Indian Institute of Technology, Madras

Foundations of Computational Fluid Dynamics - AM5630

Steady 2D Diffusion - Heat Conduction

Kanak Agarwal

September 28, 2025

1 Introduction

The 2D diffusion equation i.e., the 2D steady heat conduction heat conduction is given by,

$$\frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(k\frac{\partial T}{\partial y}\right)+S=0 \tag{1}$$

Discretizing the above equation and taking a control volume as depicted in Fig. 1,

$$\int_s^n\int_w^e \frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right)+\int_s^n\int_w^e \frac{\partial}{\partial y}\left(k\frac{\partial T}{\partial y}\right)+\int_s^n\int_w^e S dxdy=0 \tag{2}$$

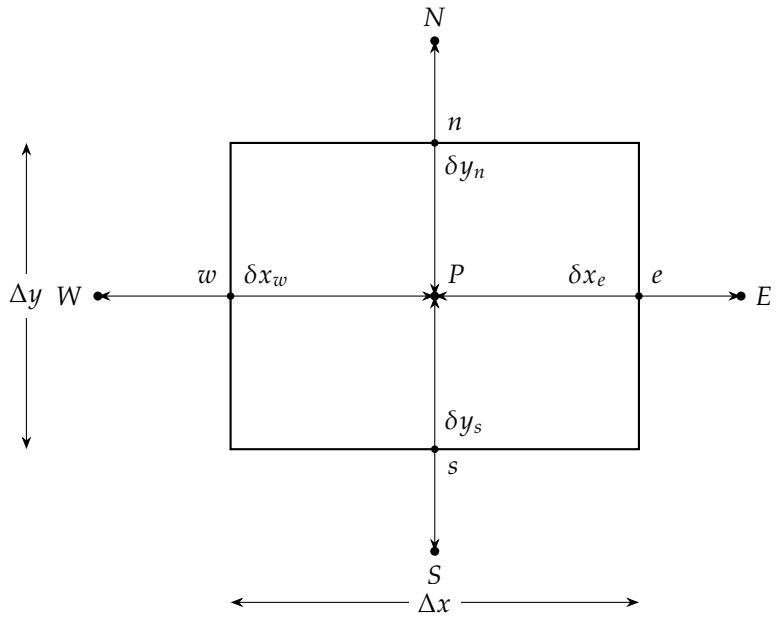


Figure 1: Typical Control Volume in Two-Dimensional Finite Volume Discretization

Integrating,

$$\int_s^n\left[\left(k\frac{\partial T}{\partial x}\right)_e-\left(k\frac{\partial T}{\partial x}\right)_w\right]dy+\int_w^e\left[\frac{\partial}{\partial y}\left(k\frac{\partial T}{\partial y}\right)_n-\frac{\partial}{\partial y}\left(k\frac{\partial T}{\partial y}\right)_s\right]dy+\overline{S}\Delta x\Delta y+S_P T_P=0 \tag{3}$$

Approximating the integrals using first order approximation,

$$\int_a^b f(y)dy = f_p \Delta y + O(y) \tag{4}$$

Therefore,

$$\left[k_e\left(\frac{T_E - T_P}{\delta x_e}\right) - k_w\left(\frac{T_P - T_W}{\delta x_w}\right)\right]\Delta y + \left[k_n\left(\frac{T_N - T_P}{\delta y_n}\right) - k_s\left(\frac{T_P - T_S}{\delta y_s}\right)\right]\Delta x + \overline{S}\Delta x\Delta y + S_P T_P = 0 \tag{5}$$

