International Journal of
Mathematical Modelling & Computations
Vol. 11, No. 01, Winter 2021, 49- 60

A Numerical Solution of Three-Dimensional Unsteady State Heat Equation

E. G. Tsega*

Department of Mathematics, College of Science, Bahir Dar University, Bahir Dar, Ethiopia.

Abstract. Heat equation is a partial differential equation that describes the distribution of temperature (heat) in a given body over time. In this study, a finite volume based method is used to solve three-dimensional heat equation. A MATLAB code is developed to implement the numerical method in a unit cube. The stability of the numerical scheme is analysed using the Von Neumann method. An example is provided in order to demonstrate the method. The numerical solution by the method is in an excellent agreement with the exact solution for the example considered. The computational procedures used in this study can provide good insights to solve a three dimensional diffusion equation arising in many physical phenomena.

Received: 16 August 2020, Revised: 09 October 2020, Accepted: 23 October 2020.

Keywords: Heat equation; Unsteady; Three-dimensional; Finite volume method; MATLAB code.

Index to information contained in this paper

1. Introduction

If parts of a body are exposed to different temperature conditions, heat flows through the body from the region of high temperature to that of lower temperature [15]. Temperature distribution includes in a metal block or wire, in a liquid, in a room, in a living tissue and many others. The temporal and spatial temperature distribution in the body is described mathematically by heat equation. In order to understand the temperature distribution, a solution of heat equation is required. Analytical and numerical methods are employed to solve heat equation [13]. Finite difference, finite element and finite volume methods have been used to obtain numerical the solutions for one or two dimension [2, 5, 12, 14, 17, 18].

Numerous numerical solutions of three-dimensional heat equation have been reported in literature. Sutradhar et al. [16] applied Laplace transform boundary element method to solve the transient heat the equation by performing the numerical implementation using a Galerkin approximation. Patil et al. [11] presented an algorithm to solve three dimensional steady state heat transfers in cube using finite volume grid technique. The computation was performed with MS excel. Ang and Gumel [1] used a boundary integral method together with finite difference method to solve three-dimensional heat equation. Gavete et al. [4] solved the three-dimensional elliptic and parabolic equations using generalized finite difference method. In solving partial differential equations (including heat equation) effective computer program and precise implementation of boundary conditions are required.

*Corresponding author. Email: endalebdumath2016@gmail.com

2. Finite volume discretization

The three-dimensional heat equation is represented as [8]:

$$\frac{\partial u}{\partial t} = \alpha\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right) \tag{1}$$

where $u(x, y, z, t)$ is temperature and the constant α is thermal diffusivity. Using del operator and taking α to be 1 [6], (1) can be written as

$$\frac{\partial u}{\partial t} = \nabla.(\nabla u) \tag{2}$$

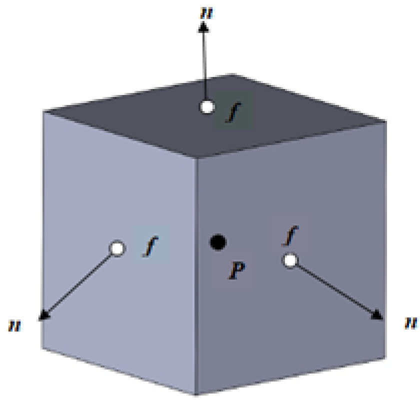

Figure 1. 3D control volume.

Integrating (2) over the control volume Ω (see Figure 1), we get

$$\iiint_{\Omega} \frac{\partial u}{\partial t} d\Omega = \iiint_{\Omega} \nabla.(\nabla u) d\Omega \tag{3}$$

The integral in the left side of (3) is evaluated as [10]

$$\iiint_{\Omega} \frac{\partial u}{\partial t} d\Omega = \frac{\partial u}{\partial t} \Delta V \tag{4}$$

where ΔV is the volume of the control volume.
Applying Gauss's Divergence Theorem in the left right side of (3) we have

$$\iiint_{\Omega} \nabla.(\nabla u) d\Omega = \iint_{S} (\nabla u).\boldsymbol{n} dS \tag{5}$$

where S is the boundary surface of the control volume and $\boldsymbol{n}$ is the outward unit normal to S.

$$\iint_{S} (\nabla u).\boldsymbol{n} dS = \sum_{f}\left(\iint_{S} (\nabla u)_f .\boldsymbol{n}_f dS\right) \approx \sum_{f} (\overline{\nabla u})_f .\boldsymbol{n}_f A_f \approx \sum_{f} (\nabla u)_f .\boldsymbol{n}_f A_f \tag{6}$$

where f indicates summation over face of each control volume, $\left(\overline{\nabla u}\right)_f$ is the average temperature gradient at the face *f*, $(\nabla u)_f$ is the temperature gradient at the center of the face *f*, $\boldsymbol{n}_f$ is the outward unit normal to the *f* and A_f is the area of the face *f* [9].

From (3), (4) and (6), we get

$$\frac{\partial u}{\partial t}\Delta V = \sum_f (\nabla u)_f . \boldsymbol{n}_f A_f \tag{7}$$

In order to illustrate the finite volume solution procedure to three-dimensional heat equation, let us consider unsteady heat flow in a unit cube shown in Figure 2.

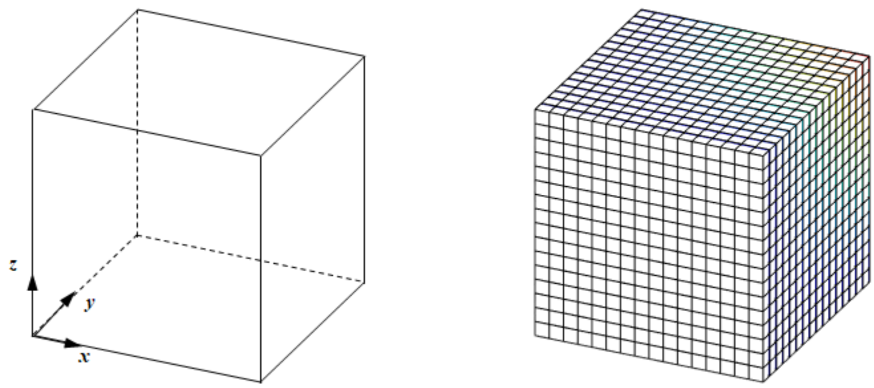

Figure 2. Geometry and finite volume mesh of heat flow domain.

