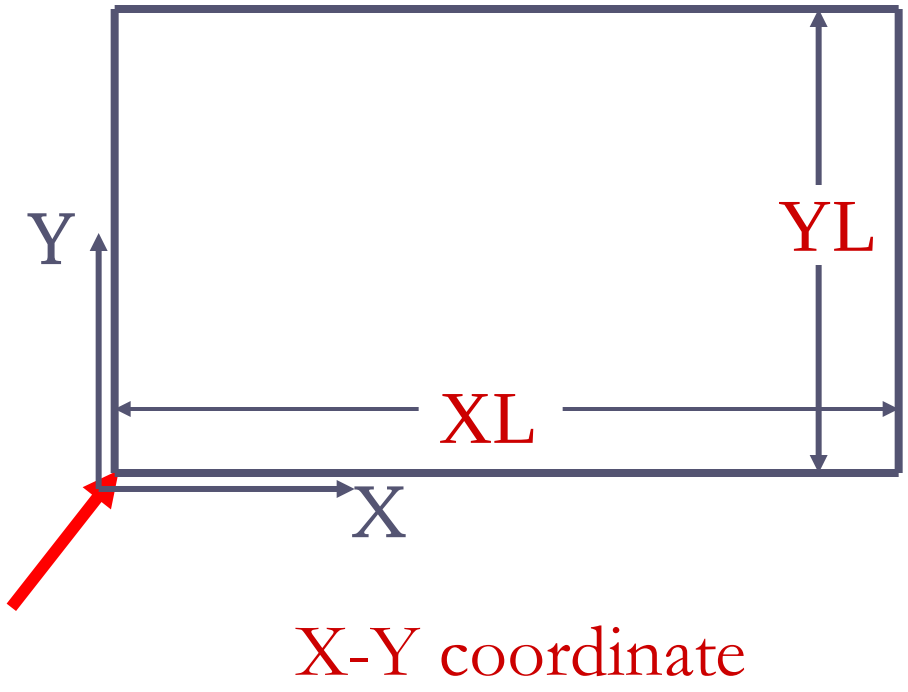
10.4.1 Regulations for three coordinates

1. Cartesian coordinate

(1) MODE＝1；

(2) Unit thickness in *Z*-direction；

(3) Origin of the coordinate locates in the left-bottom position.



X-Y coordinate

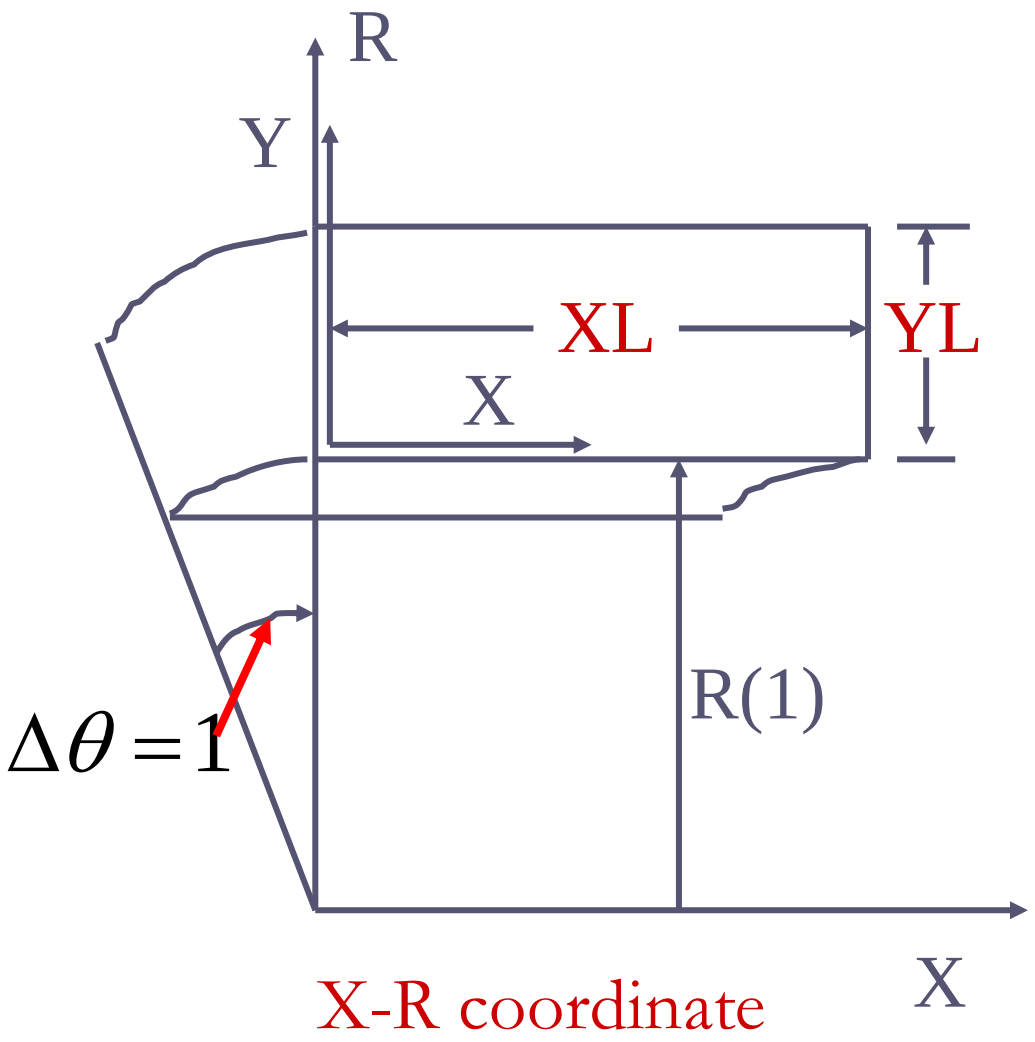
2. Cylindrical Symmetric coordinate

(1) MODE＝2;

(2) Simulation is conducted for $\Delta\theta = 1$ radian (弧度);

(3) R(j) starts from symmetric axis

(4) R(1) should be given.