Simplifying,

$$\left(\frac{k_e}{\delta x_e}\Delta y + \frac{k_w}{\delta x_w}\Delta y + \frac{k_n}{\delta y_n}\Delta x + \frac{k_s}{\delta y_s}\Delta x - S_P\right)T_P = \frac{k_e}{\delta x_e}\Delta y T_E + \frac{k_w}{\delta x_w}\Delta y T_W + \frac{k_n}{\delta y_n}\Delta x T_N + \frac{k_s}{\delta y_s}\Delta x T_S + \overline{S}\Delta x\Delta y \tag{6}$$

$$a_P T_P = a_E T_E + a_W T_W + a_N T_N + a_S T_S + S_u \tag{7}$$

2 Code Architecture

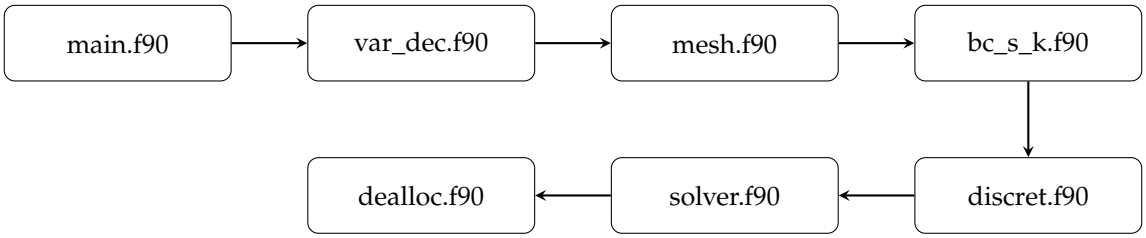


Figure 2: Code Architecture

The architecture of the code developed is as depicted in Fig. 2. The program is initiated by *main.f90*, this in turn calls the *var_dec.f90* module which declares all the variables, arrays and the precision to be used by the code. After this the meshing subroutine, *mesh.f90* is called which reads the relevant domain and mesh parameters from the input file *bc_s_k.txt* and allocates and initialises the mesh arrays. Post-meshing the boundary conditions, the thermal conductivities and the source terms are allocated and initialised by the *bc_s_k.f90* subroutine which in turn reads these from the input file.

In order to evaluate the mathematical expressions for these inputs they are inputted as strings and evaluated using the string parsing code, *fparser* (v1.1.0) developed by Roland Schmehl. Once evaluated, these arrays are calculated and the boundary conditions are incorporated (Neumann and Dirichlet boundary conditions are appropriately handled). Further, the discretisation subroutine *discret.f90* is invoked which calculates the coefficents of the dicretised equation througout the domain. This is followed then by the solver subroutine, *solver.f90* which incorporates a Gauss-Seidel solver and checks for the residual at each iteration against that prescribed by the user. Finally, the *dealloc.f90* subroutine deallocates all the allocated arrays.

Few additional utilities have been also developed. A *makefile* is used to make the executable. Custom bash scripts to plot both the residuals and the mesh have been developed using *gnuplot*. Further the temperature and heat flux fields are written to a *.tec* file and the post-processing is carried out in ParaView. Aditionally, the grid independence is carried out in Python (Jupyter notebook - *post.ipynb*).

3 Results

3.1 Computational Domain and Initial Setup

The computational domain is as follows,

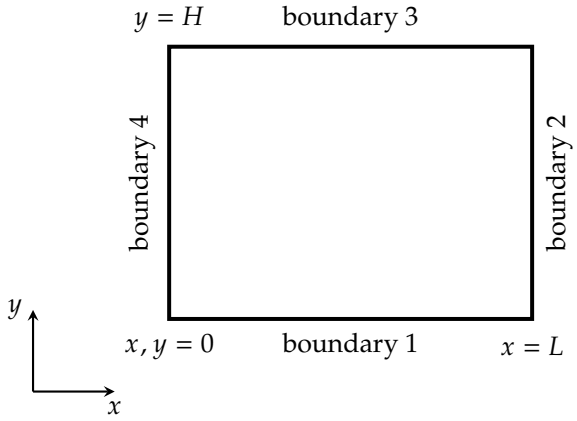


Figure 3: Computational Domain

The boundary conditions (case 4) are as follows,

$$
\begin{aligned}
T1 &= 15 \\
T2 &= 5(1 - y/H) + 15\sin(\pi y/H) \\
T3 &= 10 \\
T4 &\rightarrow q = -5000 \\
S &= -1.5 \\
k &= 16(y/H + 1) \\
L &= 1 \\
H &= 0.5
\end{aligned}
$$