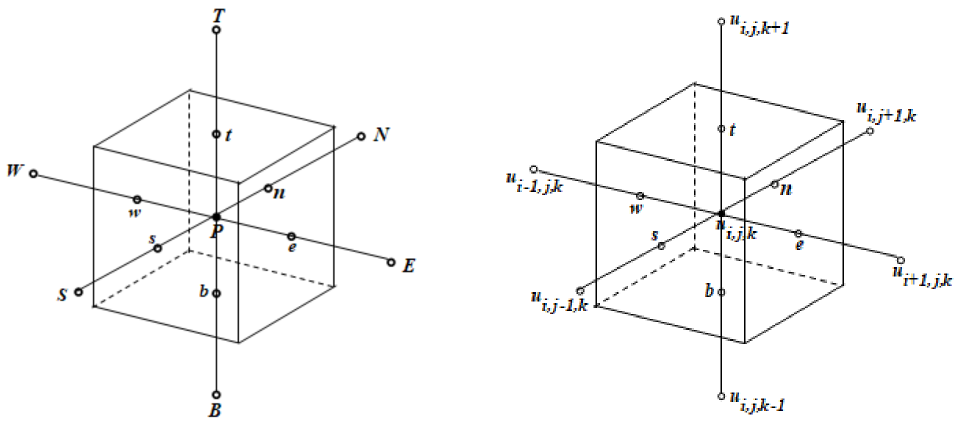

Figure 3. Face and center values at a control volume.

If we apply (7) on a control volume (Figure 3), we get

$$\frac{\partial u}{\partial t}\Delta x\Delta y\Delta z = \left[\left((\frac{\partial u}{\partial x})_e - (\frac{\partial u}{\partial x})_w\right)\Delta y\Delta z + \left((\frac{\partial u}{\partial y})_n - (\frac{\partial u}{\partial y})_s\right)\Delta x\Delta z + \left((\frac{\partial u}{\partial z})_t - (\frac{\partial u}{\partial z})_b\right)\Delta x\Delta y\right] \tag{8}$$

Here the expressions in the brackets with *e*, *w*, *n*, *s*, *t* and *b* represent the values of the expressions at the center of east, west, north, south, top and bottom faces of the control volume respectively and Δx, Δy and Δz are the grid spacing in *x*, *y* and *z* directions. The symbols *P*, *E*, *W*, *N*, *S*, *T* and *B* in Figure 3 indicate center of the control volumes at which the values of temperature field are computed. Approximating the time derivative using forward difference and spatial derivatives using central difference [10] in (8), for interior control volumes we have

$$\frac{u_{i,j,k}^{n+1}-u_{i,j,k}^{n}}{\Delta t}\Delta x\Delta y\Delta z=\Bigg[(\frac{u_{i+1,j,k}^{n}-u_{i,j,k}^{n}}{\Delta x}-\frac{u_{i,j,k}^{n}-u_{i-1,j,k}^{n}}{\Delta x})\Delta y\Delta z$$
$$+(\frac{u_{i,j+1,k}^{n}-u_{i,j,k}^{n}}{\Delta y}-\frac{u_{i,j,k}^{n}-u_{i,j-1,k}^{n}}{\Delta y})\Delta x\Delta z+(\frac{u_{i,j,k+1}^{n}-u_{i,j,k}^{n}}{\Delta z}-\frac{u_{i,j,k}^{n}-u_{i,j,k-1}^{n}}{\Delta z})\Delta x\Delta y\Bigg] \quad (9)$$

where n is the current time index and Δt is the step size for time discretization. (9) can be written as

$$u_{i,j,k}^{n+1}=u_{i,j,k}^{n}+\Delta t\Bigg[(\frac{u_{i+1,j,k}^{n}-u_{i,j,k}^{n}}{\Delta x^2}-\frac{u_{i,j,k}^{n}-u_{i-1,j,k}^{n}}{\Delta x^2})+(\frac{u_{i,j+1,k}^{n}-u_{i,j,k}^{n}}{\Delta y^2}-\frac{u_{i,j,k}^{n}-u_{i,j-1,k}^{n}}{\Delta y^2})$$
$$+(\frac{u_{i,j,k+1}^{n}-u_{i,j,k}^{n}}{\Delta z^2}-\frac{u_{i,j,k}^{n}-u_{i,j,k-1}^{n}}{\Delta z^2})\Bigg] \quad (10)$$

For boundary control volumes, backward and forward difference approximations of derivative are used additionally. For the control volume at the bottom left corner, (9) is modified as

$$\frac{u_{i,j,k}^{n+1}-u_{i,j,k}^{n}}{\Delta t}\Delta x\Delta y\Delta z=\Bigg[(\frac{u_{i+1,j,k}^{n}-u_{i,j,k}^{n}}{\Delta x}-\frac{u_{i,j,k}^{n}-uL(j,k)}{\Delta x/2})\Delta y\Delta z$$
$$+(\frac{u_{i,j+1,k}^{n}-u_{i,j,k}^{n}}{\Delta y}-\frac{u_{i,j,k}^{n}-uF(i,k)}{\Delta y/2})\Delta x\Delta z+(\frac{u_{i,j,k+1}^{n}-u_{i,j,k}^{n}}{\Delta z}-\frac{u_{i,j,k}^{n}-uB(i,j)}{\Delta z/2})\Delta x\Delta y\Bigg] \quad (11)$$

or

$$u_{i,j,k}^{n+1}=u_{i,j,k}^{n}+\Delta t\Bigg[(\frac{u_{i+1,j,k}^{n}-u_{i,j,k}^{n}}{\Delta x^2}-2\frac{u_{i,j,k}^{n}-uL(j,k)}{\Delta x^2})$$
$$+(\frac{u_{i,j+1,k}^{n}-u_{i,j,k}^{n}}{\Delta y^2}-2\frac{u_{i,j,k}^{n}-uF(i,k)}{\Delta y^2})+(\frac{u_{i,j,k+1}^{n}-u_{i,j,k}^{n}}{\Delta z^2}-2\frac{u_{i,j,k}^{n}-uB(i,j)}{\Delta z^2})\Bigg] \quad (12)$$

where uL, uF and uB are the boundary values at the left, front and bottom faces of the cube respectively. The iteration scheme for the remaining boundary control volume can be formulated analogously.