3. Polar coordinate

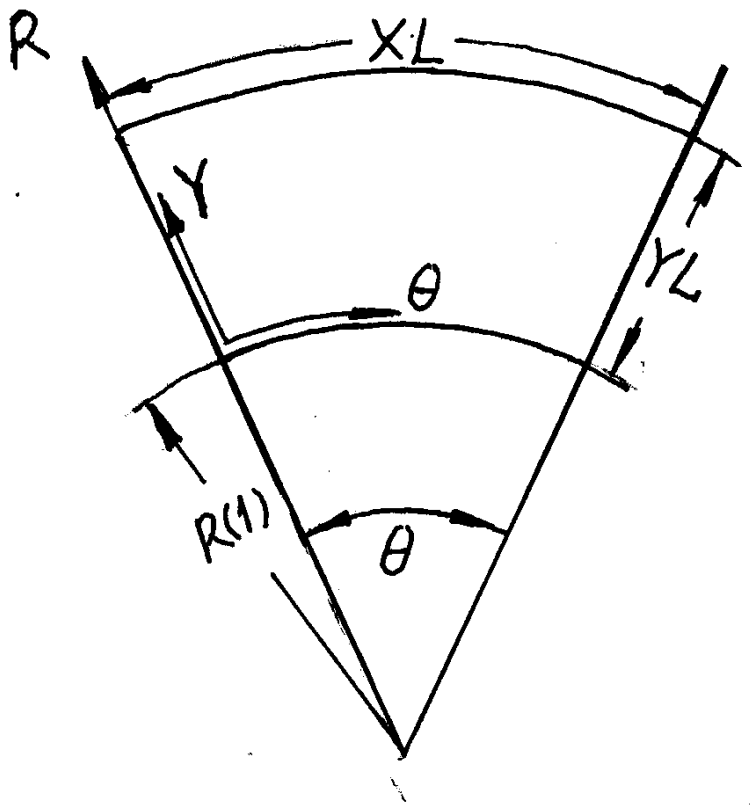
(1) MODE＝3;

(2) Unit thickness in *z*-direction;

(3) R(j) starts from circle center;

(4) R(1) should be given;

(5) Angle θ should be less than 2π



Theta-R coordinate

10.4.2 Numbering system for geometric parameters and variables

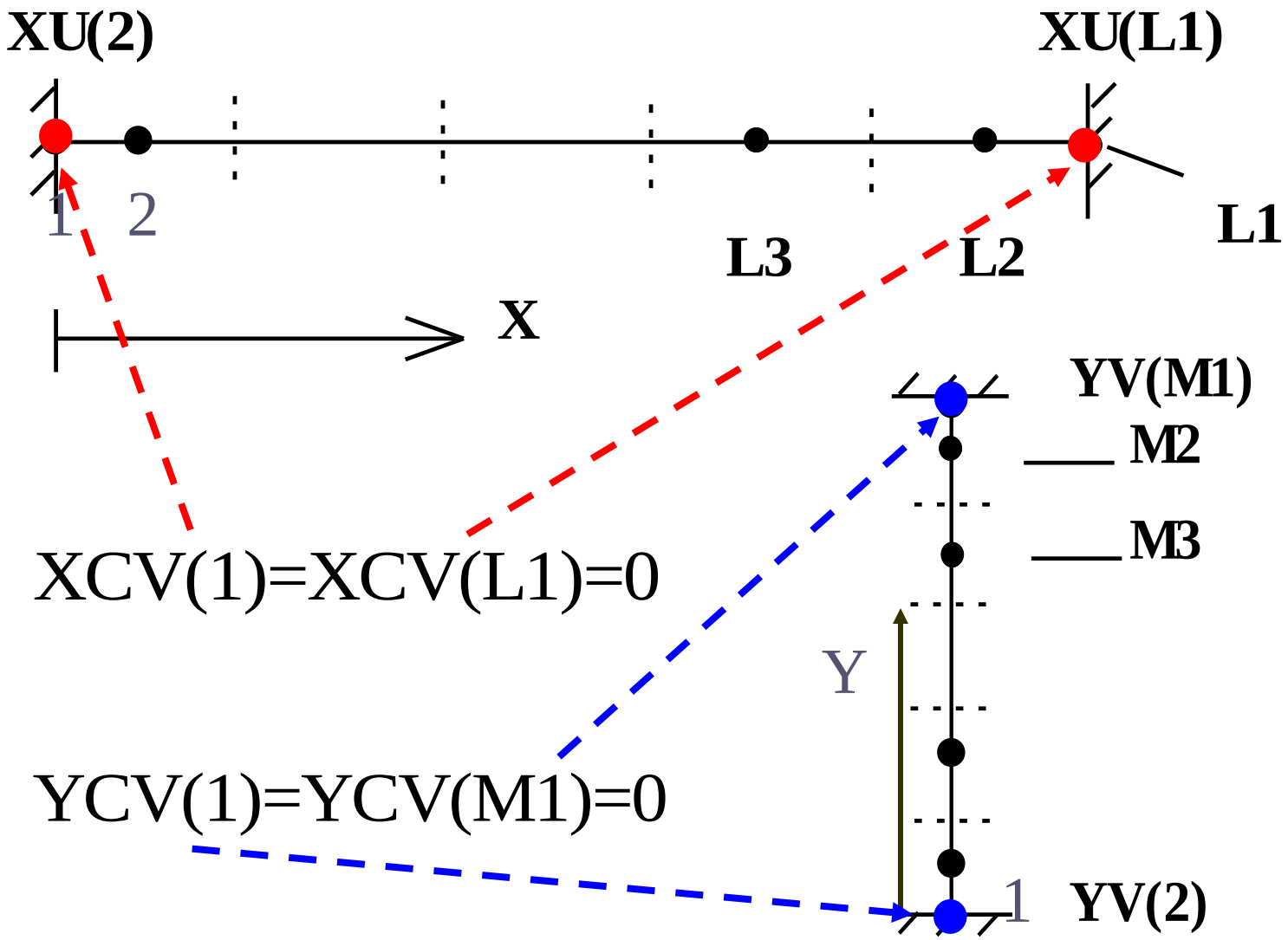
1.Interfaces of main CV : XU(i), i=2,……L1,