3.2 Grid Independence

Grid independence is established in following fashion. The relative $\ell 2$ norm of the difference in the global temperature field is calculated. This is given by,

$$
\|T^{(h)} - T^{(h/2)}\|_2 = \sqrt{\sum_{i=1}^{N_x}\sum_{j=1}^{N_y}\left(T_{i,j}^{(h)} - T_{i,j}^{(h/2)}\right)^2}
$$

$$
\text{Relative error} = \frac{\|T^{(h)} - T^{(h/2)}\|_2}{\|T^{(h/2)}\|_2} \quad \text{where} \quad \|T^{(h/2)}\|_2 = \sqrt{\sum_{i=1}^{N_x}\sum_{j=1}^{N_y}\left(T_{i,j}^{(h/2)}\right)^2}
$$

In order to achieve a one to one mapping the values at the equivalent points on the coarser grid are linearly interpolated. Further, the relative error between the volume weighted average heat fluxes in the x and y directions and the maximum and minimum temperatures are also calculated. The

Python implementation of this can be found in the Jupyter notebook *post.ipynb,* this was carried out for a coarsest grid resolution of 10×5 and a finest grid resolution of 320×160. Beyond this it is significantly expensive to run the simulation serially. It is clear from the outputs of the Python implementation that this is not enough to produce a grid independent solution.

Python Output ($\ell 2$ Norm of the Temperature field):

```
Found files: ['output_1.tec', 'output_2.tec', 'output_3.tec', 'output_4.tec', 'output_5.tec',
↪  'output_6.tec']

Comparing output_1.tec and output_2.tec:
L2 norm of temperature difference: 46.457990
Relative error: 0.262675

Comparing output_2.tec and output_3.tec:
L2 norm of temperature difference: 60.757297
Relative error: 0.193587

Comparing output_3.tec and output_4.tec:
L2 norm of temperature difference: 356.475877
Relative error: 0.591342

Comparing output_4.tec and output_5.tec:
L2 norm of temperature difference: 2831.803373
Relative error: 0.834313

Comparing output_5.tec and output_6.tec:
L2 norm of temperature difference: 22503.284112
Relative error: 0.775931
```

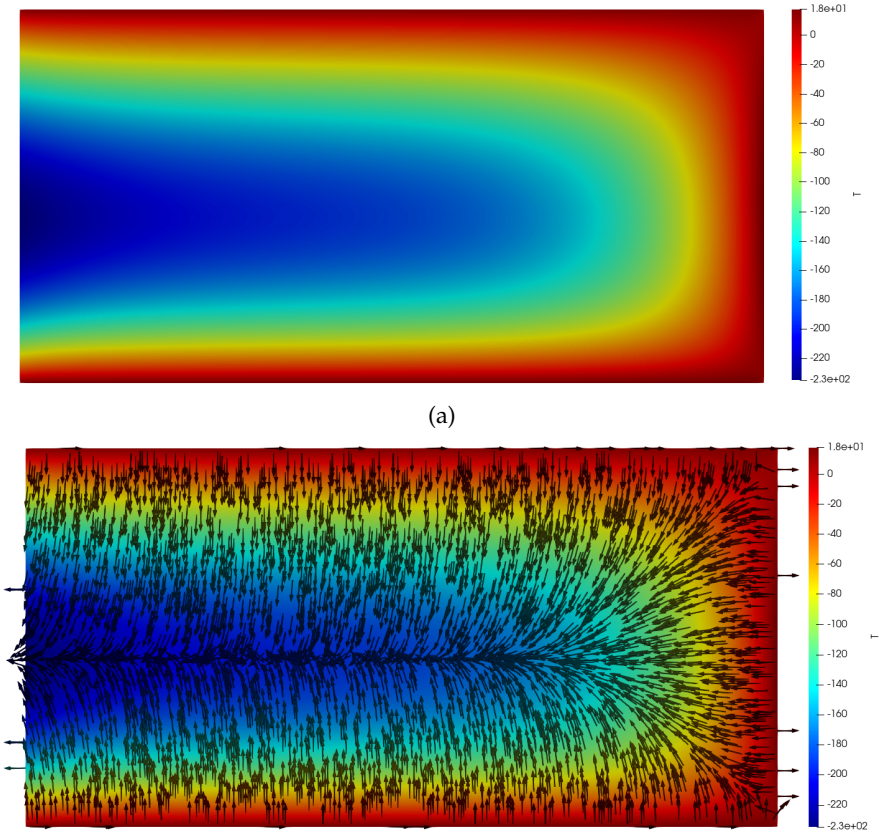
Python Output (Volume Weighted Average of the Heat fluxes in x and y):

```
Found files: ['output_1.tec', 'output_2.tec', 'output_3.tec', 'output_4.tec', 'output_5.tec',
↪  'output_6.tec']
output_1.tec: qx_avg = -584.338833, qy_avg = 107.751538
output_2.tec: qx_avg = -627.693496, qy_avg = 91.598172
output_3.tec: qx_avg = -692.977578, qy_avg = 43.494311
output_4.tec: qx_avg = -862.199877, qy_avg = -132.759848
output_5.tec: qx_avg = -1478.664791, qy_avg = -828.644369
output_6.tec: qx_avg = -3911.038159, qy_avg = -3607.608115

Comparing consecutive outputs:
Between output_1.tec and output_2.tec:
  delta qx = -43.354662, relative change = 6.907%
  delta qy = -16.153366, relative change = 17.635%
Between output_2.tec and output_3.tec:
  delta qx = -65.284082, relative change = 9.421%
  delta qy = -48.103861, relative change = 110.598%
Between output_3.tec and output_4.tec:
  delta qx = -169.222299, relative change = 19.627%
  delta qy = -176.254159, relative change = 132.762%
Between output_4.tec and output_5.tec:
  delta qx = -616.464914, relative change = 41.691%
  delta qy = -695.884521, relative change = 83.979%
Between output_5.tec and output_6.tec:
  delta qx = -2432.373368, relative change = 62.193%
  delta qy = -2778.963746, relative change = 77.031%
```