3. Stability analysis

In order to find the stability condition for the iteration scheme using Von Neumann method, we replace [7]

$$u_{p,q,r}^{n}=Ae^{i\theta_1 p\Delta x}e^{i\theta_2 q\Delta y}e^{i\theta_3 r\Delta z}e^{\alpha(n\Delta t)}=Ae^{i\theta_1 p\Delta x}e^{i\theta_2 q\Delta y}e^{i\theta_3 r\Delta z}\xi^{\alpha} \quad (13)$$

where $\xi=e^{n\Delta t}$ is an amplification factor, $i=\sqrt{-1}$, A, θ_1, θ_2 ,and θ_3 are constants. The iteration scheme in (10) is stable if $|\xi|\le 1$ [3]. Using (13) in (10) and simplifying, we get

$$\xi = 1 + \frac{e^{i\theta_1\Delta x} - 2 + e^{-i\theta_1\Delta x}}{\Delta x^2} + \frac{e^{i\theta_2\Delta y} - 2 + e^{-i\theta_2\Delta y}}{\Delta y^2} + \frac{e^{i\theta_3\Delta z} - 2 + e^{-i\theta_3\Delta z}}{\Delta z^2} \tag{14}$$

For the worst case, $\theta_1\Delta x = \pi = \theta_2\Delta y = \theta_3\Delta z$ [12]. This gives

$$\xi = 1 - \frac{4\Delta t}{\Delta x^2} - \frac{4\Delta t}{\Delta y^2} - \frac{4\Delta t}{\Delta z^2} \tag{15}$$

The stability condition $|\xi| \le 1$ yields

$$\frac{\Delta t}{\Delta x^2} + \frac{\Delta t}{\Delta y^2} + \frac{\Delta t}{\Delta z^2} \le \frac{1}{2} \tag{16}$$

or

$$\Delta t \le \frac{1}{2}\left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2} + \frac{1}{\Delta z^2}\right)^{-1} \tag{17}$$

If $\Delta x = \Delta y = \Delta z = h$, then the iteration scheme in (10) is stable if

$$\Delta t \le \frac{h^2}{6} \tag{18}$$

4. A test problem and solutions

The equation and the conditions used by [1] are chosen as a test problem for this study. The heat equation considered in [1] was

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}, \quad 0 < x, y, z < 1 \tag{19}$$

with initial condition

$$u(x,y,z,0) = \sin(\frac{\pi}{3}[x+y+z]) + xyz, \; 0 < x, y, z < 1$$

and boundary conditions

$$u(0,y,z,t) = e^{\frac{-\pi^2 t}{3}} \sin(\frac{\pi}{3}[y+z]), \qquad 0 < y, z < 1, \quad t \ge 0$$

$$u(1,y,z,t) = e^{\frac{-\pi^2 t}{3}} \sin(\frac{\pi}{3}[1+y+z]) + yz, \qquad 0 < y, z < 1, \quad t \ge 0$$

$$u(x,0,z,t) = e^{\frac{-\pi^2 t}{3}} \sin(\frac{\pi}{3}[x+z]), \qquad 0 < x, z < 1, \quad t \ge 0$$

$$u(x,1,z,t) = e^{\frac{-\pi^2 t}{3}} \sin(\frac{\pi}{3}[x+1+z]) + xz, \qquad 0 < x, z < 1, \quad t \ge 0$$

$$u(x,y,0,t) = e^{\frac{-\pi^2 t}{3}} \sin(\frac{\pi}{3}[x+y]), \quad 0 < x, y < 1, \quad t \ge 0$$

$$u(x,y,1,t) = e^{\frac{-\pi^2 t}{3}} \sin(\frac{\pi}{3}[x+y+1]) + xy, \qquad 0 < x, y < 1, \quad t \ge 0$$

The exact solution of the test problem is

$$u(x,y,z,t)=e^{\frac{-\pi^2 t}{3}}\sin(\frac{\pi}{3}[x+y+z])+xyz$$

To solve (19) with the given initial and boundary conditions by the numerical method discussed so far, 32×32×32 cubic finite volume cells of grid size $\Delta x = \Delta y = \Delta z = \frac{1}{32}$ are taken. The time step of $\Delta t = 0.0001$ is chosen based on the stability condition (18). Figure 4 shows the contours plot of the finite volume and exact solutions for t = 0.05, 0.25, 0.5, 1. The solutions along central line of the cube ($x=0.5$, $y=0.5$, $0 \le z \le 1$) for these times are also displayed in Figure 5. Since the vertical central line does not include the center of the control volumes in the discretization, the average values of temperature on the neighbouring lines ($x=0.484375$, $y=0.484375$, $0 \le z \le 1$) and ($x=0.515625$, $y=0.515625$, $0 \le z \le 1$) are used to calculate the values on the vertical central line. The solutions of heat equation using 16×16×16 cells and $t=0.5$ along the vertical central is depicted with table of values (Table 1).

Table 1. A Comparison between finite volume and exact solutions at $t = 0.5$.

z	FV solution	Exact solution	error
0.0000	0.16680689	0.16680689	0.00000000
0.0313	0.17774781	0.17771164	0.00003617
0.0938	0.19901144	0.19897236	0.00003908
0.1563	0.21952329	0.21948181	0.00004148
0.2188	0.23926275	0.23921932	0.00004343
0.2813	0.25821250	0.25816756	0.00004494
0.3438	0.27635860	0.27631255	0.00004605
0.4063	0.29369054	0.29364376	0.00004678
0.4688	0.31020128	0.31015414	0.00004714
0.5313	0.32588732	0.32584018	0.00004714
0.5938	0.34074864	0.34070186	0.00004678
0.6563	0.35478877	0.35474272	0.00004605
0.7188	0.36801474	0.36796980	0.00004494
0.7813	0.38043707	0.38039364	0.00004343
0.8438	0.39206968	0.39202819	0.00004148
0.9063	0.40292990	0.40289082	0.00003908
0.9688	0.41303834	0.41300217	0.00003617
1.0000	0.41778345	0.41778345	0.00000000

The error of the numerical method is indicated in table. As we can observe from the figures and the table, the final volume solution of three-dimensional heat equation is almost identical with that of the exact solution. To get more accurate solution with the numerical method, small mesh spacing is required. This in turn needs small time step to avoid numerical instability in the computation. The algorithm and the MATLAB code of the numerical method are indicated at the appendix.