YV(j), j=2,……M1

2.Main nodes: Last three nodes in X-direction: L1, L2, L3; in Y-direction: M1, M2, M3

3.Width of main CV: XCV(i), i = 2,……L2;

YCV(j), j=2,……M2


XU(2)
XU(L1)
1
2
L3
L2
L1
X
YV(M1)
M2
M3
XCV(1)=XCV(L1)=0
Y
YCV(1)=YCV(M1)=0
1
YV(2)

4.Distances between nodes:

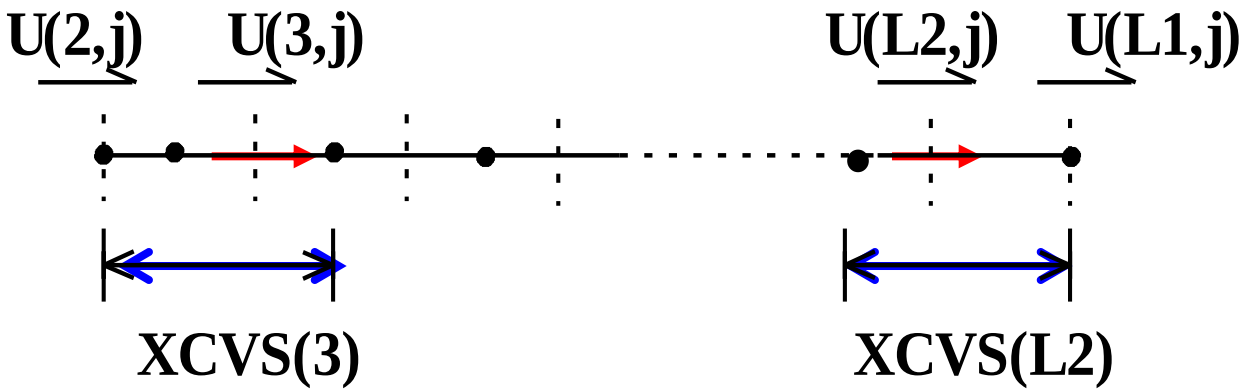
$$XDIF(i) = X(i) - X(i-1), \quad i = 2,.....L1$$

$$YDIF(j) = Y(j) - Y(j-1), \quad j = 2,.....M1$$

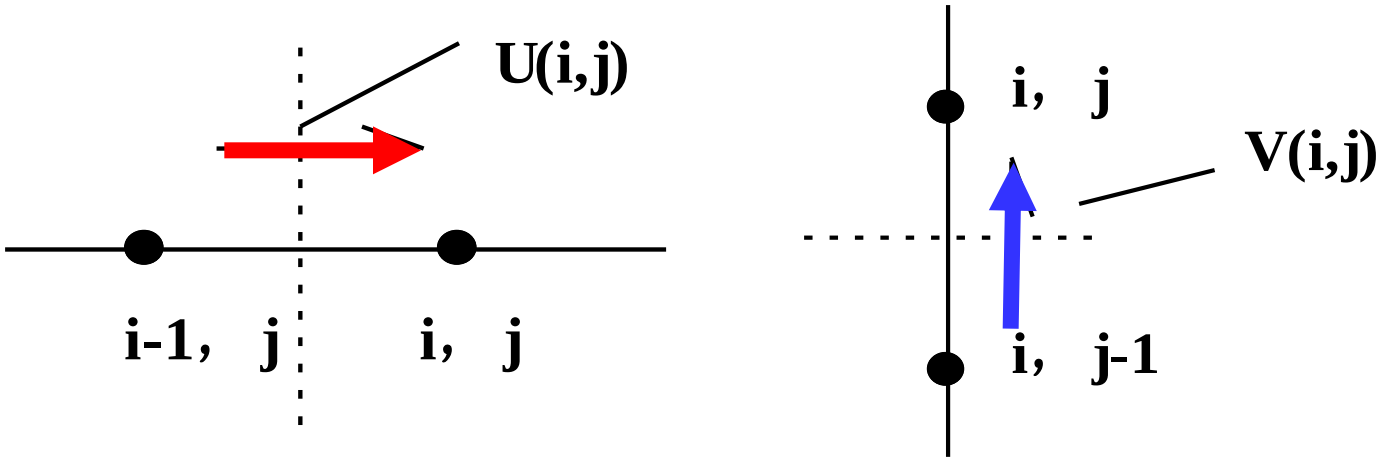
5.Widths of velocity CVs :

For component u: XCVS(i), i=3,……L2;

For component v: YCVS(j), j=3,……M2



6.Velocity numbering: the node number that was pointed by the velocity arrow is the number of velocity



7.Starting points of solution region of ABEqs.

