PRESSURE-VELOCITY COUPLING SCHEMES FOR BOUYANCY DRIVEN FLOW IN A DIFFERENTIALLY HEATED CAVITY USING F.V.M AND MATLAB

Purity Mberia, -Author

Department of, Meru University of Science and Technology, Kenya.

Stephen Karanja, -Supervisor

Department of Mathematics, Meru University of Science and Technology, Kenya.

Mark Kimathi, -Supervisor

Department of pure and applied mathematics, Machakos University. , Kenya.

Abstract

Numerical analysis of fluid flow is anchored on the laws of conservation. A challenge in solving the momentum equation arises due to the unavailability of an explicit pressure equation. To avoid solving the pressure term most researchers have eliminated it by cross differentiating the x and the y two dimensional momentum equations and subtracting them. This method introduces more variables to be solved in comparison to the primitive variables and is restricted to two-dimensional flows as streamlines do not exist in three-dimension. This method thus presents a serious limitation in analysis of fluid flow. In this study an equation for computing pressure has been developed using pressure - velocity coupling and used in solving the governing equations. The performance of three pressure velocity schemes namely; the Semi Implicit Method for Pressure linked Equation (SIMPLE), SIMPLE Revised (SIMPLER) and SIMPLE Consistent (SIMPLEC) for laminar buoyancy driven flow has been tested in order to establish the scheme that gives results consistent with bench mark data. The equations governing the flow are solved iteratively using finite volume method together with the central difference interpolating scheme. The solutions are presented for Rayleigh numbers of 10^3, 10^4, and 10^5. This resulted in the velocity profiles for the SIMPLE, SIMPLER, and SIMPLEC algorithm for a Rayleigh number of 10^4 and 10^5 converging to the same path. At

a Rayleigh number of 10^3 however, SIMPLER algorithm undergoes a degradation in convergence with grid refinement at the baffle region. Results predicted by using the SIMPLEC algorithm are thus able to effectively compute the velocity of fluid flow in a differentially heated square enclosure with baffles for both low and higher Rayleigh numbers irrespective of the grid size.

KEY WORDS

Buoyancy, Convection, Discretisation, Pressure-Velocity schemes, Laminar.Nomenclature

g	Gravitational acceleration, (m/s^2)
p	Pressure of the fluid, (N/m^2)
p^*	Pressure Dimensionless pressure
Ra	Rayleigh number
u,v	Velocity in the x and y directions
u^*,v^*	Dimensionless velocity component
v_{nb},u_{nb}	Neighbouring finite volumes
θ	Characteristic temperature
Pr	Prandtl number

Acronyms

SIMPLE	Semi Implicit Method for Pressure Linked Equations
SIMPLER	SIMPLE Revised
SIMPLEC	SIMPLE Consistent
FVM	Finite Volume Method

1.0 Background of the Study

Computation methods for buoyancy driven flow has many applications where heating and cooling takes place in an enclosure. Some of these applications are the drying and preservation of farm produce, and heat distribution in greenhouses. Temperature and velocity profiles of

these enclosures are important in the design of thermal structure and are obtained from the solution of the governing equations of fluid flow. Explicit equations for all the flow variables in the governing equations of flow are available except for pressure. To solve the momentum equation the pressure fields must be therefore determined first. The pressure-velocity schemes are used to derive an equation for computing pressure. The coupling between the momentum and the continuity equation is the key to the pressure- velocity coupling. These schemes are based on the finite volume discretization of the momentum equation. These schemes are an iterative process in which the discretized momentum equation is first solved in each finite volume using a guessed pressure field and the velocity field obtained is supposed to satisfy continuity equation.

The two assumptions made in the SIMPLE algorithm are that there is no interconnection between the initial pressure and initial velocity fields (Tao, (1) and that the neighbouring velocity component term is dropped to simplify the solution. The pressure is first estimated and computations made with the aim of improving the guessed pressure p* so that it will progressively get closer to satisfying the continuity equation. A pressure correction equation is obtained from the continuity equation and used to correct both the velocity and pressure. The corrected pressure is treated as a new guessed pressure P* and the whole procedure repeated until a converged solution is obtained.

SIMPLER is an acronym for Semi-Implicit Method for Pressure-Linked Equations Revised developed by Patankar, (2). This algorithm does not make the first assumption of the SIMPLE algorithm. An initial velocity estimate is made and used to compute a good pressure field from the continuity equation. The pressure correction equation of the SIMPLE is used to correct velocity but not pressure. SIMPLEC is an acronym for Semi-Implicit Method for Pressure Linked Equations Consistent developed by van Doormal and Rauthby (3). It avoids dropping velocity neighbour correction terms like the SIMPLE algorithm by retaining them and instead making an approximation.

Review of previously related studies;

Most researchers have used a certain pressure velocity schemes to derive an equation for pressure. S Jani et al, (4) studied fluid flow and natural convection heat transfer in a differentially heated square cavity with a fin attached to its cold wall. The governing equations were written in terms of the primitive variables and solved numerically using the finite volume method and the SIMPLER algorithm. The effects of the Rayleigh number, length of the fin and its position on the flow pattern and heat transfer inside the enclosure are investigated. The results show that for high Rayleigh numbers, a longer fin placing at the middle of the right wall has a more remarkable effect on the flow field and heat transfer inside the cavity.

Himsar et al, (5) carried out a numerical study on laminar natural convection heat transfer in a differentially heated air filled cavity with two insulated baffles attached to its horizontal walls. The SIMPLE algorithm was used to couple the velocity, pressure and temperature fields. It was observed that the two baffles trap some fluid in the cavity and affect the flow fields. The flow for cavities with low Rayleigh tend to circulate as a primary vortex strangled by the baffles while at high Rayleigh numbers it tends to separate into two different vortexes.

Bilgen et al (6), carried out a numerical study on laminar natural convection in enclosures with partial partitions. The two dimensional conservation equations were solved using the finite difference method. The pressure was deduced using the SIMPLER method. The aspect ratio was varied from 0.3 to 0.4, partition position changed from 0.5 to 0.6, and the Rayleigh number varied from 10^4 to 10^{11}. Isotherms and streamlines are produced for various Rayleigh

numbers. The results showed that heat transfer was reduced when two partitions were used instead of one, when the aspect ratio was made smaller and when the position of the partitions was further away from the hot wall.

Other researchers have compared the performance of various pressure velocity schemes.

Jang et al., (7) carried out a study on the performance of the PISO, SIMPLER and SIMPLEC algorithms for the treatment of the pressure-velocity coupling in steady flow problems by comparing the computational effort required to obtain the same level of convergence. SIMPLER and SIMPLEC exhibit better behaviour and reasonable solutions with the PISO algorithm are obtained only for small time steps. Clear superiority of SIMPLER over SIMPLEC or vice versa is not observed.

Hosseinzadeh et al., (8) solved the continuity and momentum equation using the SIMPLE and the SIMPLER algorithm for cavity flow at a Reynold number of 1000 and 3000. Results showed that SIMPLER algorithm gave a faster convergence than SIMPLE. Yin, R., et al., (9) studied the comparison of the SIMPLE, SIMPLER, SIMPLEC and PISO in computation of pressure in differentially heated enclosure. Results showed that the flow variables predicted by the four algorithms are the same though the pressure distributions are quite different. Two groupings of predicted pressure were arrived at, the SIMPLE, SIMPLEC and PISO resulted in one set and the SIMPLER resulted in another set.

From the above citations most studies have computed pressure using a certain pressure velocity scheme, other studies have compared the performance of various pressure velocity schemes, however no study has analysed the performance of SIMPLE, SIMPLER and SIMPLEC pressure-velocity scheme for laminar buoyancy driven flow in a differentially heated cavity with baffles. This study does this and the results obtained are then compared to M. Zeng et al., (10) benchmark results for validation.

2.0 Governing Equations

General equations are modelled to represent laminar buoyancy driven flows and the following assumptions made

i) The thermal properties such as density of the flow are assumed constant except for the buoyancy term which sustains the flow in the momentum equation.

ii) Buoyancy forces are experienced on the gravity direction only.

iii) The flow is steady.

iv) No slip condition is applied on the velocity at all four walls.

v) Heat transfer in the enclosure is by convection. Heat transfer by conduction is assumed negligible.

Vi) Heat dissipation is neglected.

vii) Flow is assumed to be in the x and y directions only

The non- dimensional equations for incompressible, steady, buoyancy driven flow are

2.0.1 Continuity equation

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

(1)

2.0.2 Momentum equation in x direction

$$u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{\partial P}{\partial x} + \Pr\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right)$$

(2)

2.0.3 Momentum equation in y direction

$$u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{\partial P}{\partial y} + \Pr\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) + Ra\Pr\theta$$

(3)

2.0.4 Energy equation

$$u\frac{\partial \theta}{\partial x} + v\frac{\partial \theta}{\partial y} = \left(\frac{\partial^2 \theta}{\partial x^2} + \frac{\partial^2 \theta}{\partial y^2}\right)$$

(4)

2.1 Discretised Equations

2.1.1 Continuity equation

Integrating the continuity equation over the control volume

$$\int_s^n \int_w^e \frac{\partial u}{\partial x} dxdy + \int_w^e \int_s^n \frac{\partial u}{\partial y} dydx$$

(5)

Gives

$$\left[\left(A_e u_e - A_w u_w\right) + \left(A_n v_n - A_s v_s\right)\right] = 0$$

(6)

2.1.2 Momentum equation in x direction

Integrating the momentum equation the x direction over the control volume

$$\int_s^n \int_w^e u\frac{\partial u}{\partial x} dxdy + \int_w^e \int_s^n v\frac{\partial u}{\partial y} dydx = \int_s^n \int_w^e \frac{\partial p}{\partial x} dxdy + \Pr\int_s^n \int_w^e \frac{\partial}{\partial x}\left(\frac{\partial u}{\partial x}\right) dxdy + \Pr\int_w^e \int_s^n \frac{\partial}{\partial y}\left(\frac{\partial u}{\partial y}\right) dydx$$

(7)

Results to,

$$a_P u_{i,J} = a_W u_{i-1,J} + a_E u_{i+1,J} + a_N u_{i,J+1} + a_S u_{i,J-1} + (P_{I-1,J} - P_{I,J})A_{i,J}$$

(8)

In standard form

$$a_P u_{i,J} = \sum a_{nb} u_{nb} + \left(P_{I-1,J} - P_{I,J}\right)A_{i,J} \tag{9}$$

2.1.3 Momentum equation in y direction

Integrating the y momentum equation over the control volume

$$\int_s^n \int_w^e u \frac{\partial v}{\partial x} dxdy + \int_w^e \int_s^n v \frac{\partial v}{\partial y} dydx = -\int_w^e \int_s^n \frac{\partial p}{\partial y} dydx + \Pr \int_s^n \int_w^e \frac{\partial^2 v}{\partial x^2} dxdy + \Pr \int_w^e \int_s^n \frac{\partial^2 v}{\partial y^2} dydx + Ra \Pr \int_w^e \int_s^n \theta dydx$$

(10)

Results in

$$a_P v_{I,j} = \sum a_{nb} v_{nb} + b_{I,j} + \left(P_{I,J-1} - P_{I,J}\right)A_{I,j}$$

(11)

The discretised momentum equations can now be solved iteratively using the SIMPLE, SIMPLER and SIMPLEC pressure-velocity scheme.