5. Conclusion

In this study, three-dimensional heat equation is solved in a unit cube using a finite volume based numerical method. A MATLAB code is used perform the computation. The stability of the numerical method is analyzed. The solutions obtained by the numerical method are compared with exact solutions and show an excellent agreement. This indicates that the three-dimensional heat equation is discretised properly, the

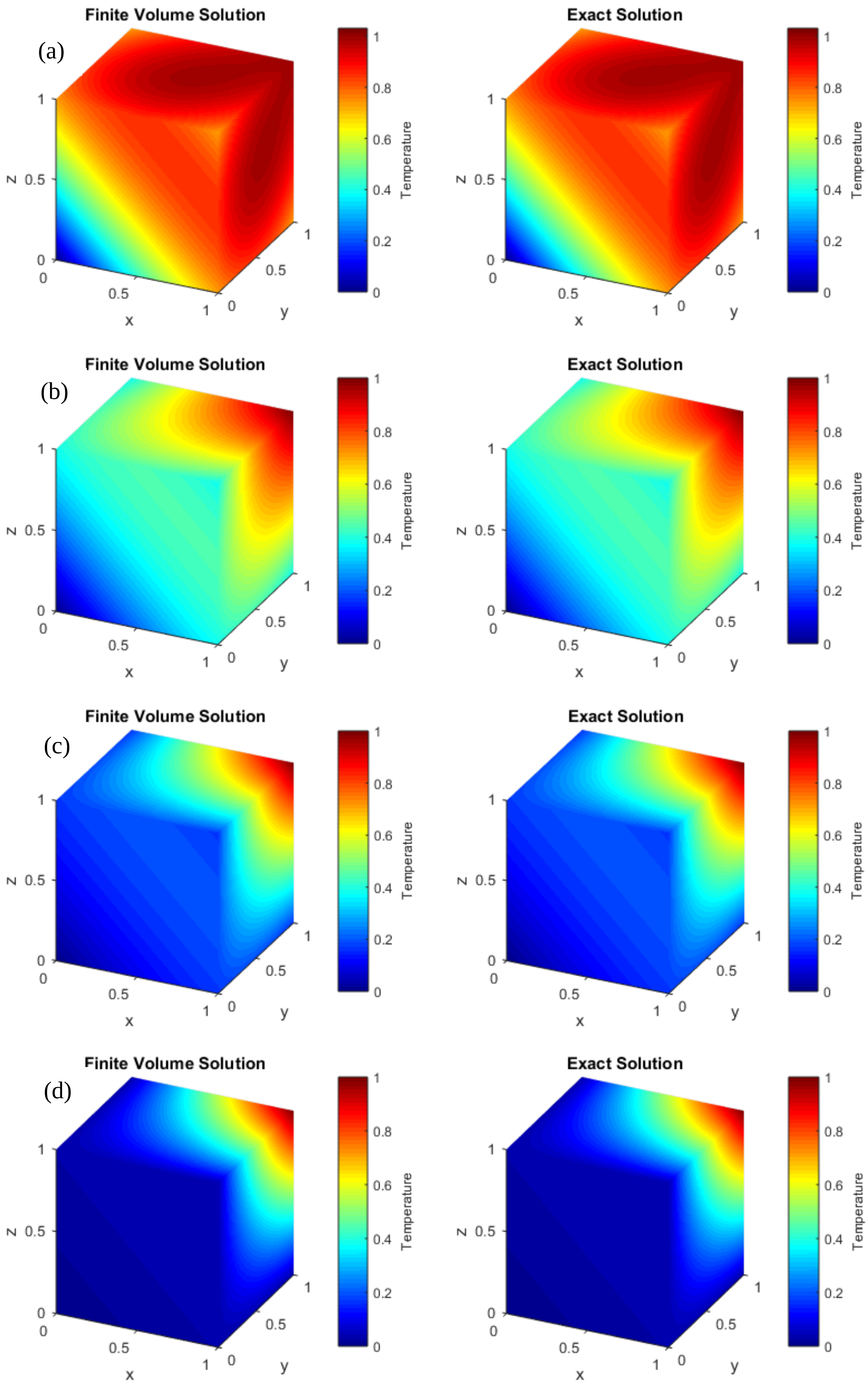

Figure 4. Contour plot of temperature at (a) t = 0.05 (b) t = 0.25 (c) t = 0.5 (d) t = 1.

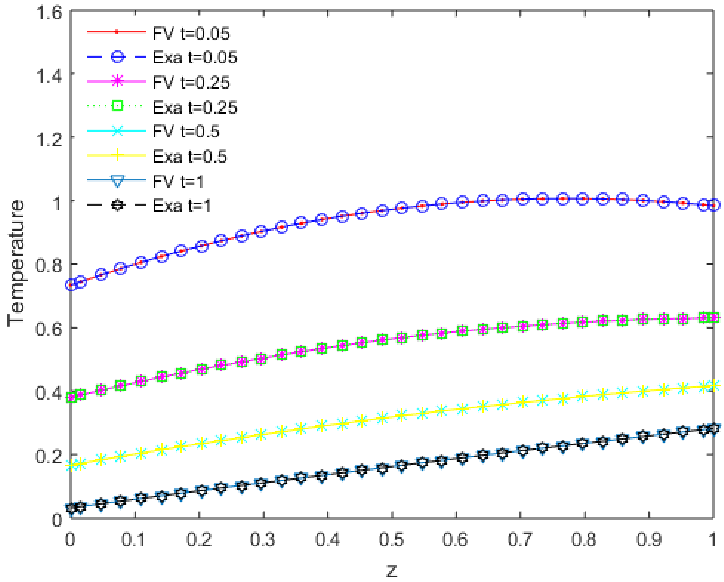

Figure 5. Temperature distribution along the vertical centreline of the cube at different times.

boundary conditions are implemented precisely and the code is written correctly. The findings of this study may provide useful technique to implement a numerical method on a computer in solving partial differential equations in three dimensions.

References

[1] W. T. Ang and A. B. Gumel, A boundary integral method for the three-dimensional heat equation subject to specification of heat energy, Journal of Computational and Applied Mathematics, **135** (2001) 303-311.

[2] C. N. Dawson, Q. Du and T. F. Dupont, A finite difference domain decomposition algorithm for numerical solution for the heat equation, Mathematics of Computation, **57 (195)** (1991) 63-71.