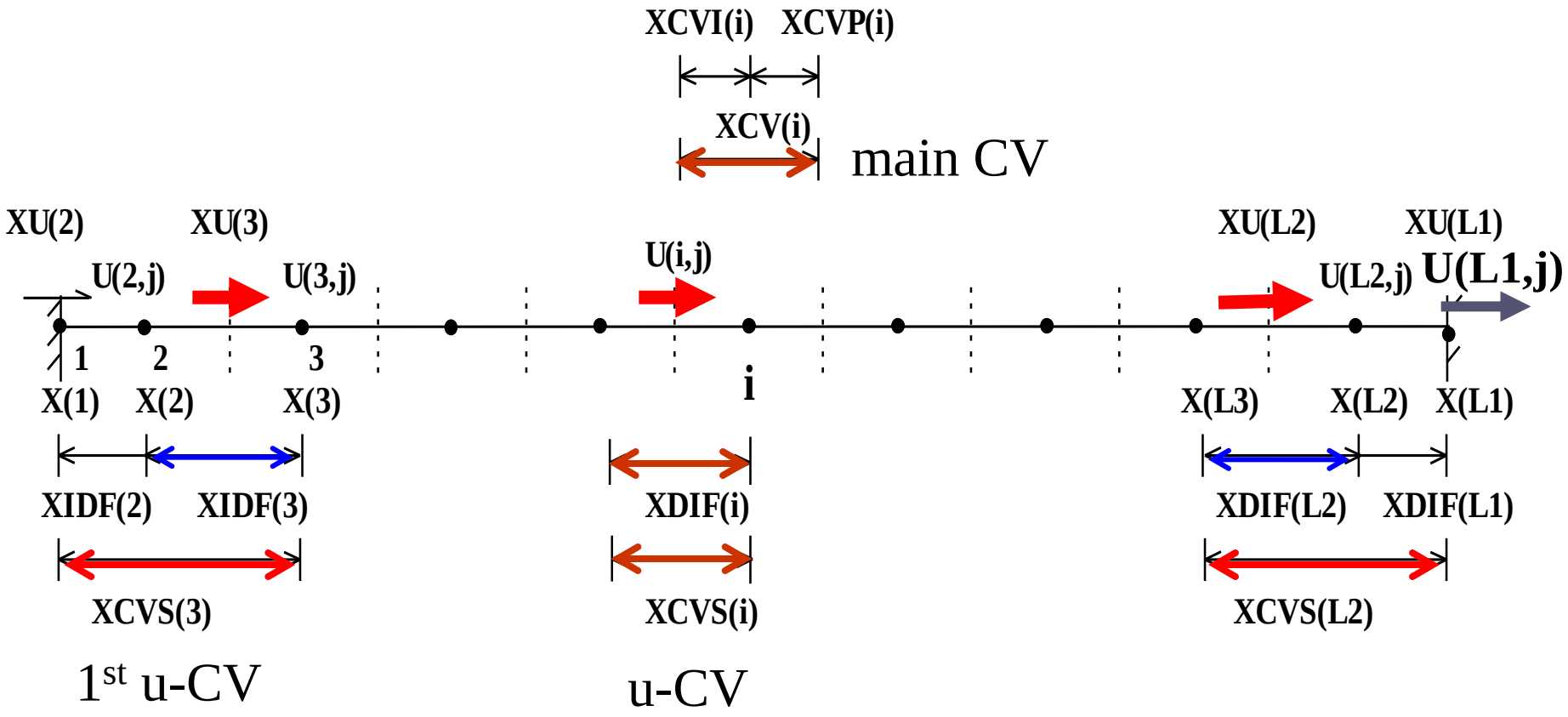
Because all boundaries are treated as 1st kind, solution is conducted within the inner region; **the starting nodes of solutions for *u*, *v*, *p* are different**, denoted in the code by FORTRAN variables IST, JST.

IST，JST represent the number of starting node in X,Y iteration, respectively:

Variable	IST	JST
ϕ, p, p'	2	2
u	3	2
v	2	3

10.4.3 Composite picture of coordinates

Composite figure in X-direction



Similarly in Y-direction.

10.4.4 Explanations of pressure field simulation

For incompressible flow, pressure value is relative to some reference**; For output purpose, a reference point is specified by (IPREF,JPREF) :**

$$p(i,j) = p(i,j) - p(IPREF, JPREF)$$

In the code, the default values are 1 for both IPREF and JPREF.

10. 5 Techniques Adopted in the Code

10.5.1 Ten dependent variables can be solved and 14 variables can be printed out

10.5.2 Iteration for nonlinear steady problem is treated the same as marching process of linear unsteady problem

10.5.3 Methods for saving memories

10.5.4 Methods for saving computational times

10. 5 Techniques Adopted in the Code

10.5.1 Ten dependent variables can be solved and 14 variables can be printed out

1. Define a simple variable NF, its maximum value is **10** (**NFMAX**) ; NF from 1 to 4 represents u, v, p' and T, respectively; NF≥5 can represent other variables defined by user.

```
MODULE START_L
 PARAMETER(NI=100,NJ=200,NIJ=NI,NFMAX=10,NFX4=NFMAX+4)
```

2. Define a 3-D array F(NI,NJ,NFX4), NFX4=14;

p, ρ, Γ, and c_p are regarded as the 11th, 12th , 13th and 14th variable, respectively.

```
REAL*8,DIMENSION(NI,NJ,NFX4)::F
```

3. Define two logic arrays： **LSOLVE(NF)** and **LPRINT(NF)**，and their default values are .FALSE.; In USER, if the value of LSOLVE(NF), say for NF＝1, is set as .T.， then in SETUP 2 this variable is solved.

```
DATA LSTOP,LSOLVE,LPRINT/.FALSE.,NFX4*.FALSE.,NFX4*.FALSE./
```

4. In SETUP2，Visit NF from 1 to NFMAX in order;

When some value of NF is visited and **LSOLVE(NF)=.T.， then this variable is solved**; Similarly in PRINT SUBROUTINE NF is visited from 1 to NFX4(=14) in order，as long as **LPRINT(NF)＝.T.**， the variable is printed out.

10.5.2 Iteration for nonlinear steady problem is treated the same as marching process of linear unsteady problem

Finishing one outer iteration for nonlinear steady problem (ITER=ITER+1) is equivalent to one time step forward for unsteady linear problem (Time=Time+DT).

1. ITER＋1 implies finishing one outer iteration，which is equivalent to Time=Time+DT， forward one time step.

```
TIME=TIME+DT
ITER=ITER+1
IF(ITER>=LAST) LSTOP=.TRUE.
RETURN
END
```

In order to guarantee the convergence of inner iteration for solving ABEqs., several cycles are included in one outer iteration and the cycle number is adjustable.