2.1.4 Discretised energy equation

Integration of the terms in the energy equation

$$\int_s^n \int_w^e u \frac{\partial \theta}{\partial x} dxdy + \int_w^e \int_s^n v \frac{\partial \theta}{\partial y} dydx = \int_s^n \int_w^e \frac{\partial^2 \theta}{\partial x^2} dxdy + \int_w^e \int_s^n \frac{\partial^2 \theta}{\partial y^2}$$

(12)

Yields

$$a_P \theta_{I,J} = a_E \theta_{I+1,J} + a_W \theta_{I-1,J} + a_N \theta_{I,J+1} + a_S \theta_{I,J-1}$$

(13)

2.2. Numerical Methodology

2.2.1 SIMPLE algorithm

Using an estimated pressure and velocity field momentum equations is solved and the

Correction formulae applied which results in

$$a_{i,J}u'_{i,J} = \sum a_{nb}u'_{nb} + (P'_{I-1,J} - P'_{I,J})A_{i,J} + b_{i,J}$$

(14)

$$a_{I,j}v'_{I,j} = \sum a_{nb}v'_{nb} + \left(P'_{I,J-1} - P'_{I,J}\right)A_{I,j} + b_{I,j}$$

(15)

Omitting the summation and further Substitution of the equations into the discretised continuity equation gives

$$\begin{aligned}&(A)_{i+1,J}\left(u^{*}_{i+1,J} + d_{i+1,J}\left(P^{/}_{I,J} \_ P^{/}_{I+1,J}\right)\right)\_(A)_{i,J}\left(u^{*}_{i,J} + d_{i,J}\left(P^{/}_{I-1,J} - P^{/}_{I,J}\right)\right)\\&+(A)_{I,j+1}\left(v^{*}_{I,j+1} + d_{I,j+1}\left(P^{/}_{I,J} - P^{/}_{I,J+1}\right)\right) - (A)_{I,j}\left(v^{*}_{I,j} + d_{I,j}\left(P^{/}_{I,J-1} - P^{/}_{I,J}\right)\right)\end{aligned}$$

(16)

Which is now expressed as

$$a_{I,J}P'_{I,J} = a_{I+1,J}P'_{I+1,J} + a_{I-1,J}P'_{I-1,J} + a_{I,J+1}P'_{I,J+1} + a_{I,J-1}P'_{I,J-1} + b'_{I,J}$$

(17)

The equation above is solved to obtain the pressure correction field at all points. The correct pressure field at all points P is calculated by adding P′ to P*. Velocities are calculated from their starred values using the velocity-correction equations. The corrected pressure P obtained is treated as a new guessed pressure P* and the procedure repeated until a converged solution is obtained. The discretization equation for other dependent variables such as temperature are then solved.

2.2.2 SIMPLER algorithm

The steps implemented in the SIMPLER algorithm are similar to those of the SIMPLE except that

i) The pressure is not guessed but is instead obtained by substitution of pseudal velocities of the discretised momentum equation into the discretised continuity equation which results to

$$\begin{aligned}&A_{i+1,J}\left(\hat{u}_{i+1,J} + d_{i+1,J}(P_{I,J} - P_{I+1,J}\right) - A_{i,J}\left(\hat{u}_{i,J} + d_{i,J}(P_{I-1,J} - P_{I,J}\right)\\&+A_{I,j+1}(\hat{v}_{I,j+1} + d_{I,j+1}(P_{I,J} - P_{I,J+1}) - A_{I,j}\left(\hat{v}_{I.j} + d_{I,j}(P_{I,J-1} - P_{I,J}\right)\end{aligned}$$

(18)

Which is now rearranged to give the discretised pressure equation below

$$a_{I,J}P_{I,J} = a_{I+1,J}P_{I+1,J} + a_{I-1,J}P_{I-1,J} + a_{I,J+1}P_{I,J+1} + a_{I,J-I}P_{I,J-1} + b_{I,J}$$

(19)

ii) The summation in the discretised momentum equation is not omitted.

2.2.3 SIMPLEC algorithm

The steps implemented in the SIMPLEC are similar to those of the SIMPLE except that the summation term $\sum a_{nb} u'_{nb}$ and $\sum a_{nb} v'_{nb}$ are not omitted but retained by instead making an approximation that,

$$\sum a_{nb} u'_{nb} \approx u'_{i,J} \sum a_{nb} \quad (20)$$

$$\sum a_{nb} v'_{nb} \approx v'_{I,j} \sum a_{nb} \quad (21)$$

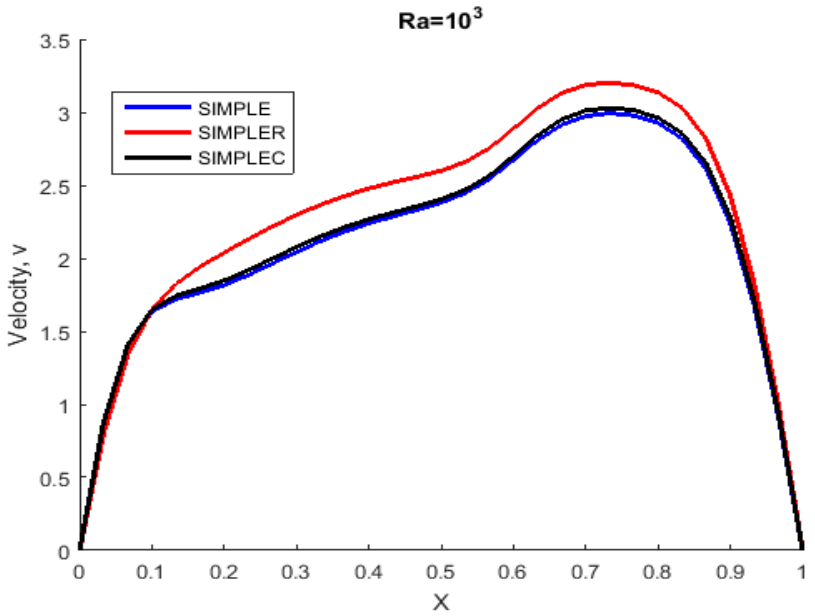
3. Results and Discussions.

Results on the effect of varying the pressure velocity schemes and the Rayleigh number on velocity have been presented by use of velocity profiles.

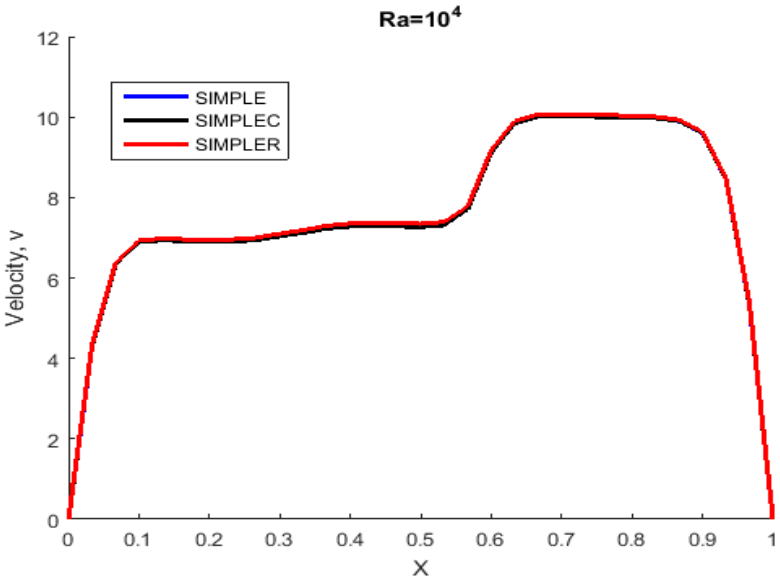
3.1 Effect of varying the Rayleigh number and pressure- velocity schemes on velocity profiles

The results of the primary velocity were negligible because buoyancy which is reason for flow of air in the cavity is assumed to be in the y-momentum equation.

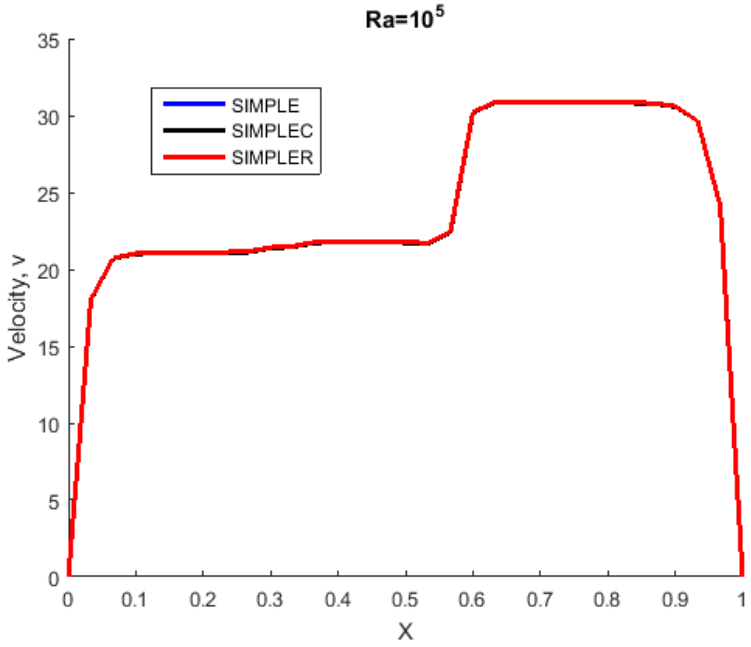
Fig 1 shows secondary velocities obtained for different Rayleigh numbers for the SIMPLE, SIMPLER and SIMPLEC algorithms.