[3] W. Ehlers, S. Zinatbakhsh and B. Markert, Stability analysis of finite difference schemes revisited: A study of decoupled solution strategies for coupled multifield problems, International Journal of Numerical Methods for Engineering, **94** (2013) 758-786.

[4] L. Gavete, J. J. Benito and F. Urena, Generalized finite differences for solving 3D elliptic and parabolic equations, Applied Mathematical Modelling, **40 (2)** (2016) 955-965.

[5] V. Gülkaç, 2010. A numerical solution of the two-dimensional fusion problem with convective boundary conditions. International Journal for Computational Methods in Engineering Science and Mechanics, **11 (1)** (2010) 20-26. doi:10.1080/15502280903446853.

[6] M. Hasnat, N. Kaid, M. Bensafi and A. Belkacem, A numerical technique finite volume method for solving diffusion 2D problem, The International Journal of Engineering and Science, **4 (10)** (2015) 35-41.

[7] M. K. Jain, S. R. K. Iyengar and R. K. Jain, Numerical Methods for Scientific and Engineering Computation, Seventh Edition, New Age International Publisher, New Delhi, (2019).

[8] S. Mazumder, Numerical Methods for Partial Differential Equations: Finite Difference and Finite Volume Methods, Academic Press, New York, (2016).

[9] F. Moukalled, L. Mangani and M. Darwish, The Finite Volume Method in Computational Fluid Dynamics: An Advanced Introduction with OpenFOAM and MATLAB, Springer, Berlin, (2015).

[10] S. V. Patankar, Numerical Heat Transfer and Fluid Flow, Hemisphere Publishing Corporation, New York, (1980).

[11] P. V. Patil, P. N. Bhirud, J. S. V. R. K. Prasad, Algorithm: three dimensional finite volume numerical grid technique, Global Journal of Pure and Applied Mathematics, **13 (9)** (2017) 5655-5671.

[12] M. N. Rajput, A. A. Shaikh and S. A. Kamboh, Computational analysis of the stability of 2D heat equation on elliptical domain using finite difference method, Asian Research Journal of Mathematics, **16 (3)** (2020) 8-19.

[13] J. I. Ramos, Analytical and numerical solutions to some one-dimensional relativistic heat equations, Applied Mathematical Modelling, **46** (2017) 181-202.

[14] J. N. Reddy, An introduction to Finite Element Method, Third Edition, McGraw-Hill, New York, (2006).

[15] I. N. Sneddon, Fourier Transforms, Dover Publication, New York, (1995).

[16] A. Sutradhar, G. H. Paulino and L. J. Gray, Transient heat conduction in homogeneous and non-homogeneous materials by Laplace transform Galerkin boundary element method, Engineering Analysis with Boundary Elements, **26** (2002) 119-132.

[17] H. K. Versteeg and W. Malalasekera, An Introduction to Computational Fluid Dynamics: The Finite Volume

Method, Second edition, Prentice Hall, USA, (2007).
[18] G. Zhao and R. Zhang, The new numerical method for solving the system of two-dimensional Burgers' equations, Computers & Mathematics with Applications, **62 (8)** (2011) 3279-3291.

Appendix A: Algorithm for the numerical method

1. Divide the computational domain into uniform Control volumes (cells).
2. Set initial conditions at the center of all control volumes.
3. Impose boundary conditions over the control volume faces at the boundaries of the domain.
4. Compute the temperature at the center of each control volume for the time level using (10), (11), (12) and other related equations.
5. Repeat step 4 for the next time increment until the convergence criteria is satisfied.
6. Use 3D Matrix to represent temperature distribution in the domain.
7. Display the solution result in tabular form or as graphs.