2. Set up an array, NTIMES(NF)，for indicating cycle number in one iteration. Its default value is one. If **NTIMES(2)＝4**，then algebraic equations of the second variable (NF=2) will be iteratively solved by four cycles.

```
DO 999 NT=1,NTIMES(NF)
N=NF
```

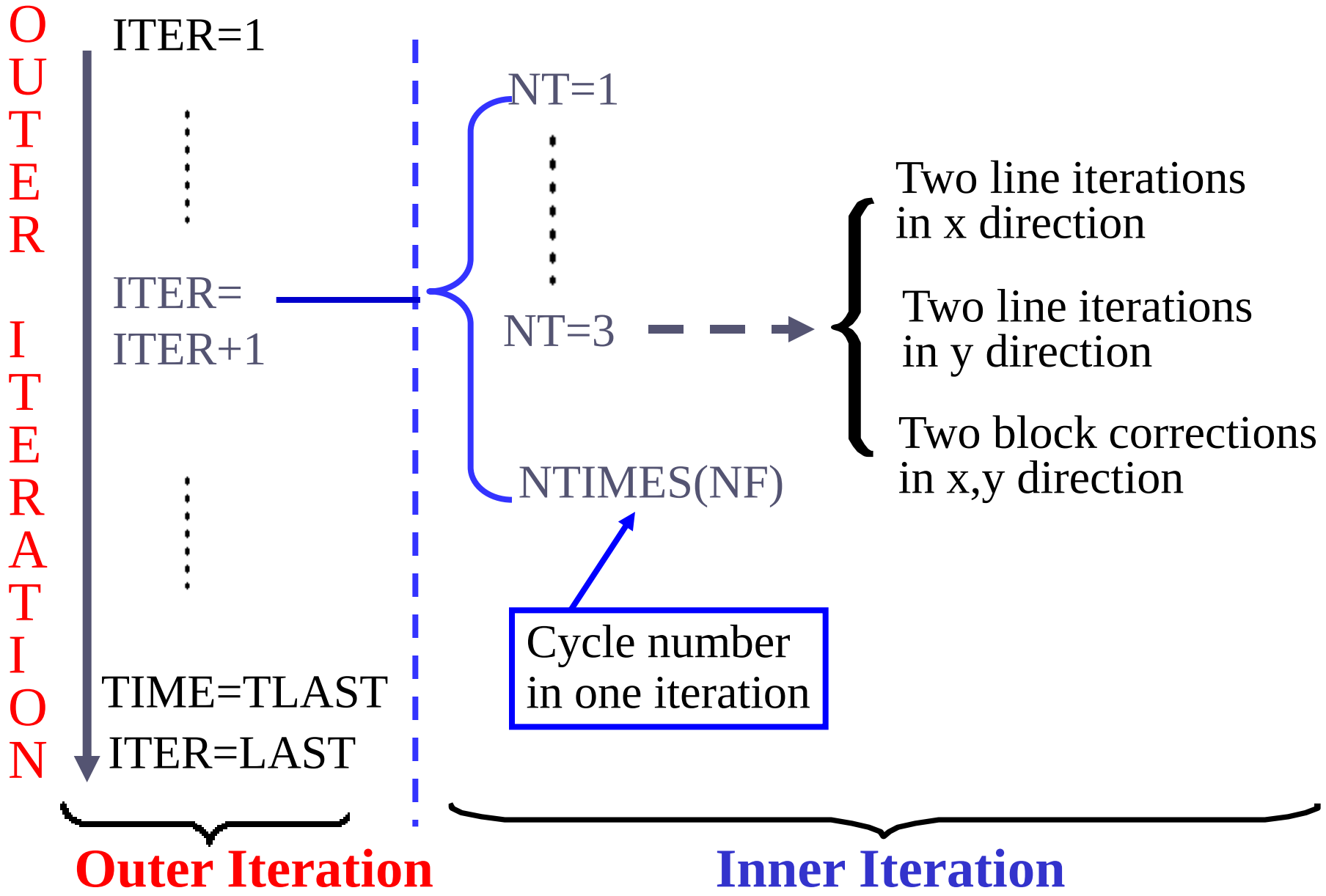
```
DATA RELAX,NTIMES/NFX4*1.,NFX4*1/
```

3. Every cycle includes 6 solution practices:

In X, Y-direction two block corrections;

In X-direction back-and forth (来回) line iterations;

In Y-direction, upward and downward line iterations .


OUTER ITERATION
ITER=1
ITER=
ITER+1
TIME=TLAST
ITER=LAST
NT=1
NT=3
NTIMES(NF)
Two line iterations
in x direction
Two line iterations
in y direction
Two block corrections
in x,y direction
Cycle number
in one iteration
Outer Iteration
Inner Iteration

4. Principles for selecting value of NTIMES(NF)

(1) Steady and nonlinear problems:

NTIMES(NF) takes values of 1～2, because the coefficients are to be further updated;

(2) Unsteady ad linear problems:

NTIMES(NF) may take a larger value, say 4～5, to ensure that for every outer iteration the solution of algebraic equations are converged.