(a) Rayleigh number of 10^3



(b) Rayleigh number of 10^4



(c) Rayleigh number of 10^5

Figure 1; Secondary velocities at a Rayleigh numbers of 10^5 for the SIMPLE, SIMPLER and SIMPLEC algorithm

The velocity of the fluid adjacent to the walls of the enclosure is zero for all values of the Rayleigh number due to adhesive forces holding these fluid particles to the stationary walls. As the layers of the fluid close to the hot wall gets heated up, there density reduces, hence rise up. The region they had occupied is now replaced by denser fluid layer. This recirculation of fluid leads to an increase in the kinetic energy of the fluid hence a rapid increase in the fluid velocity. At the baffle region, heat flow is inhibited hence the velocity remains constant for all the Rayleigh numbers. As the fluid layers acquire heat and rise past the baffle a rapid rise in velocity is seen. At the cold wall region the fluid velocity decreases rapidly to zero as the fluid loses heat to the cold trapped fluid. At a Rayleigh number of 10^3 the SIMPLER secondary velocity graph has different values of velocity within the baffle region as compared to that of the SIMPLE and SIMPLEC. At Rayleigh numbers of 10^4 and 10^5 the secondary velocities for the SIMPLE, SIMPLER and SIMPLEC converge to the same path.
At higher Rayleigh numbers of 10^4, 10^5 heat flow is through conduction and convection implying greater heat flow hence the reason for higher velocities.

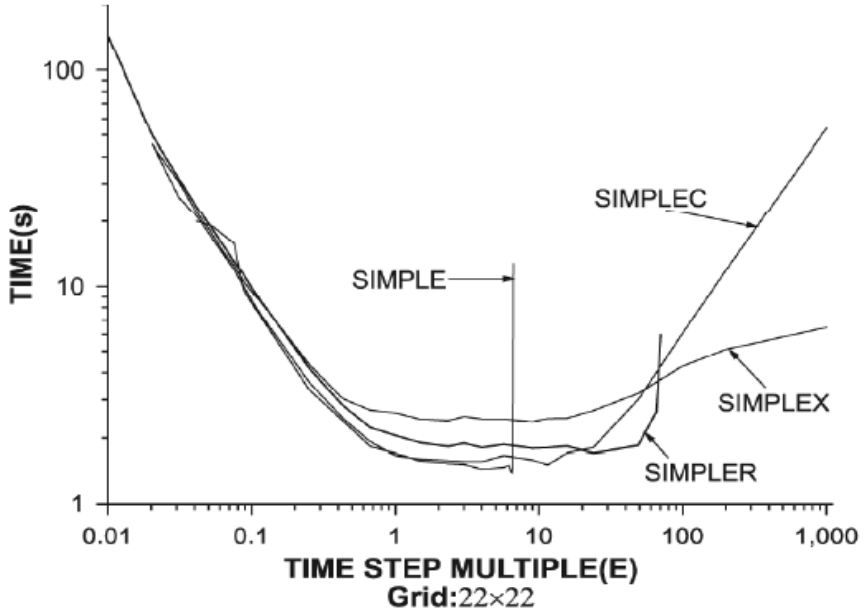
3.2 Comparison with results of previously published studies

Hosseinzadeh et al., (2011) solved the continuity and momentum equations using the SIMPLE and the SIMPLER pressure velocity schemes for fluid flow in an enclosure. It was evident that SIMPLER algorithm produced faster convergence than SIMPLE since it requires lesser iterations for convergence.

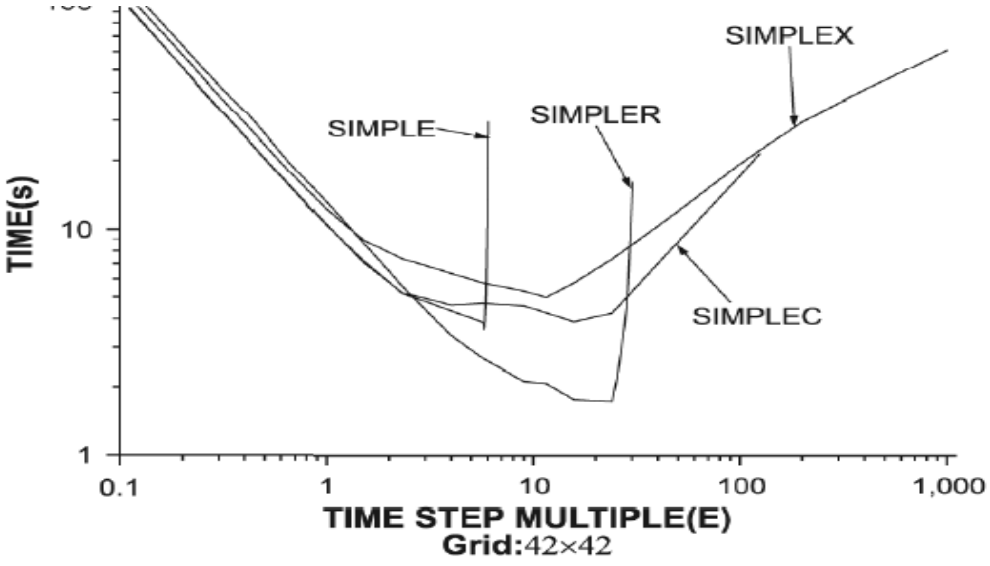
Yin, R., et al., (2003) carried out a study on comparison of the SIMPLE, SIMPLEC and PISO algorithm in computing pressure in a differentially heated cavity. It was found that the flow variables predicted by the four algorithms are the same though the pressure distributions are quite different. Two sets of predicted pressure were arrived at, the SIMPLE, SIMPLEC and PISO resulted in one set and the SIMPLER resulted in another set.

3.3 Comparison with the benchmark results

M. Zeng (2002), carried out a study on the convergence characteristics of the SIMPLE, SIMPLER, SIMPLEX and SIMPLEC algorithm for natural convection in a square enclosure at varying grid sizes. The figure below shows the SIMPLE, SIMPLER, SIMPLEX and SIMPLEC algorithm profiles at varying grid sizes for a Rayleigh number of 10^4



(a) 22 × 22 grid system



(b) 42 × 42 grid system

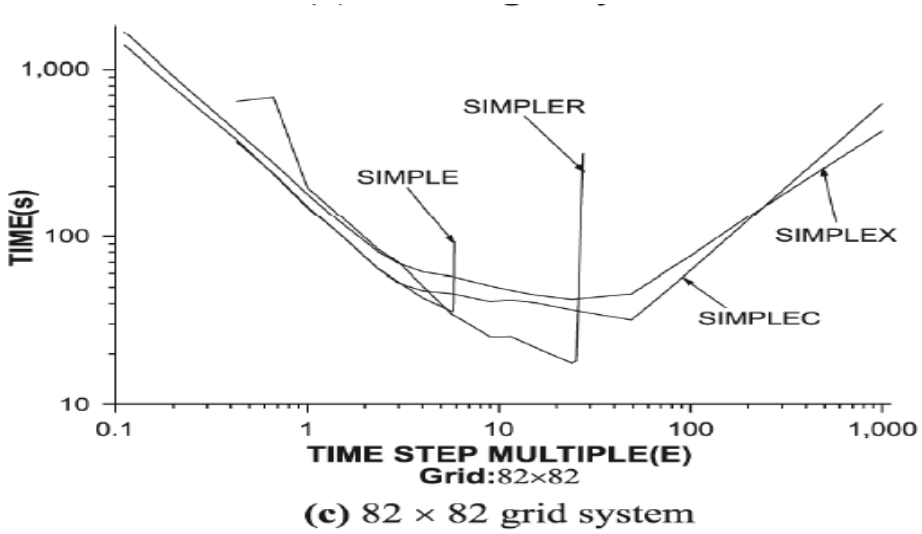


Figure 2; Robustness comparison of the SIMPLE, SIMPLER, SIMPLEX and SIMPLEC algorithm at a Rayleigh number of 10^4 for natural convection in a square differentially heated cavity by m.zeng (2002)

Fig 2 above shows that the SIMPLE and SIMPLER algorithm undergoes degradation with convergence as the grid is made finer. SIMPLEC and SIMPLEX are able to make computations at varying grid sizes.

In this study, at a Rayleigh number of 10^3, we observe that the SIMPLER velocity profiles at the baffle region give different values of velocity as compared to those of the SIMPLE and SIMPLEC algorithm. At higher Rayleigh numbers of 10^4, 10^5 the SIMPLE, SIMPLER SIMPLEC algorithms velocity profile converge to the same path. At the baffle region, velocity is constant for all Rayleigh numbers. An insulated baffle placed in a differentially heated enclosure hinders fluid flow and creates a fluid flow regime which exhibits characteristics of a finer grid in comparison to the regions not affected by baffles. Use of a bigger baffle would result in the SIMPLE algorithm also degrading with convergence as evidenced by results of M Zeng et al (2011)

4. Conclusion

Based on the above findings we conclude that for laminar buoyancy driven flow in a differentially heated cavity with baffles, SIMPLEC algorithm is recommended for pressure computation.

5. REFERENCES

1. Patankar, S. V. (1980). Numerical heat transfer and heat flow, series in computational methods in mechanics and thermal science

2. Galpin, P. F., & Raithby, G. D. (1986). Numerical solution of problems in incompressible fluid flow: treatment of the temperature-velocity coupling

3. Jani, S., Amini, M., & Mahmoodi, M. (2011). Numerical study of free convection heat transfer in a square cavity with a fin attached to its cold wall. *Heat Transfer Research, 42(3).*

4. Ambarita, H., Kishinami, K., Daimaruya, M., Saitoh, T., Takahashi, H., & Suzuki, J. (2006). Laminar natural convection heat transfer in an air filled square cavity with two insulated baffles attached to its horizontal walls. *Thermal science and engineering*, *14*(3), 35-46.

5. Bilgen, E. (2002). Natural convection in enclosures with partial partitions. *Renewable Energy*, 26 (2), 257-270.

6. Jang, D. S., Jetli, R., & Acharya, S. (1986). Comparison of the PISO, SIMPLER, and SIMPLEC algorithms for the treatment of the pressure-velocity coupling in steady flow problems. *Numerical Heat Transfer, Part A: Applications*, *10*(3), 209-228

7. Hosseinzadeh, S., Ostadhossein, R., Mirshahvalad, H. R., & Seraj, J. (2017). Using simpler algorithm for cavity flow problem. *Mechatronics and Applications*: *An International Journal* (*MECHATROJ), 1*(1).

8. Yin, R., & Chow, W. K. (2003). Comparison of four algorithms for solving pressure-velocity linked equations in simulating atrium fire. *International Journal on Architectural Science, 4(1), 24-35*.