Appendix B: MATLAB code for the numerical method

```
%heat3d
clear all; close all; clc;
T=1;
dt=0.0001; %Von Neumann stability analysis:For dx=dy=dz=h, take dt<=h^2/6
Nt=T/dt;
t=(1:Nt)*dt;
Nx=32;Ny=32;Nz=32;
dx=1/Nx;dy=1/Ny;dz=1/Nz;
xc=dx/2:dx:(1-dx/2);
yc=dy/2:dy:(1-dy/2);
zc=dz/2:dz:(1-dz/2);
x=[0 xc 1];
y=[0 yc 1];
z=[0 zc 1];
u=zeros(Nx,Ny,Nz);
u2=zeros(Nx,Ny,Nz);
%Initial condition
tic
for k=1:Nz
    for j=1:Ny
        for i=1:Nx
            u(i,j,k)=sin((pi/3)*(xc(i)+yc(j)+zc(k)))+xc(i)*yc(j)*zc(k);
        end
    end
end
for K=1:Nt
    for k=1:Nz
        for j=1:Ny
            for i=1:Nx
        %Boundary Conditions at the faces
       uB(i,j)=exp(-pi^2*t(K)/3).*sin((pi/3)*(xc(i)+yc(j)));  %bottom face
       uT(i,j)=exp(-pi^2*t(K)/3).*sin((pi/3)*(xc(i)+yc(j)+1))+xc(i).*yc(j); %top face
       uF(i,k)=exp(-pi^2*t(K)/3).*sin((pi/3)*(xc(i)+zc(k)));              %front face
       uBC(i,k)=exp(-pi^2*t(K)/3).*sin((pi/3)*(xc(i)+zc(k)+1))+xc(i).*zc(k);%back face
       uL(j,k)=exp(-pi^2*t(K)/3).*sin((pi/3)*(yc(j)+zc(k)));              %left face
       uR(j,k)=exp(-pi^2*t(K)/3).*sin((pi/3)*(yc(j)+zc(k)+1))+yc(j).*zc(k); %right face
       %Boundary conditions at the edges
       uLBE(j)=exp(-pi^2*t(K)/3).*sin((pi/3)*(0+yc(j)));           %left bottom edge
       uRBE(j)=exp(-pi^2*t(K)/3).*sin((pi/3)*(1+yc(j)));           %right bottom edge
       uLTE(j)=exp(-pi^2*t(K)/3).*sin((pi/3)*(0+yc(j)+1));         %left top edge
       uRTE(j)=exp(-pi^2*t(K)/3).*sin((pi/3)*(1+yc(j)+1))+yc(j);   %right top edge
       uFLE(k)=exp(-pi^2*t(K)/3).*sin((pi/3)*(0+zc(k)));           %Front left edge
       uFRE(k)=exp(-pi^2*t(K)/3).*sin((pi/3)*(1+zc(k)));           %Front right edge
       uLBCE(k)=exp(-pi^2*t(K)/3).*sin((pi/3)*(1+zc(k)));          %left back edge
       uRBCE(k)=exp(-pi^2*t(K)/3).*sin((pi/3)*(1+zc(k)+1))+zc(k);  %Right back edge
            end
        end
    end
    % Boundary conditions at the edges
    for k=1:Nz+2
        for j=1:Ny+2
```

```
            for i=1:Nx+2
             uFBE(i)=exp(-pi^2*t(K)/3).*sin((pi/3)*(x(i)+0)); %Front bottom edge
             uBCBE(i)=exp(-pi^2*t(K)/3).*sin((pi/3)*(x(i)+1)); %back bottom edge
             uFTE(i)=exp(-pi^2*t(K)/3).*sin((pi/3)*(x(i)+0+1)); %Front top edge
             uBCTE(i)=exp(-pi^2*t(K)/3).*sin((pi/3)*(x(i)+1+1))+x(i);%Back top edge
            end
        end
    end
    for k=1:Nz
        for j=1:Ny
            for i=1:Nx
                if i==1
                    if j==1
                        if k==1
                          u2(i,j,k)=dt*((u(i+1,j,k)-u(i,j,k))/(dx^2)-2*(u(i,j,k)-uL(j,k))/(dx^2) ...
                            +(u(i,j+1,k)-u(i,j,k))/(dy^2)-2*(u(i,j,k)-uF(i,k))/(dy^2) ...
                            +(u(i,j,k+1)-u(i,j,k))/(dz^2)-2*(u(i,j,k)-uB(i,j))/(dz^2))+u(i,j,k);
                        elseif k==Nz
                          u2(i,j,k)=dt*((u(i+1,j,k)-u(i,j,k))/(dx^2)-2*(u(i,j,k)-uL(j,k))/(dx^2) ...
                            +(u(i,j+1,k)-u(i,j,k))/(dy^2)-2*(u(i,j,k)-uF(i,k))/(dy^2) ...
                            +2*(uT(i,j)-u(i,j,k))/(dz^2)-(u(i,j,k)-u(i,j,k-1))/(dz^2))+u(i,j,k);
                        else
                          u2(i,j,k)=dt*((u(i+1,j,k)-u(i,j,k))/(dx^2)-2*(u(i,j,k)-uL(j,k))/(dx^2) ...
                            +(u(i,j+1,k)-u(i,j,k))/(dy^2)-2*(u(i,j,k)-uF(i,k))/(dy^2) ...
                            +(u(i,j,k+1)-u(i,j,k))/(dz^2)-(u(i,j,k)-u(i,j,k-1))/(dz^2))+u(i,j,k);
                        end
                    elseif j==Ny
                        if k==1
                          u2(i,j,k)=dt*((u(i+1,j,k)-u(i,j,k))/(dx^2)-2*(u(i,j,k)-uL(j,k))/(dx^2) ...
                            +2*(uBC(i,k)-u(i,j,k))/(dy^2)-(u(i,j,k)-u(i,j-1,k))/(dy^2) ...
                            +(u(i,j,k+1)-u(i,j,k))/(dz^2)-2*(u(i,j,k)-uB(i,j))/(dz^2))+u(i,j,k);
                        elseif k==Nz
                          u2(i,j,k)=dt*((u(i+1,j,k)-u(i,j,k))/(dx^2)-2*(u(i,j,k)-uL(j,k))/(dx^2)...
                            +2*(uBC(i,k)-u(i,j,k))/(dy^2)-(u(i,j,k)-u(i,j-1,k))/(dy^2) ...
                            +2*(uT(i,j)-u(i,j,k))/(dz^2)-(u(i,j,k)-u(i,j,k-1))/(dz^2))+u(i,j,k);
                        else
                          u2(i,j,k)=dt*((u(i+1,j,k)-u(i,j,k))/(dx^2)-2*(u(i,j,k)-uL(j,k))/(dx^2)...
                            +2*(uBC(i,k)-u(i,j,k))/(dy^2)-(u(i,j,k)-u(i,j-1,k))/(dy^2) ...
                            +(u(i,j,k+1)-u(i,j,k))/(dz^2)-(u(i,j,k)-u(i,j,k-1))/(dz^2))+u(i,j,k);