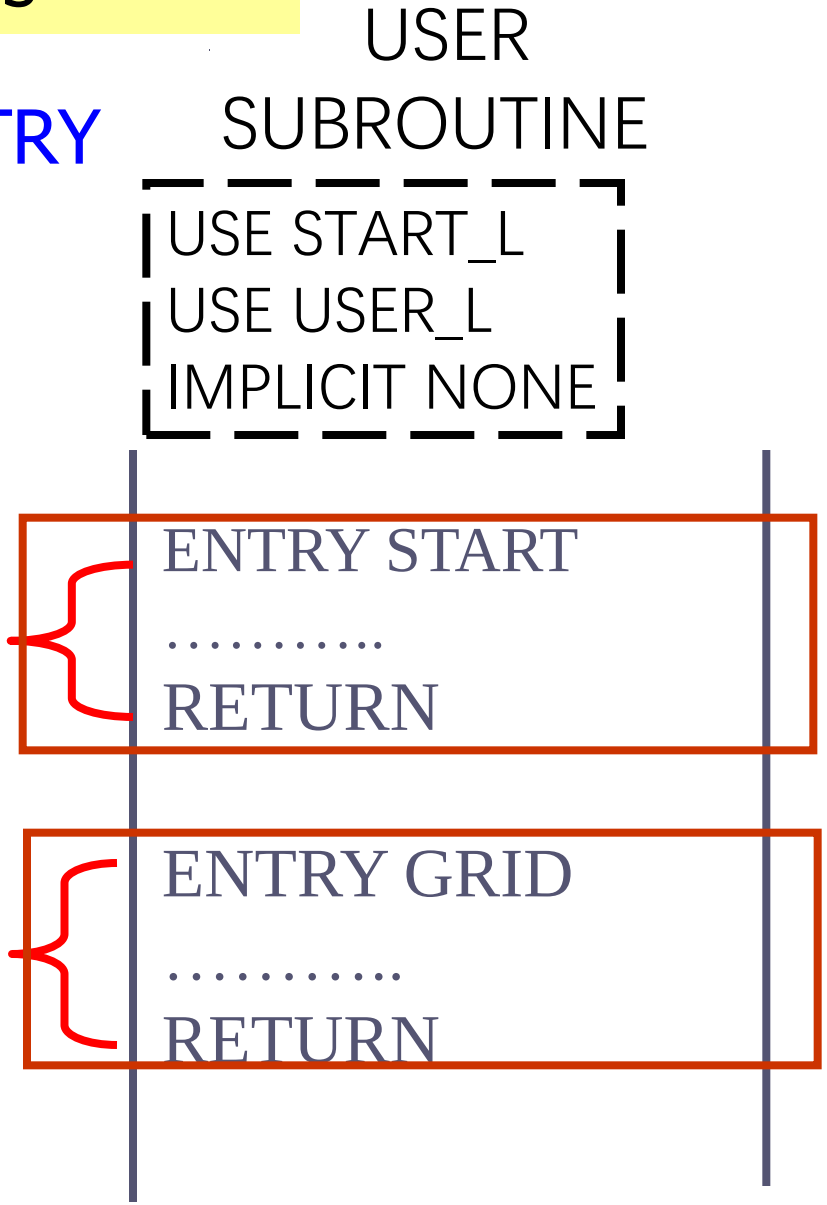
The present code is not recommended to apply for solving unsteady and nonlinear problems: For such problem, the coefficients within one time step should be updated, while in the present code within one time step coefficients remain unchanged. Then the time step has to be small enough.

10.5.3 Methods for saving memories

1.Adopt multiple inlet statement ENTRY

Function of ENTRY:

All ENTRYs within the same subroutine can share all MODULE located in the beginning part of the subroutine, and each ENTRY can be called individually.



2. A compromise is made between memory and computational time

All one-dimensional geometric parameters have their own individual array, totally 23 arrays, including $x(i)$ and $y(j)$;

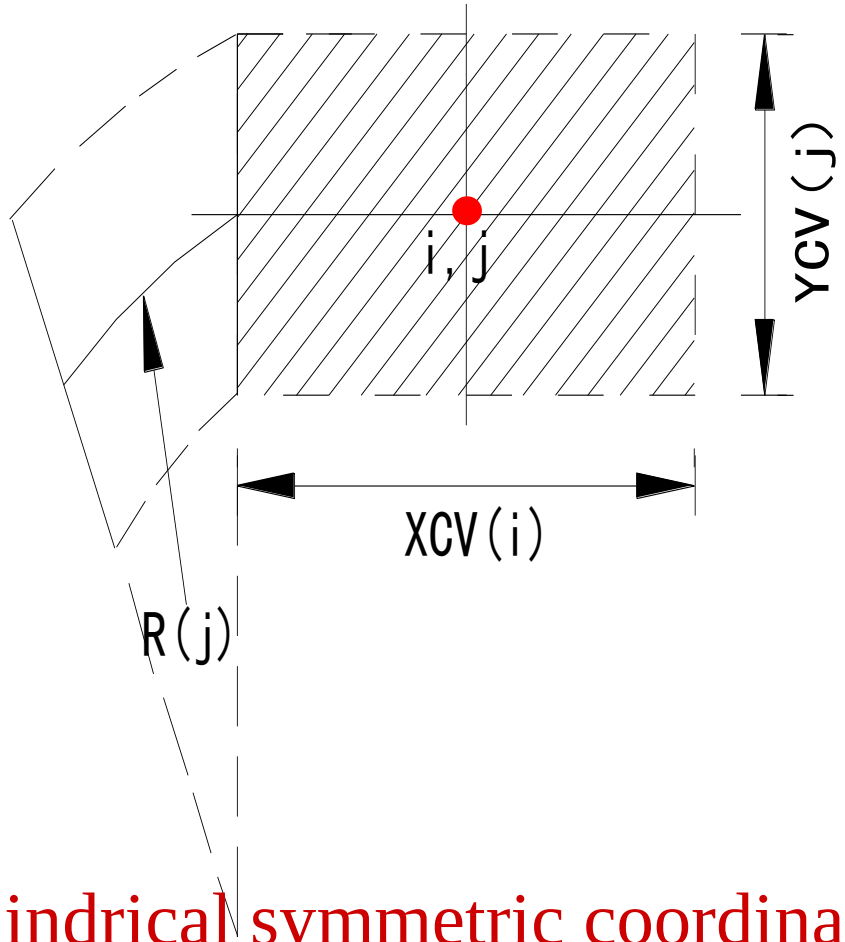
All two-dimensional parameters, including $\Delta V(i, j)$ are not stored: when needed they are temporary calculated, rather than stored.