9. Zeng, M., & Tao, W.Q. (2002). A comparison study of the convergence characteristic and robustness for four variants of SIMPLE-family at fine grid.

APPENDICES

MATLAB CODE

```
function pMberiaLE()
clear all; clc;
nx=31;ny=31;K=25;L=1;
x0=0;xend=1;dx=(xend-x0)/(nx-1);
y0=0;yend=L;dy=(yend-y0)/(ny-1);
x=x0:dx:xend;y=y0:dy:yend;
```

```
yloc=find(y>=0.2 & y<0.25);xloc=find(x<0.6);
yloc2=find(y>=0.7 & y<0.75);xloc2=find(x>0.3);
baffle=zeros(nx,ny);baffle(xloc,yloc)=1;baffle(xloc2,yloc2)=1;

us=zeros(nx,ny);vs=zeros(nx,ny);pp=zeros(nx,ny);
u=zeros(nx,ny);v=zeros(nx,ny);p=zeros(nx,ny);T=zeros(nx,ny);

uks=zeros(nx,ny,K);vks=zeros(nx,ny,K);pkp=zeros(nx,ny,K);pks=zeros(nx,ny,K);Tkm=zeros(
nx,ny,K);
Usrhs=zeros(nx,ny,K);Vsrhs=zeros(nx,ny,K);Pprhs=zeros(nx,ny,K);Trhs=zeros(nx,ny,K);
uk=zeros(nx,ny,K);vk=zeros(nx,ny,K);pk=zeros(nx,ny,K);Tk=zeros(nx,ny,K);
%==========INITIAL GUESS/prescribed boundary conditions==================
ps=0.5*ones(nx,ny);
us(1,1:ny)=0;vs(1,1:ny)=0;%T(1,1:ny)=1;%condition on left wall
us(nx,1:ny)=0;vs(nx,1:ny)=0;%T(nx,1:ny)=0;%condition on right wall
us(1:nx,1)=0;vs(1:nx,1)=0;T(1:nx,1)=1;%condition on bottom wall
us(1:nx,ny)=0;vs(1:nx,ny)=0;T(1:nx,ny)=0;%condition on top wall

Ae=dy;Aw=dy;An=dx;As=dx; De=Ae/dx;Dw=Aw/dx;Dn=An/dy;Ds=As/dy;
rho=1;Ra=10^5;Pr=0.71;tol=10^(-1);color='b';
for k=1:K
uks(:,:,1)=us(:,:);vks(:,:,1)=vs(:,:);pks(:,:,1)=ps(:,:);Tkm(:,:,1)=T(:,:);
uk(:,:,1)=u(:,:);vk(:,:,1)=v(:,:);Tk(:,:,1)=T(:,:);pkp(:,:,1)=pp(:,:);pk(:,:,1)=p(:,:);
 for i=2:length(x)-1
 for j=2:length(y)-1
if(j>=min(yloc))&& (j<=max(yloc))&&(i<=max(xloc))
uks(i,j,k)=0;uk(i,j,k)=0;vks(i,j,k)=0;vk(i,j,k)=0;
Tkm(i,j,k)=0;Tk(i,j,k)=0;pk(i,j,k)=0;
elseif(j>=min(yloc2))&& (j<=max(yloc2))&&(i>=min(xloc2))
uks(i,j,k)=0;uk(i,j,k)=0;vks(i,j,k)=0;vk(i,j,k)=0;
Tkm(i,j,k)=0;Tk(i,j,k)=0;pk(i,j,k)=0;
else
%==============STEP 1 OF SIMPLE ALGORITHM=====================
[aEu,aWu,aNu,aSu,aPu]=Ufluxes(uks(i-1,j,k),uks(i,j,k),uks(i+1,j,k),vks(i,j,k),vks(i-
1,j,k),vks(i,j+1,k),vks(i-1,j+1,k));
[aEv,aWv,aNv,aSv,aPv]=Vfluxes(vks(i,j-1,k),vks(i,j,k),vks(i,j+1,k),uks(i,j,k),uks(i,j-
1,k),uks(i+1,j,k),uks(i+1,j-1,k));

Usrhs(2:nx-1,2:ny-1,k)=uRHS(dx,dy,aNu,aSu,uks(2:nx-1,1:ny-2,k),uks(2:nx-
1,3:ny,k),pks(1:nx-2,2:ny-1,k),pks(2:nx-1,2:ny-1,k));
uCoeffMatstar=Utridmatrix(nx,aEu,aWu,aPu);[Qustar,Rustar]=lu(uCoeffMatstar);
uks(2:nx-1,j,k)=Rustar(2:nx-1,2:nx-1)\(Qustar(2:nx-1,2:nx-1)\Usrhs(2:nx-1,j,k));
Vsrhs(2:nx-1,2:ny-1,k)=vRHS(dx,dy,aEv,aWv,vks(1:nx-2,2:ny-1,k),vks(3:nx,2:ny-
1,k),pks(2:nx-1,1:ny-2,k),pks(2:nx-1,2:ny-1,k),Tk(2:nx-1,2:ny-1,k));
vCoeffMatstar=Vtridmatrix(ny,aNv,aSv,aPv);[Qvstar,Rvstar]=lu(vCoeffMatstar);
vks(i,2:ny-1,k)=Rvstar(2:ny-1,2:ny-1)\(Qvstar(2:ny-1,2:ny-1)\Vsrhs(i,2:ny-1,k)');%TO BE
VERIFIED
```

```
%=========STEP 2: SOLVE PRESSURE CORRECTION
EQUATION==================================
Pprhs(2:nx-1,2:ny-1,k)=pRHS(dx,dy,uks(2:nx-1,2:ny-1,k),uks(3:nx,2:ny-1,k),vks(2:nx-1,2:ny-
1,k),vks(2:nx-1,3:ny,k),pkp(2:nx-1,1:ny-2,k),pkp(2:nx-1,3:ny,k));
pCoeffMat=Ptridmatrix(ny,dx,dy);[Qpprime,Rpprime]=lu(pCoeffMat);
pkp(2:nx-1,j,k)=Rpprime(2:nx-1,2:nx-1)\(Qpprime(2:nx-1,2:nx-1)\Pprhs(2:nx-1,j,k));
%========STEP 3: CORRECT PRESSURE AND
VELOCITIES============================
pk(i,j,k)=pks(i,j,k)+pkp(i,j,k);
uk(i,j,k)=uks(i,j,k)+(Ae/(eps+aPu))*(pk(i-1,j,k)-pk(i,j,k));
vk(i,j,k)=vks(i,j,k)+(Aw/(eps+aPv))*(pk(i,j-1,k)-pk(i,j,k));

%========STEP 4: SOLVE OTHER TRANSPORT
EQUATIONS============================
[aEt,aWt,aNt,aSt,aPt]=Tfluxes(uk(i-1,j,k),uk(i,j,k),uk(i+1,j,k),vk(i,j,k),vk(i-
1,j,k),vk(i,j+1,k),vk(i-1,j+1,k));
Trhs(2:nx-1,2:ny-1,k)=tRHS(aNt,aSt,Tkm(2:nx-1,1:ny-2,k),Tkm(2:nx-1,3:ny,k));
tCoeffMat=Ttridmatrix(nx,aEt,aWt,aPt);[Qt,Rt]=lu(tCoeffMat);
Tkm(2:nx-1,j,k)=Rt(2:nx-1,2:nx-1)\(Qt(2:nx-1,2:nx-1)\Trhs(2:nx-1,j,k));
Tk(i,j,k)=Tkm(i,j,k);
Tk(1,j,k)=Tk(2,j,k);Tk(nx,j,k)=Tk(nx-1,j,k);
end
end
end
uks(1:nx,1:ny,k+1)=uks(1:nx,1:ny,k);vks(1:nx,1:ny,k+1)=vks(1:nx,1:ny,k);
pks(1:nx,1:ny,k+1)=pks(1:nx,1:ny,k);Tkm(1:nx,1:ny,k+1)=Tkm(1:nx,1:ny,k);

uk(1:nx,1:ny,k+1)=uk(1:nx,1:ny,k);vk(1:nx,1:ny,k+1)=vk(1:nx,1:ny,k);
pkp(1:nx,1:ny,k+1)=pkp(1:nx,1:ny,k);pk(1:nx,1:ny,k+1)=pk(1:nx,1:ny,k);
Tk(1:nx,1:ny,k+1)=Tk(1:nx,1:ny,k);
end
aWv;pkp(:,:,K);
figure(1)
% mesh(pk(:,:,K))
% title('pressure')
quiver(x(2:2:nx-1),y(2:2:ny-1),uk(2:2:nx-1,2:2:ny-1,K)',vk(2:2:nx-1,2:2:ny-1,K)',1)
axis([0 xend 0 yend])
xlabel('X');ylabel('Y');title('velocity vector plot')