                        end
                    else
                        if k==1
                          u2(i,j,k)=dt*((u(i+1,j,k)-u(i,j,k))/(dx^2)-2*(u(i,j,k)-uL(j,k))/(dx^2)...
                            +(u(i,j+1,k)-u(i,j,k))/(dy^2)-(u(i,j,k)-u(i,j-1,k))/(dy^2) ...
                            +(u(i,j,k+1)-u(i,j,k))/(dz^2)-2*(u(i,j,k)-uB(i,j))/(dz^2))+u(i,j,k);
                        elseif k==Nz
                          u2(i,j,k)=dt*((u(i+1,j,k)-u(i,j,k))/(dx^2)-2*(u(i,j,k)-uL(j,k))/(dx^2)...
                            +(u(i,j+1,k)-u(i,j,k))/(dy^2)-(u(i,j,k)-u(i,j-1,k))/(dy^2) ...
                            +2*(uT(i,j)-u(i,j,k))/(dz^2)-(u(i,j,k)-u(i,j,k-1))/(dz^2))+u(i,j,k);
                        else
                          u2(i,j,k)=dt*((u(i+1,j,k)-u(i,j,k))/(dx^2)-2*(u(i,j,k)-uL(j,k))/(dx^2)...
                            +(u(i,j+1,k)-u(i,j,k))/(dy^2)-(u(i,j,k)-u(i,j-1,k))/(dy^2) ...
                            +(u(i,j,k+1)-u(i,j,k))/(dz^2)-(u(i,j,k)-u(i,j,k-1))/(dz^2))+u(i,j,k);
                     end
                    end
                elseif i==Nx
                    if j==1
                        if k==1
                          u2(i,j,k)=dt*(2*(uR(j,k)-u(i,j,k))/(dx^2)-(u(i,j,k)-u(i-1,j,k))/(dx^2)...
                            +(u(i,j+1,k)-u(i,j,k))/(dy^2)-2*(u(i,j,k)-uF(i,k))/(dy^2) ...
                            +(u(i,j,k+1)-u(i,j,k))/(dz^2)-2*(u(i,j,k)-uB(i,j))/(dz^2))+u(i,j,k);
                        elseif k==Nz
                          u2(i,j,k)=dt*(2*(uR(j,k)-u(i,j,k))/(dx^2)-(u(i,j,k)-u(i-1,j,k))/(dx^2)...
                            +(u(i,j+1,k)-u(i,j,k))/(dy^2)-2*(u(i,j,k)-uF(i,k))/(dy^2) ...
                            +2*(uT(i,j)-u(i,j,k))/(dz^2)-(u(i,j,k)-u(i,j,k-1))/(dz^2))+u(i,j,k);
                        else
                          u2(i,j,k)=dt*(2*(uR(j,k)-u(i,j,k))/(dx^2)-(u(i,j,k)-u(i-1,j,k))/(dx^2)...
                            +(u(i,j+1,k)-u(i,j,k))/(dy^2)-2*(u(i,j,k)-uF(i,k))/(dy^2) ...
                            +(u(i,j,k+1)-u(i,j,k))/(dz^2)-(u(i,j,k)-u(i,j,k-1))/(dz^2))+u(i,j,k);
                        end
                    elseif j==Ny
                        if k==1
                          u2(i,j,k)=dt*(2*(uR(j,k)-u(i,j,k))/(dx^2)-(u(i,j,k)-u(i-1,j,k))/(dx^2)...
                            +2*(uBC(i,k)-u(i,j,k))/(dy^2)-(u(i,j,k)-u(i,j-1,k))/(dy^2) ...
                            +(u(i,j,k+1)-u(i,j,k))/(dz^2)-2*(u(i,j,k)-uB(i,j))/(dz^2))+u(i,j,k);
                        elseif k==Nz
                          u2(i,j,k)=dt*(2*(uR(j,k)-u(i,j,k))/(dx^2)-(u(i,j,k)-u(i-1,j,k))/(dx^2)...
                            +2*(uBC(i,k)-u(i,j,k))/(dy^2)-(u(i,j,k)-u(i,j-1,k))/(dy^2) ...
                            +2*(uT(i,j)-u(i,j,k))/(dz^2)-(u(i,j,k)-u(i,j,k-1))/(dz^2))+u(i,j,k);
                        else
                          u2(i,j,k)=dt*(2*(uR(j,k)-u(i,j,k))/(dx^2)-(u(i,j,k)-u(i-1,j,k))/(dx^2)...
                            +2*(uBC(i,k)-u(i,j,k))/(dy^2)-(u(i,j,k)-u(i,j-1,k))/(dy^2) ...
                            +(u(i,j,k+1)-u(i,j,k))/(dz^2)-(u(i,j,k)-u(i,j,k-1))/(dz^2))+u(i,j,k);
                        end
                 else
                        if k==1
```

```
                    u2(i,j,k)=dt*(2*(uR(j,k)-u(i,j,k))/(dx^2)-(u(i,j,k)-u(i-1,j,k))/(dx^2)...
                      +(u(i,j+1,k)-u(i,j,k))/(dy^2)-(u(i,j,k)-u(i,j-1,k))/(dy^2) ...
                      +(u(i,j,k+1)-u(i,j,k))/(dz^2)-2*(u(i,j,k)-uB(i,j))/(dz^2))+u(i,j,k);
                elseif k==Nz
                    u2(i,j,k)=dt*(2*(uR(j,k)-u(i,j,k))/(dx^2)-(u(i,j,k)-u(i-1,j,k))/(dx^2)...
                      +(u(i,j+1,k)-u(i,j,k))/(dy^2)-(u(i,j,k)-u(i,j-1,k))/(dy^2) ...
                      +2*(uT(i,j)-u(i,j,k))/(dz^2)-(u(i,j,k)-u(i,j,k-1))/(dz^2))+u(i,j,k);
                else
                    u2(i,j,k)=dt*(2*(uR(j,k)-u(i,j,k))/(dx^2)-(u(i,j,k)-u(i-1,j,k))/(dx^2)...
                      +(u(i,j+1,k)-u(i,j,k))/(dy^2)-(u(i,j,k)-u(i,j-1,k))/(dy^2) ...
                      +(u(i,j,k+1)-u(i,j,k))/(dz^2)-(u(i,j,k)-u(i,j,k-1))/(dz^2))+u(i,j,k);
                end
            end
        else
            if j==1
                if k==1
                    u2(i,j,k)=dt*((u(i+1,j,k)-u(i,j,k))/(dx^2)-(u(i,j,k)-u(i-1,j,k))/(dx^2)...
                      +(u(i,j+1,k)-u(i,j,k))/(dy^2)-2*(u(i,j,k)-uF(i,k))/(dy^2) ...
                      +(u(i,j,k+1)-u(i,j,k))/(dz^2)-2*(u(i,j,k)-uB(i,j))/(dz^2))+u(i,j,k);
                elseif k==Nz