Volume calculation of 3 CVs of cylindrical coordinate:

$$VOL = XCV(i) * YCV(j) * R(j) = XCV(i) * YCVR(j)$$

Area normal to flow direction

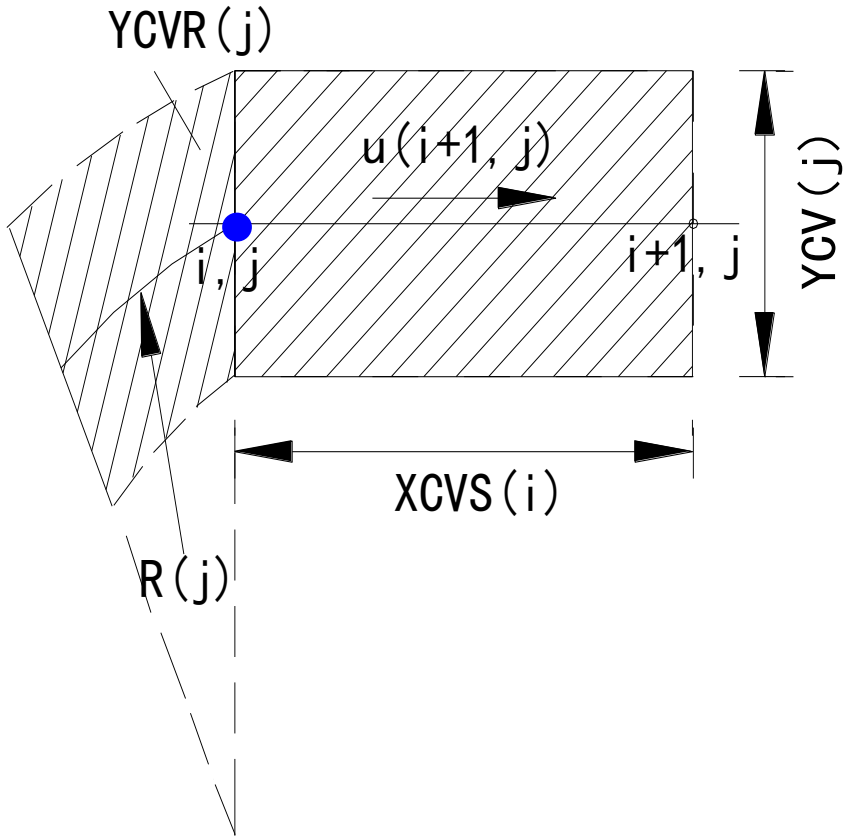
Main CV:

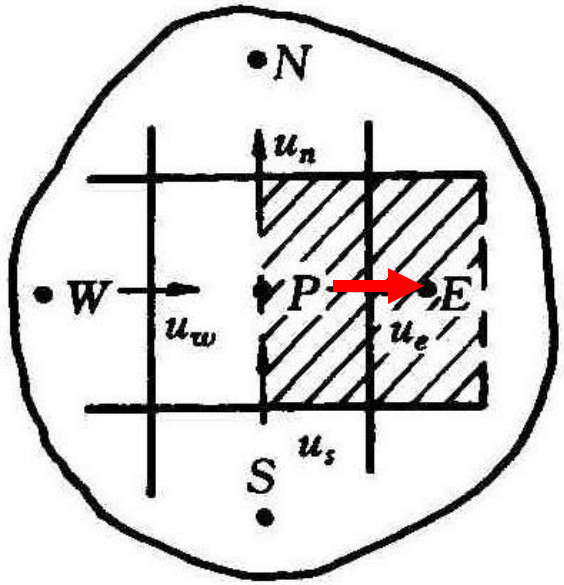


Cylindrical symmetric coordinate

u-CV:

$$VOL = XCVS(i) * YCV(j) * R(j) = XCVS(i) * YCVR(j)$$

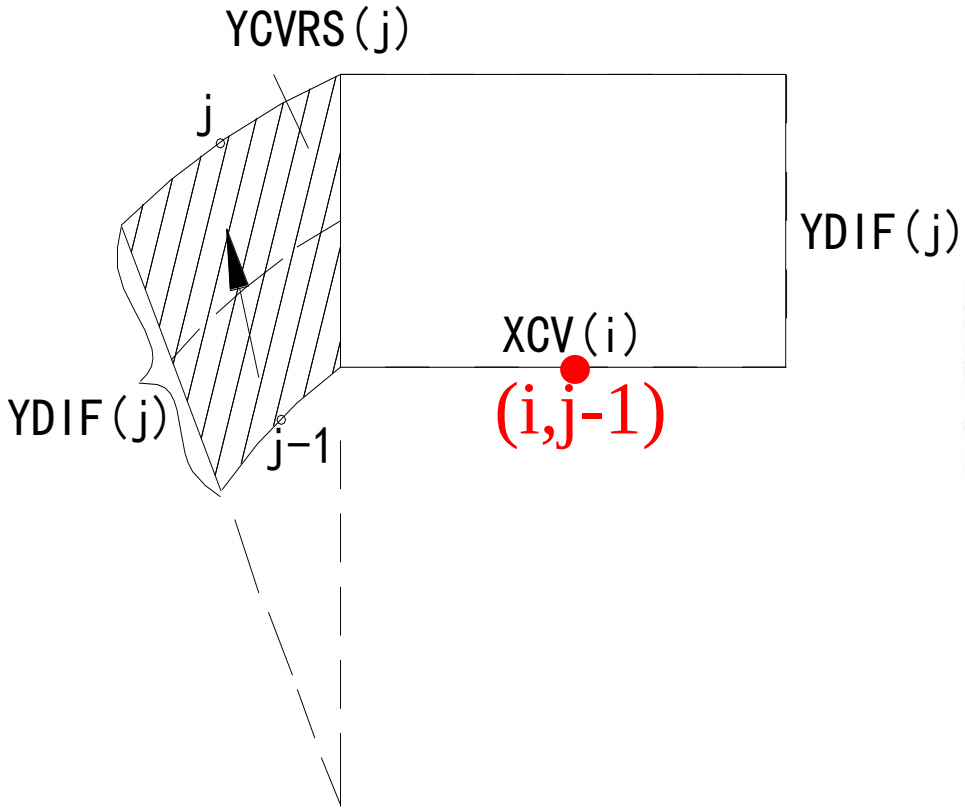


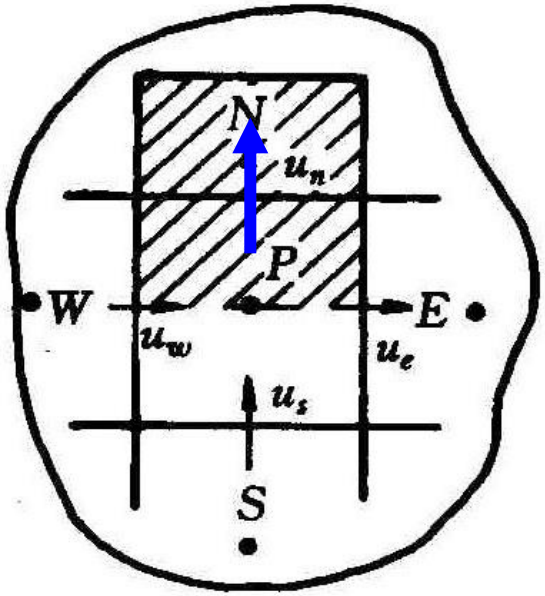


Cylindrical symmetric coordinate, u-CV

v-CV:

$$VOL = XCV(i) * YDIF(j) * \frac{R(j) + R(j-1)}{2} = XCV(i) * YCVRS(j)$$





Cylindrical symmetric coordinate, v-CV

3.Terminate outer iteration by specifying LAST, rather than by comparison of two subsequent iterations, thus saving one 3-D array. An appropriate value of LAST should be determined during iteration.

Nonlinear: high value of LAST

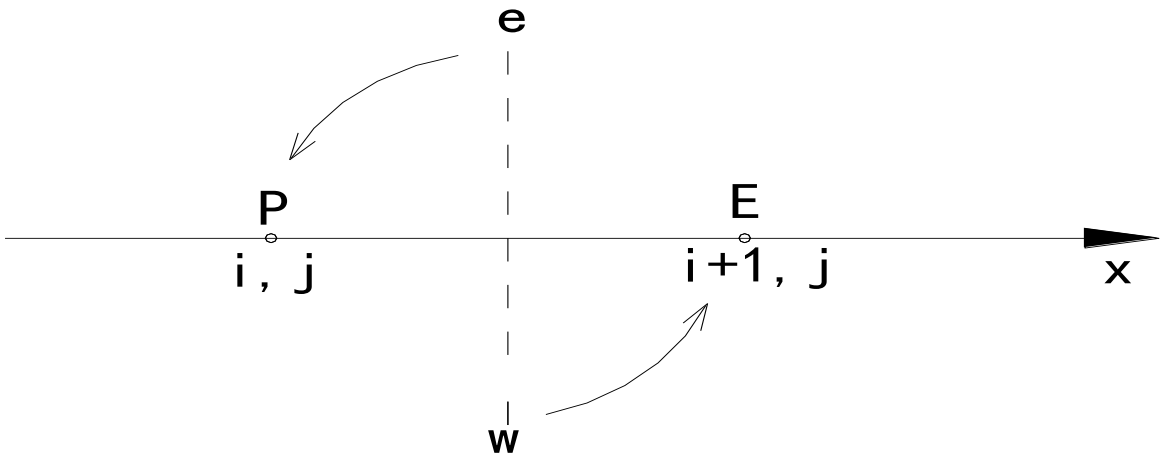
10.5.4 Methods for saving computational times

1.Data transfer between different units of the code by using MODULE. It is an efficient way for data transfer.

```
      SUBROUTINE DIFLOW
C*************************************
      USE START_L
      IMPLICIT NONE
```

2.Take advantage of relationship between coefficients, saving computational time .

It has been shown that for the five 3-point schemes:



$$a_E(i,j) = D_e A(|P_{\Delta e}|) + [|-F_e, 0|]$$

$$a_W(i+1,j) = D_w A(|P_{\Delta w}|) + [|F_w, 0|]$$

$$a_W(i+1,j) - a_E(i,j) = [|F_w, 0|] - [|-F_e, 0|] = F_e$$

Thus $a_E(i, j)$ can be easily obtained from $a_W(i+1, j)$

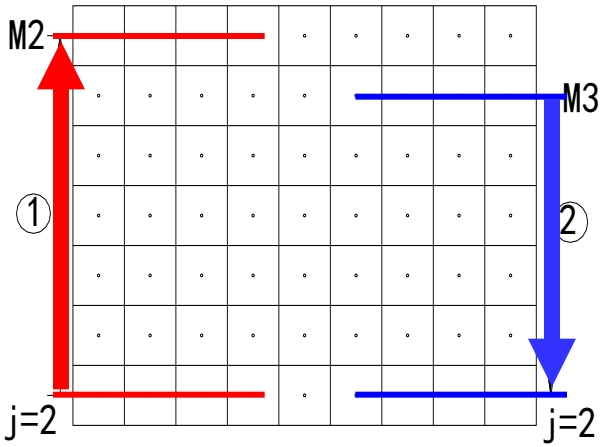
$$a_E(i, j) = a_W(i+1, j) - FLOW$$

Similarly:

$$a_N(i, j) = a_S(i, j+1) - FLOW$$

3.Time saving during ADI-line iteration

When scanning from bottom to top along y, J=2 to M2; then back scanning can start from M3 rather from M2, because the line of M2 is just solved in the upward scanning and all the coefficients and constants in that line remain unchanged.



本组网页地址：http://nht.xjtu.edu.cn 欢迎访问！

Teaching PPT will be loaded on our website

同舟共济
渡彼岸！

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