figure(2)
mesh(x(2:nx-1),y(2:ny-1),uk(2:nx-1,2:ny-1,K)')
xlabel('X');ylabel('Y');title('u-velocity')
figure(3)
mesh(x(2:nx-1),y(2:ny-1),vk(2:nx-1,2:ny-1,K)')
xlabel('X');ylabel('Y');title('v-velocity')
figure(4)
mesh(x(1:nx-1),y(1:ny-1),Tk(1:nx-1,1:ny-1,K))
title('temperature');xlabel('Y');ylabel('X')
figure(5)
% surf(x(2:nx-1),y(2:ny-1),baffle(2:nx-1,2:ny-1)');shading flat;view(2);colormap(flipud(gray));
```

```
 hold on
[C,h] = contour(x(2:nx-1),y(2:ny-1),Tk(2:nx-1,2:ny-1,K)','Linewidth',1.2);
 %%clabel(C,'FontSize',10,'Color','k','Rotation',0)
 set(h,'ShowText','off','TextStep',get(h,'LevelStep'));%colormap(flipud(hsv));
 colormap(flipud(hsv(7)));colorbar('eastoutside')
hold off
xlabel('X');ylabel('Y');title('Isotherms:SIMPLE')
figure(6)
% surf(x(2:nx-1),y(2:ny-1),baffle(2:nx-1,2:ny-1)');shading flat;view(2);colormap(flipud(gray));
 hold on
[C,h] = contour(x(2:nx-1),y(2:ny-1),vk(2:nx-1,2:ny-1,K)','Linewidth',1.2);
 %%clabel(C,'FontSize',10,'Color','k','Rotation',0)
 set(h,'ShowText','off','TextStep',get(h,'LevelStep'));%colormap(flipud(hsv));
 colormap(cool(7));colorbar('eastoutside')
hold off
xlabel('X');ylabel('Y');title('v velocity:SIMPLE')

figure(7)
% surf(x(2:nx-1),y(2:ny-1),baffle(2:nx-1,2:ny-1)');shading flat;view(2);colormap(flipud(gray));
 hold on
[C,h] = contour(x(2:nx-1),y(2:ny-1),uk(2:nx-1,2:ny-1,K)','Linewidth',1);
 %%clabel(C,'FontSize',10,'Color','k','Rotation',0)
 set(h,'ShowText','on','TextStep',get(h,'LevelStep'));%colormap(flipud(hsv));
 colormap hot
hold off
xlabel('X');ylabel('Y');title('Velocity u')

figure(8)
hold on
plot(x(1:nx),uk(1:nx,floor(0.5*ny),K),color,'Linewidth',2)
xlabel('X')
ylabel('Velocity, u')
hold off
figure(9)
hold on
plot(x(1:nx),vk(1:nx,floor(0.5*ny),K),color,'Linewidth',2)
xlabel('X')
ylabel('Velocity, v')
hold off
figure(10)
hold on
plot(x(1:nx),Tk(1:nx,floor(0.5*ny),K),color,'Linewidth',2)
xlabel('X')
ylabel('Temperature, T')
hold off
function A=Utridmatrix(nx,aEu,aWu,aPu)
A=zeros(nx,nx);
A=diag(diag(aPu*ones(nx)),0)+diag(-aEu*ones(nx-1,1),1)+diag(-aWu*ones(nx-1,1),-
1);%creates ny*ny tridiagonal matrix
```

```
end
function Urhs=uRHS(dx,dy,aNu,aSu,uIl,uIr,plJ,pcJ)
upress=((plJ-pcJ)/dx)*dx*dy;
Urhs=aSu*uIl+aNu*uIr+upress;
end
function [aEu,aWu,aNu,aSu,aPu]=Ufluxes(ulJ,ucJ,urJ,vIc,vImc,vIr,vImr)
Feu=0.5*(rho*urJ+rho*ucJ);Fwu=0.5*(rho*ucJ+rho*ulJ);
Fsu=0.5*(rho*vIc+rho*vImc);Fnu=0.5*(rho*vIr+rho*vImr);
aEuu=-0.5*Ae*Feu+Pr*De; aWuu=0.5*Aw*Fwu+Pr*Dw;
aNuu=-0.5*An*Fnu+Pr*Dn; aSuu=0.5*As*Fsu+Pr*Ds;
aWu=max(Fwu,max(aWuu,0));aEu=max(-Feu,max(aEuu,0));
aSu=max(Fsu,max(aSuu,0));aNu=max(-Fnu,max(aNuu,0));
aPu=aWu+aEu+aSu+aNu;
end
function B=Vtridmatrix(ny,aNv,aSv,aPv)
B=zeros(ny,ny);
B=diag(diag(aPv*ones(ny)),0)+diag(-aNv*ones(ny-1,1),1)+diag(-aSv*ones(ny-1,1),-
1);%creates ny*ny tridiagonal matrix
end
function Vrhs=vRHS(dx,dy,aEv,aWv,vlJ,vrJ,pIl,pIc,TIc)
vpress=((pIl-pIc)/dy)*dx*dy;
sourceTerm=Pr*Ra*TIc*dx*dy;
Vrhs=aEv*vrJ+aWv*vlJ+vpress+1*sourceTerm;
end
function [aEv,aWv,aNv,aSv,aPv]=Vfluxes(vIl,vIc,vIr,uIc,uIl,uIpc,uIpl)
Fev=0.5*(rho*uIpc+rho*uIpl);Fwv=0.5*(rho*uIc+rho*uIl);
Fsv=0.5*(rho*vIl+rho*vIc);Fnv=0.5*(rho*vIc+rho*vIr);
aEvv=-0.5*Ae*Fev+Pr*De; aWvv=0.5*Aw*Fwv+Pr*Dw;
aNvv=-0.5*An*Fnv+Pr*Dn; aSvv=0.5*As*Fsv+Pr*Ds;
aWv=max(Fwv,max(aWvv,0));aEv=max(-Fev,max(aEvv,0));
aSv=max(Fsv,max(aSvv,0));aNv=max(-Fnv,max(aNvv,0));
aPv=aWv+aEv+aSv+aNv;
end
function C=Ptridmatrix(ny,dx,dy)
AimJ=dy;AipJ=dy;AIjp=dx;AIjm=dx;
aIJ=AimJ+AipJ+AIjp+AIjm;
C=zeros(ny,ny);
C=diag(diag(aIJ*ones(ny)),0)+diag(-AipJ*ones(ny-1,1),1)+diag(-AimJ*ones(ny-1,1),-
1);%creates ny*ny tridiagonal matrix
end
function Prhs=pRHS(dx,dy,ucJ,urJ,vIc,vIr,pIl,pIr)
AimJ=rho*dy;AipJ=rho*dy;AIjp=rho*dx;AIjm=rho*dx;
Prhs=AIjm*pIl+AIjp*pIr+1*(rho*AimJ*ucJ-rho*AipJ*urJ)+1*(rho*AIjm*vIc-rho*AIjp*vIr);
end
function D=Ttridmatrix(nx,aNt,aSt,aPt)
D=zeros(nx,nx);
D=diag(diag(aPt*ones(nx)),0)+diag(-aNt*ones(nx-1,1),1)+diag(-aSt*ones(nx-1,1),-1);%creates
ny*ny tridiagonal matrix
end
```

```
function Trhs=tRHS(aEt,aWt,TIl,TIr)
Trhs=aWt*TIl+aEt*TIr;
end
function [aEt,aWt,aNt,aSt,aPt]=Tfluxes(ulJ,ucJ,urJ,vIc,vImc,vIr,vImr)
Fet=0.5*(rho*urJ+rho*ucJ);Fwt=0.5*(rho*ucJ+rho*ulJ);
Fst=0.5*(rho*vIc+rho*vImc);Fnt=0.5*(rho*vIr+rho*vImr);
aEtt=-0.5*Ae*Fet+De; aWtt=0.5*Aw*Fwt+Dw;
aNtt=-0.5*An*Fnt+Dn; aStt=0.5*As*Fst+Ds;
aWt=max(Fwt,max(aWtt,0));aEt=max(-Fet,max(aEtt,0));
aSt=max(Fst,max(aStt,0));aNt=max(-Fnt,max(aNtt,0));
aPt=aWt+aEt+aSt+aNt;
end
end
```