                    u2(i,j,k)=dt*((u(i+1,j,k)-u(i,j,k))/(dx^2)-(u(i,j,k)-u(i-1,j,k))/(dx^2)...
                      +(u(i,j+1,k)-u(i,j,k))/(dy^2)-2*(u(i,j,k)-uF(i,k))/(dy^2) ...
                      +2*(uT(i,j)-u(i,j,k))/(dz^2)-(u(i,j,k)-u(i,j,k-1))/(dz^2))+u(i,j,k);
                else
                    u2(i,j,k)=dt*((u(i+1,j,k)-u(i,j,k))/(dx^2)-(u(i,j,k)-u(i-1,j,k))/(dx^2)...
                      +(u(i,j+1,k)-u(i,j,k))/(dy^2)-2*(u(i,j,k)-uF(i,k))/(dy^2) ...
                      +(u(i,j,k+1)-u(i,j,k))/(dz^2)-(u(i,j,k)-u(i,j,k-1))/(dz^2))+u(i,j,k);
             end
            elseif j==Ny
                if k==1
                    u2(i,j,k)=dt*((u(i+1,j,k)-u(i,j,k))/(dx^2)-(u(i,j,k)-u(i-1,j,k))/(dx^2)...
                      +2*(uBC(i,k)-u(i,j,k))/(dy^2)-(u(i,j,k)-u(i,j-1,k))/(dy^2) ...
                      +(u(i,j,k+1)-u(i,j,k))/(dz^2)-2*(u(i,j,k)-uB(i,j))/(dz^2))+u(i,j,k);
                elseif k==Nz
                    u2(i,j,k)=dt*((u(i+1,j,k)-u(i,j,k))/(dx^2)-(u(i,j,k)-u(i-1,j,k))/(dx^2)...
                      +2*(uBC(i,k)-u(i,j,k))/(dy^2)-(u(i,j,k)-u(i,j-1,k))/(dy^2) ...
                      +2*(uT(i,j)-u(i,j,k))/(dz^2)-(u(i,j,k)-u(i,j,k-1))/(dz^2))+u(i,j,k);
                else
                    u2(i,j,k)=dt*((u(i+1,j,k)-u(i,j,k))/(dx^2)-(u(i,j,k)-u(i-1,j,k))/(dx^2)...
                      +2*(uBC(i,k)-u(i,j,k))/(dy^2)-(u(i,j,k)-u(i,j-1,k))/(dy^2) ...
                      +(u(i,j,k+1)-u(i,j,k))/(dz^2)-(u(i,j,k)-u(i,j,k-1))/(dz^2))+u(i,j,k);
                end
            else
                if k==1
                    u2(i,j,k)=dt*((u(i+1,j,k)-u(i,j,k))/(dx^2)-(u(i,j,k)-u(i-1,j,k))/(dx^2)...
                      +(u(i,j+1,k)-u(i,j,k))/(dy^2)-(u(i,j,k)-u(i,j-1,k))/(dy^2) ...
                      +(u(i,j,k+1)-u(i,j,k))/(dz^2)-2*(u(i,j,k)-uB(i,j))/(dz^2))+u(i,j,k);
                elseif k==Nz
                    u2(i,j,k)=dt*((u(i+1,j,k)-u(i,j,k))/(dx^2)-(u(i,j,k)-u(i-1,j,k))/(dx^2)...
                      +(u(i,j+1,k)-u(i,j,k))/(dy^2)-(u(i,j,k)-u(i,j-1,k))/(dy^2) ...
                      +2*(uT(i,j)-u(i,j,k))/(dz^2)-(u(i,j,k)-u(i,j,k-1))/(dz^2))+u(i,j,k);
                else
                    u2(i,j,k)=dt*((u(i+1,j,k)-u(i,j,k))/(dx^2)-(u(i,j,k)-u(i-1,j,k))/(dx^2)...
                      +(u(i,j+1,k)-u(i,j,k))/(dy^2)-(u(i,j,k)-u(i,j-1,k))/(dy^2) ...
                      +(u(i,j,k+1)-u(i,j,k))/(dz^2)-(u(i,j,k)-u(i,j,k-1))/(dz^2))+u(i,j,k);
                end
            end
        end
    end
  end
end
U=zeros(Nx+2,Ny+2,Nz+2);
U(2:Nx+1,2:Ny+1,2:Nz+1)=u2;
U(2:Nx+1,2:Ny+1,1)=uB;
U(2:Nx+1,2:Ny+1,Nz+2)=uT;
U(1,2:Ny+1,2:Nz+1)=uL;
U(Nx+2,2:Ny+1,2:Nz+1)=uR;
U(2:Nx+1,1,2:Nz+1)=uF;
U(2:Nx+1,Ny+2,2:Nz+1)=uBC;
U(1:Nx+2,1,1)=uFBE;
U(1:Nx+2,Ny+2,1)=uBCBE;
U(1:Nx+2,1,Nz+2)=uFTE;
U(1:Nx+2,Ny+2,Nz+2)=uBCTE;
U(1,2:Ny+1,1)=uLBE;
U(Nx+2,2:Ny+1,1)=uRBE;
U(1,2:Ny+1,Nz+2)=uLTE;
U(Nx+2,2:Ny+1,Nz+2)=uRTE;
U(1,1,2:Nz+1)=uFLE;
U(Nx+2,1,2:Nz+1)=uFRE;
U(1,Ny+2,2:Nz+1)=uLBCE;
U(Nx+2,Ny+2,2:Nz+1)=uRBCE;
u=u2;
%Exact solution
uexa=zeros(Nx+2,Ny+2,Nz+2);
```

```
    for k=1:Nz+2
        for j=1:Ny+2
            for i=1:Nx+2
uexa(i,j,k)=exp(-pi^2*t(K)/3).*sin((pi/3)*(x(i)+y(j)+z(k)))+x(i)*y(j)*z(k);
            end
        end
    end
end
toc
% U
% uexa
%[Xc,Yc,Zc]=meshgrid(xc,yc,zc);
%slice(X,Y,Z,u2,xc,yc,zc)
[X,Y,Z]=meshgrid(x,y,z);
% figure % Finite volume solution
subplot(1,2,1)
slice(X,Y,Z,U,x,y,z);
xlabel('x');
ylabel('y');
zlabel('z')
view(25,26)
colormap(jet)
cb=colorbar;
cb.Label.String='Temperature';
title('Finite Volume Solution')
shading interp
% figure %Exact solution
subplot(1,2,2)
slice(X,Y,Z,uexa,x,y,z);
xlabel('x');
ylabel('y');
zlabel('z')
view(25,26)
colormap(jet)
cb=colorbar;
cb.Label.String='Temperature';
title('Exact Solution')
shading interp
figure % Along the vertical central line
uxmid=(U(:,(Ny/2)+1,(Nz/2)+1)+U(:,(Ny/2)+2,(Nz/2)+2))/2;
uzmid=(U((Nx/2)+1,(Ny/2)+1,:)+U((Nx/2)+2,(Ny/2)+2,:))/2;
uexaxmid=(uexa(:,(Ny/2)+1,(Nz/2)+1)+uexa(:,(Ny/2)+2,(Nz/2)+2))/2;
uexazmid=(uexa((Nx/2)+1,(Ny/2)+1,:)+uexa((Nx/2)+2,(Ny/2)+2,:))/2;
for k=1:Nz+2;
    uzmid1(k)=uzmid(k);
    uexazmid1(k)=uexazmid(k);
end
err=abs(uexazmid1-uzmid1);
plot(z,uzmid1,'r.-',z,uexazmid1,'bo-');
legend('FV solution','Exact solution','Location','NorthWest');
xlabel('z')
ylabel('Temperature')
% Table of values
fprintf(' z       FV solution     Exact solution      error\n')
for k=1:Nz+2
    fprintf('%2.5f    %2.8f      %2.8f      %2.8f \n',z(k),uzmid1(k),uexazmid1(k),err(k))
end
```