SIMPLEC

```
function pMberiaLEC2()
clear all; clc;
nx=11;ny=11;K=5;L=1;
x0=0;xend=1;dx=(xend-x0)/(nx-1);
y0=0;yend=L;dy=(yend-y0)/(ny-1);
x=x0:dx:xend;y=y0:dy:yend;
yloc=find(y>=0.2 & y<0.25);xloc=find(x<0.6);
yloc2=find(y>=0.7 & y<0.75);xloc2=find(x>0.3);
baffle=zeros(nx,ny);baffle(xloc,yloc)=1;baffle(xloc2,yloc2)=1;
us=zeros(nx,ny);vs=zeros(nx,ny);pp=zeros(nx,ny);
u=zeros(nx,ny);v=zeros(nx,ny);p=zeros(nx,ny);T=zeros(nx,ny);
uks=zeros(nx,ny,K);vks=zeros(nx,ny,K);pkp=zeros(nx,ny,K);pks=zeros(nx,ny,K);Tkm=zeros(
nx,ny,K);
Usrhs=zeros(nx,ny,K);Vsrhs=zeros(nx,ny,K);Pprhs=zeros(nx,ny,K);Trhs=zeros(nx,ny,K);
uk=zeros(nx,ny,K);vk=zeros(nx,ny,K);pk=zeros(nx,ny,K);Tk=zeros(nx,ny,K);
%INITIAL GUESS/prescribed boundary conditions
ps=0.5*ones(nx,ny);
us(1,1:ny)=0;vs(1,1:ny)=0;%T(1,1:ny)=1;%condition on left wall
us(nx,1:ny)=0;vs(nx,1:ny)=0;%T(nx,1:ny)=0;%condition on right wall
us(1:nx,1)=0;vs(1:nx,1)=0;T(1:nx,1)=1;%condition on bottom wall
us(1:nx,ny)=0;vs(1:nx,ny)=0;T(1:nx,ny)=0;%condition on top wall
%INITIAL GUESS/prescribed boundary conditions
Ae=dy;Aw=dy;An=dx;As=dx; De=Ae/dx;Dw=Aw/dx;Dn=An/dy;Ds=As/dy;
rho=1;Ra=10^3;Pr=0.71;tol=10^(-1);color='b';simplec=1;
for k=1:K
uks(:,:,1)=us(:,:);vks(:,:,1)=vs(:,:);pks(:,:,1)=ps(:,:);Tkm(:,:,1)=T(:,:);
uk(:,:,1)=u(:,:);vk(:,:,1)=v(:,:);Tk(:,:,1)=T(:,:);pkp(:,:,1)=pp(:,:);pk(:,:,1)=p(:,:);
 for i=2:length(x)-1
for j=2:length(y)-1
```

```
if(j>=min(yloc))&& (j<=max(yloc))&&(i<=max(xloc))
uks(i,j,k)=0;uk(i,j,k)=0;vks(i,j,k)=0;vk(i,j,k)=0;
Tkm(i,j,k)=0;Tk(i,j,k)=0;pk(i,j,k)=0;
elseif(j>=min(yloc2))&& (j<=max(yloc2))&&(i>=min(xloc2))
uks(i,j,k)=0;uk(i,j,k)=0;vks(i,j,k)=0;vk(i,j,k)=0;
Tkm(i,j,k)=0;Tk(i,j,k)=0;pk(i,j,k)=0;
else
%==================START STEP 1 OF SIMPLEX
ALGORITHM=====================
[aEu,aWu,aNu,aSu,aPu,anbu]=Ufluxes(uks(i-1,j,k),uks(i,j,k),uks(i+1,j,k),vks(i,j,k),vks(i-
1,j,k),vks(i,j+1,k),vks(i-1,j+1,k));
[aEv,aWv,aNv,aSv,aPv,anbv]=Vfluxes(vks(i,j-1,k),vks(i,j,k),vks(i,j+1,k),uks(i,j,k),uks(i,j-
1,k),uks(i+1,j,k),uks(i+1,j-1,k));

Usrhs(2:nx-1,2:ny-1,k)=uRHS(dx,dy,aNu,aSu,uks(2:nx-1,1:ny-2,k),uks(2:nx-
1,3:ny,k),pks(1:nx-2,2:ny-1,k),pks(2:nx-1,2:ny-1,k),Tk(2:nx-1,2:ny-1,k));
uCoeffMatstar=Utridmatrix(nx,aEu,aWu,aPu);[Qustar,Rustar]=lu(uCoeffMatstar);
uks(2:nx-1,j,k)=Rustar(2:nx-1,2:nx-1)\(Qustar(2:nx-1,2:nx-1)\Usrhs(2:nx-1,j,k));
Vsrhs(2:nx-1,2:ny-1,k)=vRHS(dx,dy,aEv,aWv,vks(1:nx-2,2:ny-1,k),vks(3:nx,2:ny-
1,k),pks(2:nx-1,1:ny-2,k),pks(2:nx-1,2:ny-1,k),Tk(2:nx-1,2:ny-1,k));
vCoeffMatstar=Vtridmatrix(ny,aNv,aSv,aPv);[Qvstar,Rvstar]=lu(vCoeffMatstar);
vks(i,2:ny-1,k)=Rvstar(2:ny-1,2:ny-1)\(Qvstar(2:ny-1,2:ny-1)\Vsrhs(i,2:ny-1,k)');%TO BE
VERIFIED
Pprhs(2:nx-1,2:ny-1,k)=pRHS(dx,dy,uks(2:nx-1,2:ny-1,k),uks(3:nx,2:ny-1,k),vks(2:nx-1,2:ny-
1,k),vks(2:nx-1,3:ny,k),pkp(2:nx-1,1:ny-2,k),pkp(2:nx-1,3:ny,k));
pCoeffMat=Ptridmatrix(ny,dx,dy);[Qpprime,Rpprime]=lu(pCoeffMat);
pkp(2:nx-1,j,k)=Rpprime(2:nx-1,2:nx-1)\(Qpprime(2:nx-1,2:nx-1)\Pprhs(2:nx-1,j,k));
%=========END STEP 1 OF SIMPLEX
ALGORITHM=================================
%========START CORRECTED PRESSURE AND
VELOCITIES============================
pk(i,j,k)=pks(i,j,k)+pkp(i,j,k);
uk(i,j,k)=uks(i,j,k)+(Ae/(eps+(anbu)))*(pk(i-1,j,k)-pk(i,j,k));
vk(i,j,k)=vks(i,j,k)+(Aw/(eps+(anbv)))*(pk(i,j-1,k)-pk(i,j,k));
%========END CORRECTED PRESSURE AND
VELOCITIES============================
[aEt,aWt,aNt,aSt,aPt]=Tfluxes(uk(i-1,j,k),uk(i,j,k),uk(i+1,j,k),vk(i,j,k),vk(i-
1,j,k),vk(i,j+1,k),vk(i-1,j+1,k));
Trhs(2:nx-1,2:ny-1,k)=tRHS(aNt,aSt,Tkm(2:nx-1,1:ny-2,k),Tkm(2:nx-1,3:ny,k));
tCoeffMat=Ttridmatrix(nx,aEt,aWt,aPt);[Qt,Rt]=lu(tCoeffMat);
Tkm(2:nx-1,j,k)=Rt(2:nx-1,2:nx-1)\(Qt(2:nx-1,2:nx-1)\Trhs(2:nx-1,j,k));
Tk(i,j,k)=Tkm(i,j,k);
Tk(1,j,k)=Tk(2,j,k);Tk(nx,j,k)=Tk(nx-1,j,k);
end
end
end
uks(1:nx,1:ny,k+1)=uks(1:nx,1:ny,k);vks(1:nx,1:ny,k+1)=vks(1:nx,1:ny,k);
pks(1:nx,1:ny,k+1)=pks(1:nx,1:ny,k);Tkm(1:nx,1:ny,k+1)=Tkm(1:nx,1:ny,k);
```

```
uk(1:nx,1:ny,k+1)=uk(1:nx,1:ny,k);vk(1:nx,1:ny,k+1)=vk(1:nx,1:ny,k);pkp(1:nx,1:ny,k+1)=pk
p(1:nx,1:ny,k);
pk(1:nx,1:ny,k+1)=pk(1:nx,1:ny,k);Tk(1:nx,1:ny,k+1)=Tk(1:nx,1:ny,k);
end
aWv;pkp(:,:,K);
figure(1)
% mesh(pk(:,:,K))
% title('pressure')
quiver(x(2:2:nx-1),y(2:2:ny-1),uk(2:2:nx-1,2:2:ny-1,K)',vk(2:2:nx-1,2:2:ny-1,K)',1)
axis([0 xend 0 yend])
xlabel('X');ylabel('Y');title('velocity vector plot')
figure(2)
mesh(x(2:nx-1),y(2:ny-1),uk(2:nx-1,2:ny-1,K)')
xlabel('X');ylabel('Y');title('u-velocity')
figure(3)
mesh(x(2:nx-1),y(2:ny-1),vk(2:nx-1,2:ny-1,K)')
xlabel('X');ylabel('Y');title('v-velocity')
figure(4)
mesh(x(1:nx-1),y(1:ny-1),Tk(1:nx-1,1:ny-1,K))
title('temperature');xlabel('Y');ylabel('X')
figure(5)
% surf(x(2:nx-1),y(2:ny-1),baffle(2:nx-1,2:ny-1)');shading flat;view(2);colormap(flipud(gray));
 hold on
[C,h] = contour(x(2:nx-1),y(2:ny-1),Tk(2:nx-1,2:ny-1,K)','Linewidth',1.5);
 %%clabel(C,'FontSize',10,'Color','k','Rotation',0)
 set(h,'ShowText','on','TextStep',get(h,'LevelStep'));%colormap(flipud(hsv));
 colormap cool
hold off
xlabel('X');ylabel('Y');title('Isotherms')
figure(6)
% surf(x(2:nx-1),y(2:ny-1),baffle(2:nx-1,2:ny-1)');shading flat;view(2);colormap(flipud(gray));
 hold on
[C,h] = contour(x(2:nx-1),y(2:ny-1),vk(2:nx-1,2:ny-1,K)','Linewidth',1.5);
 %%clabel(C,'FontSize',10,'Color','k','Rotation',0)
 set(h,'ShowText','on','TextStep',get(h,'LevelStep'));%colormap(flipud(hsv));
 colormap hot
hold off
xlabel('X');ylabel('Y');title('velocity v')
figure(7)
hold on
plot(y(1:ny),uk(floor(0.5*nx),1:ny,K),color,'Linewidth',2)
xlabel('Y')
ylabel('Velocity, u')
hold off
figure(8)
hold on
plot(x(1:nx),vk(1:nx,floor(0.25*ny),K),color,'Linewidth',2)
xlabel('X')
ylabel('Velocity, v')
```

```
hold off

function A=Utridmatrix(nx,aEu,aWu,aPu)
A=zeros(nx,nx);
%A(1,1)=aPu;A(1,2)=-aEu;A(nx,nx-1)=-aWu;A(nx,nx)=aPu;%inputs the 1st and last rows of
matrix A
A=diag(diag(aPu*ones(nx)),0)+diag(-aEu*ones(nx-1,1),1)+diag(-aWu*ones(nx-1,1),-
1);%creates ny*ny tridiagonal matrix
%A(2:nx-1,1:nx)=Ain(2:nx-1,1:nx);% specified the inner tridiagonal matrix of matrix A
end
function Urhs=uRHS(dx,dy,aNu,aSu,uIl,uIr,plJ,pcJ,TIc)
upress=((plJ-pcJ)/dx)*dx*dy;
sourceTerm=Pr*Ra*TIc*dx*dy;
Urhs=aSu*uIl+aNu*uIr+upress+0*sourceTerm;
end
function [aEu,aWu,aNu,aSu,aPu,anbu]=Ufluxes(ulJ,ucJ,urJ,vIc,vImc,vIr,vImr)
Feu=0.5*(rho*urJ+rho*ucJ);Fwu=0.5*(rho*ucJ+rho*ulJ);
Fsu=0.5*(rho*vIc+rho*vImc);Fnu=0.5*(rho*vIr+rho*vImr);
aEuu=-0.5*Ae*Feu+Pr*De; aWuu=0.5*Aw*Fwu+Pr*Dw;
aNuu=-0.5*An*Fnu+Pr*Dn; aSuu=0.5*As*Fsu+Pr*Ds;
aWu=max(Fwu,max(aWuu,0));aEu=max(-Feu,max(aEuu,0));
aSu=max(Fsu,max(aSuu,0));aNu=max(-Fnu,max(aNuu,0));
aPu=aWu+aEu+aSu+aNu;
anbu=simplec*(Feu*Ae-Fwu*Aw+Fnu*An-Fsu*As);
end
function B=Vtridmatrix(ny,aNv,aSv,aPv)
B=zeros(ny,ny);
%B(1,1)=aPv;B(1,2)=-aNv;B(ny,ny-1)=-aSv;B(ny,ny)=aPv;%inputs the 1st and last rows of
matrix B
B=diag(diag(aPv*ones(ny)),0)+diag(-aNv*ones(ny-1,1),1)+diag(-aSv*ones(ny-1,1),-
1);%creates ny*ny tridiagonal matrix
%B(2:ny-1,1:ny)=Bin(2:ny-1,1:ny);% specifies the inner tridiagonal matrix of matrix B
end
function Vrhs=vRHS(dx,dy,aEv,aWv,vlJ,vrJ,pIl,pIc,TIc)
vpress=((pIl-pIc)/dy)*dx*dy;
sourceTerm=Pr*Ra*TIc*dx*dy;
Vrhs=aEv*vrJ+aWv*vlJ+vpress+1*sourceTerm;
end
function [aEv,aWv,aNv,aSv,aPv,anbv]=Vfluxes(vIl,vIc,vIr,uIc,uIl,uIpc,uIpl)
Fev=0.5*(rho*uIpc+rho*uIpl);Fwv=0.5*(rho*uIc+rho*uIl);
Fsv=0.5*(rho*vIl+rho*vIc);Fnv=0.5*(rho*vIc+rho*vIr);
aEvv=-0.5*Ae*Fev+Pr*De; aWvv=0.5*Aw*Fwv+Pr*Dw;
aNvv=-0.5*An*Fnv+Pr*Dn; aSvv=0.5*As*Fsv+Pr*Ds;
aWv=max(Fwv,max(aWvv,0));aEv=max(-Fev,max(aEvv,0));
aSv=max(Fsv,max(aSvv,0));aNv=max(-Fnv,max(aNvv,0));
aPv=aWv+aEv+aSv+aNv;
anbv=simplec*(Fev*Ae-Fwv*Aw+Fnv*An-Fsv*As);
end
function C=Ptridmatrix(ny,dx,dy)
```

```
AimJ=dy;AipJ=dy;AIjp=dx;AIjm=dx;
aIJ=AimJ+AipJ+AIjp+AIjm;
C=zeros(ny,ny);
%C(1,1)=aIJ;C(1,2)=-AipJ;C(ny,ny-1)=-AimJ;C(ny,ny)=aIJ;%inputs the 1st and last rows of
matrix B
C=diag(diag(aIJ*ones(ny)),0)+diag(-AipJ*ones(ny-1,1),1)+diag(-AimJ*ones(ny-1,1),-
1);%creates ny*ny tridiagonal matrix
%C(2:ny-1,1:ny)=Cin(2:ny-1,1:ny);% specifies the inner tridiagonal matrix of matrix B
end
function Prhs=pRHS(dx,dy,ucJ,urJ,vIc,vIr,pIl,pIr)
AimJ=rho*dy;AipJ=rho*dy;AIjp=rho*dx;AIjm=rho*dx;
Prhs=AIjm*pIl+AIjp*pIr+1*(rho*AimJ*ucJ-rho*AipJ*urJ)+1*(rho*AIjm*vIc-rho*AIjp*vIr);
end
function D=Ttridmatrix(nx,aNt,aSt,aPt)
D=zeros(nx,nx);
%D(1,1)=aPt;D(1,2)=-aEt;D(nx,nx-1)=-aWt;D(nx,nx)=aPt;%inputs the 1st and last rows of
matrix D
D=diag(diag(aPt*ones(nx)),0)+diag(-aNt*ones(nx-1,1),1)+diag(-aSt*ones(nx-1,1),-1);%creates
ny*ny tridiagonal matrix
%D(2:nx-1,1:nx)=Din(2:nx-1,1:nx);% specified the inner tridiagonal matrix of matrix D
end
function Trhs=tRHS(aEt,aWt,TIl,TIr)
Trhs=aWt*TIl+aEt*TIr;
end
function [aEt,aWt,aNt,aSt,aPt]=Tfluxes(ulJ,ucJ,urJ,vIc,vImc,vIr,vImr)
%function [aEt,aWt,aNt,aSt,aPt]=Tfluxes(vIl,vIc,vIr,uIc,uIl,uIpc,uIpl)
Fet=0.5*(rho*urJ+rho*ucJ);Fwt=0.5*(rho*ucJ+rho*ulJ);
Fst=0.5*(rho*vIc+rho*vImc);Fnt=0.5*(rho*vIr+rho*vImr);
%Fet=0.5*(rho*uIpc+rho*uIpl);Fwt=0.5*(rho*uIc+rho*uIl);
%Fst=0.5*(rho*vIl+rho*vIc);Fnt=0.5*(rho*vIc+rho*vIr);
aEtt=-0.5*Ae*Fet+De; aWtt=0.5*Aw*Fwt+Dw;
aNtt=-0.5*An*Fnt+Dn; aStt=0.5*As*Fst+Ds;
aWt=max(Fwt,max(aWtt,0));aEt=max(-Fet,max(aEtt,0));
aSt=max(Fst,max(aStt,0));aNt=max(-Fnt,max(aNtt,0));
aPt=aWt+aEt+aSt+aNt;
end
```

```
function simpLER()
clear all; clc;
```

```
nx=31;ny=31;K=25;L=1;
x0=0;xend=1;dx=(xend-x0)/(nx-1);y0=0;yend=L;dy=(yend-y0)/(ny-1);
x=x0:dx:xend;y=y0:dy:yend;

yloc=find(y>=0.2 & y<0.25);xloc=find(x<0.6);yloc2=find(y>=0.7 &
y<0.75);xloc2=find(x>0.3);
baffle=zeros(nx,ny);baffle(xloc,yloc)=1;baffle(xloc2,yloc2)=1;

us=zeros(nx,ny);vs=zeros(nx,ny);pp=zeros(nx,ny);
u=zeros(nx,ny);v=zeros(nx,ny);p=zeros(nx,ny);T=zeros(nx,ny);

ukh=zeros(nx,ny,K);vkh=zeros(nx,ny,K);uks=zeros(nx,ny,K);vks=zeros(nx,ny,K);pkp=zeros(n
x,ny,K);
pks=zeros(nx,ny,K);Tkm=zeros(nx,ny,K);
Usrhs=zeros(nx,ny,K);Vsrhs=zeros(nx,ny,K);Pprhs=zeros(nx,ny,K);Prhs=zeros(nx,ny,K);Trhs
=zeros(nx,ny,K)
uk=zeros(nx,ny,K);vk=zeros(nx,ny,K);pk=zeros(nx,ny,K);Tk=zeros(nx,ny,K);
%INITIAL GUESS/prescribed boundary conditions
us(1,1:ny)=0;vs(1,1:ny)=0;%T(1,1:ny)=1;%condition on left wall
us(nx,1:ny)=0;vs(nx,1:ny)=0;%T(nx,1:ny)=0;%condition on right wall
us(1:nx,1)=0;vs(1:nx,1)=0;T(1:nx,1)=1;%condition on bottom wall
us(1:nx,ny)=0;vs(1:nx,ny)=0;T(1:nx,ny)=0;%condition on top wall
%INITIAL GUESS/prescribed boundary conditions

Ae=dy;Aw=dy;An=dx;As=dx; De=Ae/dx;Dw=Aw/dx;Dn=An/dy;Ds=As/dy;
rho=1;Ra=10^5;Pr=0.71;tol=10^(-1);color='r';alphau=1;
for k=1:K
uks(:,:,1)=us(:,:);vks(:,:,1)=vs(:,:);pks(:,:,1)=ps(:,:);Tkm(:,:,1)=T(:,:);
uk(:,:,1)=u(:,:);vk(:,:,1)=v(:,:);Tk(:,:,1)=T(:,:);pkp(:,:,1)=pp(:,:);pk(:,:,1)=p(:,:);
 for i=2:length(x)-1
for j=2:length(y)-1
if(i>=min(yloc))& (j<=min(yloc))&(i<=max(xloc))
ukh(i,j,k)=0;vkh(i,j,k)=0; uks(i,j,k)=0;uk(i,j,k)=0;vks(i,j,k)=0;vk(i,j,k)=0;
Tkm(i,j,k)=0;Tk(i,j,k)=0;pk(i,j,k)=0;
elseif(i>=min(yloc2))& (j<=max(yloc2))&(i>=min(xloc2))
ukh(i,j,k)=0;vkh(i,j,k)=0; uks(i,j,k)=0;uk(i,j,k)=0;vks(i,j,k)=0;vk(i,j,k)=0;
Tkm(i,j,k)=0;Tk(i,j,k)=0;pk(i,j,k)=0;
else
%==================Estimation of initial velocities======================
[aEu,aWu,aNu,aSu,aPu,anbu,auNB]=Ufluxes(uks(i-
1,j,k),uks(i,j,k),uks(i+1,j,k),vks(i,j,k),vks(i-1,j,k),vks(i,j+1,k),vks(i-1,j+1,k));
[aEv,aWv,aNv,aSv,aPv,anbv,avNB]=Vfluxes(vks(i,j-
1,k),vks(i,j,k),vks(i,j+1,k),uks(i,j,k),uks(i,j-1,k),uks(i+1,j,k),uks(i+1,j-1,k));

Usrhs(2:nx-1,2:ny-1,k)=uRHS(dx,dy,aNu,aSu,uks(2:nx-1,1:ny-2,k),uks(2:nx-
1,3:ny,k),pks(1:nx-2,2:ny-1,k),pks(2:nx-1,2:ny-1,k),Tk(2:nx-1,2:ny-1,k));
uCoeffMatstar=Utridmatrix(nx,aEu,aWu,aPu);[Qustar,Rustar]=lu(uCoeffMatstar);
```

```
Vsrhs(2:nx-1,2:ny-1,k)=vRHS(dx,dy,aEv,aWv,vks(1:nx-2,2:ny-1,k),vks(3:nx,2:ny-
1,k),pks(2:nx-1,1:ny-2,k),pks(2:nx-1,2:ny-1,k),Tk(2:nx-1,2:ny-1,k));
vCoeffMatstar=Vtridmatrix(ny,aNv,aSv,aPv);[Qvstar,Rvstar]=lu(vCoeffMatstar);

uks(2:nx-1,j,k)=Rustar(2:nx-1,2:nx-1)\(Qustar(2:nx-1,2:nx-1)\Usrhs(2:nx-1,j,k));
vks(i,2:ny-1,k)=Rvstar(2:ny-1,2:ny-1)\(Qvstar(2:ny-1,2:ny-1)\Vsrhs(i,2:ny-1,k)');%TO BE
VERIFIED
%=========Estimation of initial velocities==================================

%====STEP 1: calculation of pseudo-velocities=============================
b(2:nx-1,2:ny-1,k)=Pr*Ra*Tk(2:nx-1,2:ny-1,k)*dx*dy;
ukh(2:nx-1,j,k)=auNB/anbu; vkh(i,2:ny-1,k)=(avNB+b(i,2:ny-1,k))/anbv;

%====STEP 2: solve pressure equation=============================
Prhs(2:nx-1,2:ny-1,k)=pRHS(dx,dy,ukh(2:nx-1,2:ny-1,k),ukh(3:nx,2:ny-1,k),vkh(2:nx-1,2:ny-
1,k),vkh(2:nx-1,3:ny,k),pkp(2:nx-1,1:ny-2,k),pkp(2:nx-1,3:ny,k));
pCoeffMat=Ptridmatrix(ny,dx,dy);[Qp,Rp]=lu(pCoeffMat);
pk(2:nx-1,j,k)=Rp(2:nx-1,2:nx-1)\(Qp(2:nx-1,2:nx-1)\Prhs(2:nx-1,j,k));

%====STEP 3: solve discretized momentum
equations=============================
Usrhs(2:nx-1,2:ny-1,k)=uRHS(dx,dy,aNu,aSu,uks(2:nx-1,1:ny-2,k),uks(2:nx-
1,3:ny,k),pk(1:nx-2,2:ny-1,k),pk(2:nx-1,2:ny-1,k),Tk(2:nx-1,2:ny-1,k));
uCoeffMatstar=Utridmatrix(nx,aEu,aWu,aPu);[Qustar,Rustar]=lu(uCoeffMatstar);

Vsrhs(2:nx-1,2:ny-1,k)=vRHS(dx,dy,aEv,aWv,vks(1:nx-2,2:ny-1,k),vks(3:nx,2:ny-
1,k),pk(2:nx-1,1:ny-2,k),pk(2:nx-1,2:ny-1,k),Tk(2:nx-1,2:ny-1,k));
vCoeffMatstar=Vtridmatrix(ny,aNv,aSv,aPv);[Qvstar,Rvstar]=lu(vCoeffMatstar);

uks(2:nx-1,j,k)=Rustar(2:nx-1,2:nx-1)\(Qustar(2:nx-1,2:nx-1)\Usrhs(2:nx-1,j,k));
vks(i,2:ny-1,k)=Rvstar(2:ny-1,2:ny-1)\(Qvstar(2:ny-1,2:ny-1)\Vsrhs(i,2:ny-1,k)');%TO BE
VERIFIED
%====STEP 4: solve pressure correction equation=============================
Pprhs(2:nx-1,2:ny-1,k)=pRHS(dx,dy,uks(2:nx-1,2:ny-1,k),uks(3:nx,2:ny-1,k),vks(2:nx-1,2:ny-
1,k),vks(2:nx-1,3:ny,k),pkp(2:nx-1,1:ny-2,k),pkp(2:nx-1,3:ny,k));
pCoeffMat=Ptridmatrix(ny,dx,dy);[Qpprime,Rpprime]=lu(pCoeffMat);
pkp(2:nx-1,j,k)=Rpprime(2:nx-1,2:nx-1)\(Qpprime(2:nx-1,2:nx-1)\Pprhs(2:nx-1,j,k));

%=====STEP 5: correct velocities============================
uk(i,j,k)=uks(i,j,k)+(alphau*Ae/(eps+aPu))*(pkp(i-1,j,k)-pkp(i,j,k));
vk(i,j,k)=vks(i,j,k)+(alphau*Aw/(eps+aPv))*(pkp(i,j-1,k)-pkp(i,j,k));

%======STEP 6: solve all other discretized transport
equations======================
[aEt,aWt,aNt,aSt,aPt]=Tfluxes(uk(i-1,j,k),uk(i,j,k),uk(i+1,j,k),vk(i,j,k),vk(i-
1,j,k),vk(i,j+1,k),vk(i-1,j+1,k));
Trhs(2:nx-1,2:ny-1,k)=tRHS(aNt,aSt,Tkm(2:nx-1,1:ny-2,k),Tkm(2:nx-1,3:ny,k));
tCoeffMat=Ttridmatrix(nx,aEt,aWt,aPt);[Qt,Rt]=lu(tCoeffMat);
Tkm(2:nx-1,j,k)=Rt(2:nx-1,2:nx-1)\(Qt(2:nx-1,2:nx-1)\Trhs(2:nx-1,j,k));Tk(i,j,k)=Tkm(i,j,k);
```

```
end
end
end
figure(1)
quiver(x(2:2:nx-1),y(2:2:ny-1),uk(2:2:nx-1,2:2:ny-1,K)',vk(2:2:nx-1,2:2:ny-1,K)',1)
axis([0 xend 0 yend]);xlabel('X');ylabel('Y');title('velocity vector plot')
figure(2)
mesh(x(2:nx-1),y(2:ny-1),uk(2:nx-1,2:ny-1,K)');xlabel('X');ylabel('Y');title('u-velocity')
figure(3)
mesh(x(2:nx-1),y(2:ny-1),vk(2:nx-1,2:ny-1,K)');xlabel('X');ylabel('Y');title('v-velocity')
figure(4)
mesh(x(1:nx-1),y(1:ny-1),Tk(1:nx-1,1:ny-1,K));title('temperature');xlabel('Y');ylabel('X')
figure(5)
% surf(x(2:nx-1),y(2:ny-1),baffle(2:nx-1,2:ny-1)');shading flat;view(2);colormap(flipud(gray));
 hold on
[C,h] = contour(x(2:nx-1),y(2:ny-1),Tk(2:nx-1,2:ny-1,K)','LineWidth',1.2,'LineStyle','-');
set(h,'ShowText','off','TextStep',get(h,'LevelStep'));%colormap(flipud(hsv));
hold off
xlabel('X');ylabel('Y');title('Isotherms:SIMPLER');%title('Isotherms: SIMPLE vs SIMPLER')
figure(6)
-surf(x(2:nx-1),y(2:ny-1),baffle(2:nx-1,2:ny-1)');shading flat;view(2);colormap(flipud(gray));
 hold on
[C,h] = contour(x(2:nx-1),y(2:ny-1),vk(2:nx-1,2:ny-1,K)','Linewidth',1.2,'LineStyle','-');
 colormap(cool(7));colorbar('eastoutside')
hold off
xlabel('X');ylabel('Y');title('v velocity:SIMPLER');%title('v velocity: SIMPLE vs SIMPLER')

figure(7)
surf(x(2:nx-1),y(2:ny-1),baffle(2:nx-1,2:ny-1)');shading flat;view(2);colormap(flipud(gray));
 hold on
[C,h] = contour(x(2:nx-1),y(2:ny-1),uk(2:nx-1,2:ny-1,K)','Linewidth',1);
colormap hot
hold off
xlabel('X');ylabel('Y');title('Velocity u')

figure(8)
hold on
plot(x(1:nx),uk(1:nx,floor(0.5*ny),K),color,'Linewidth',2) xlabel('X') ylabel('Velocity, u')
hold off
figure(9)
hold on
plot(x(1:nx),vk(1:nx,floor(0.5*ny),K),color,'Linewidth',2) xlabel('X') ylabel('Velocity, v')
hold off
figure(10)
hold on
plot(x(1:nx),Tk(1:nx,floor(0.5*ny),K),color,'Linewidth',2) xlabel('X') ylabel('Temperature, T')
hold off
function A=Utridmatrix(nx,aEu,aWu,aPu)
```

```
A=diag(diag(aPu*ones(nx)),0)+diag(-aEu*ones(nx-1,1),1)+diag(-aWu*ones(nx-1,1),-
1);%creates ny*ny tridiagonal matrix
end
 function [aEu,aWu,aNu,aSu,aPu,anbu,auNB]=Ufluxes(ulJ,ucJ,urJ,vIc,vImc,vIr,vImr)
Feu=0.5*(rho*urJ+rho*ucJ);Fwu=0.5*(rho*ucJ+rho*ulJ);Fsu=0.5*(rho*vIc+rho*vImc);Fnu=0
.5*(rho*vIr+rho*vImr);  aEuu=-0.5*Ae*Feu+Pr*De; aWuu=0.5*Aw*Fwu+Pr*Dw;
aNuu=-0.5*An*Fnu+Pr*Dn; aSuu=0.5*As*Fsu+Pr*Ds;
aWu=max(Fwu,max(aWuu,0));aEu=max(-Feu,max(aEuu,0));
aSu=max(Fsu,max(aSuu,0));aNu=max(-Fnu,max(aNuu,0)); aPu=aWu+aEu+aSu+aNu;
uEu=0.5*(urJ+ucJ);uWu=0.5*(ucJ+ulJ);  uSu=0.5*(vIc+vImc);uNu=0.5*(vIr+vImr);
auNB=aWu*uWu+aEu*uEu+aSu*uSu+aNu*uNu;
saWu=Pr*Dw+max(Fwu,0);saEu=Pr*De+max(0,-Feu);
saSu=Pr*Ds+max(Fsu,0);saNu=Pr*Dn+max(0,-Fnu);
anbu=alphau*(saWu+saEu+saSu+saNu);
end
function B=Vtridmatrix(ny,aNv,aSv,aPv)
B=zeros(ny,ny);
B=diag(diag(aPv*ones(ny)),0)+diag(-aNv*ones(ny,1),1)+diag(-aSv*ones(ny-1,1),-1);%creates
ny*ny tridiagonal matrix
end
function Vrhs=vRHS(dx,dy,aEv,aWv,vlJ,vrJ,pIl,pIc,TIc)
vpress=((pIl-pIc)/dy)*dx*dy; sourceTerm=Pr*Ra*TIc*dx*dy;
Vrhs=aEv*vrJ+aWv*vlJ+vpress+1*sourceTerm;
end
function [aEv,aWv,aNv,aSv,aPv,anbv,avNB]=Vfluxes(vIl,vIc,vIr,uIc,uIl,uIpc,uIpl)
Fev=0.5*(rho*uIpc+rho*uIpl);Fwv=0.5*(rho*uIc+rho*uIl);
Fsv=0.5*(rho*vIl+rho*vIc);Fnv=0.5*(rho*vIc+rho*vIr);
aEvv=-0.5*Ae*Fev+Pr*De; aWvv=0.5*Aw*Fwv+Pr*Dw;
aNvv=-0.5*An*Fnv+Pr*Dn; aSvv=0.5*As*Fsv+Pr*Ds;
aWv=max(-Fwv,max(aWvv,0));aEv=max(Fev,max(aEvv,0));
aSv=max(Fsv,max(aSvv,0));aNv=max(Fnv,max(aNvv,0));
aPv=aWv+aEv+aSv+aNv;
vEv=0.5*(uIpc+uIpl);vWv=0.5*(uIc+uIl);  vSv=0.5*(vIl+vIc);vNv=0.5*(vIc+vIr);
avNB=aWv*vWv+aEv*vEv+aSv*vSv+aNv*vNv;
saWv=Pr*Dw+max(Fwv,0);saEv=Pr*De+max(0,-Fev);
saSv=Pr*Ds+max(Fsv,0);saNv=Pr*Dn+max(0,-Fnv);
anbv=alphau*(saWv+saEv+saSv+saNv);
end
function C=Ptridmatrix(ny,dx,dy)
AimJ=dy;AipJ=dy;AIjp=dx;AIjm=dx;
aIJ=AimJ+AipJ+AIjp+AIjm;
C=zeros(ny,ny);
C=diag(diag(aIJ*ones(ny)),0)+diag(-AipJ*ones(ny-1,1),1)+diag(-AimJ*ones(ny-1,1),-
1);%creates ny*ny tridiagonal matrix
end
function Prhs=pRHS(dx,dy,ucJ,urJ,vIc,vIr,pIl,pIr)
AimJ=rho*dy;AipJ=rho*dy;AIjp=rho*dx;AIjm=rho*dx;
Prhs=AIjm*pIl+AIjp*pIr+1*(rho*AimJ*ucJ-rho*AipJ*urJ)+1*(rho*AIjm*vIc-rho*AIjp*vIr);
end
```