Python Output (Relative Error between Maximum and Minimum Temperatures):

```
Found files: ['output_1.tec', 'output_2.tec', 'output_3.tec', 'output_4.tec', 'output_5.tec',
↪  'output_6.tec']

Relative errors between consecutive files:
Between output_1.tec and output_2.tec:
  Max temp relative error: 0.3719% (from 17.500000 to 17.565330)
  Min temp relative error: 15.7406% (from -20.989000 to -24.909990)
Between output_2.tec and output_3.tec:
  Max temp relative error: 0.0764% (from 17.565330 to 17.578760)
  Min temp relative error: 14.7071% (from -24.909990 to -29.205240)
Between output_3.tec and output_4.tec:
  Max temp relative error: 0.0271% (from 17.578760 to 17.583530)
  Min temp relative error: 26.9466% (from -29.205240 to -39.977920)
Between output_4.tec and output_5.tec:
  Max temp relative error: 0.0028% (from 17.583530 to 17.584020)
  Min temp relative error: 49.6932% (from -39.977920 to -79.468250)
Between output_5.tec and output_6.tec:
  Max temp relative error: 0.0027% (from 17.584020 to 17.584490)
  Min temp relative error: 66.1676% (from -79.468250 to -234.888200)
```

The significant difference in the respective quantities can be attributed to the increase in the magnitude of the minimum temperature. As the grid resolution increases, the gradients near the left boundary (negative flux boundary condition) are captured in greater detail hence causing this increase in error. Thus to acheive a grid independent solution using a uniform mesh will be fairly expensive. This is why the grid must be stretched in this region. The results of the uniform mesh (320×160) are as as follows,



(a)

(b)

Figure 4: *(a)* Temperature Contour and *(b)* Heat Flux Vectors Overlayed for the Uniform Mesh

The residuals are defined as,

$$R = \sum_{i=2}^{n_x} \sum_{j=2}^{n_y} \frac{\left| a_P T_P - \left(a_E T_E + a_W T_W + a_N T_N + a_S T_S + s_u(i,j) \right) \right|}{s_u(i,j) \times l_x \times l_y}$$

and is given by,

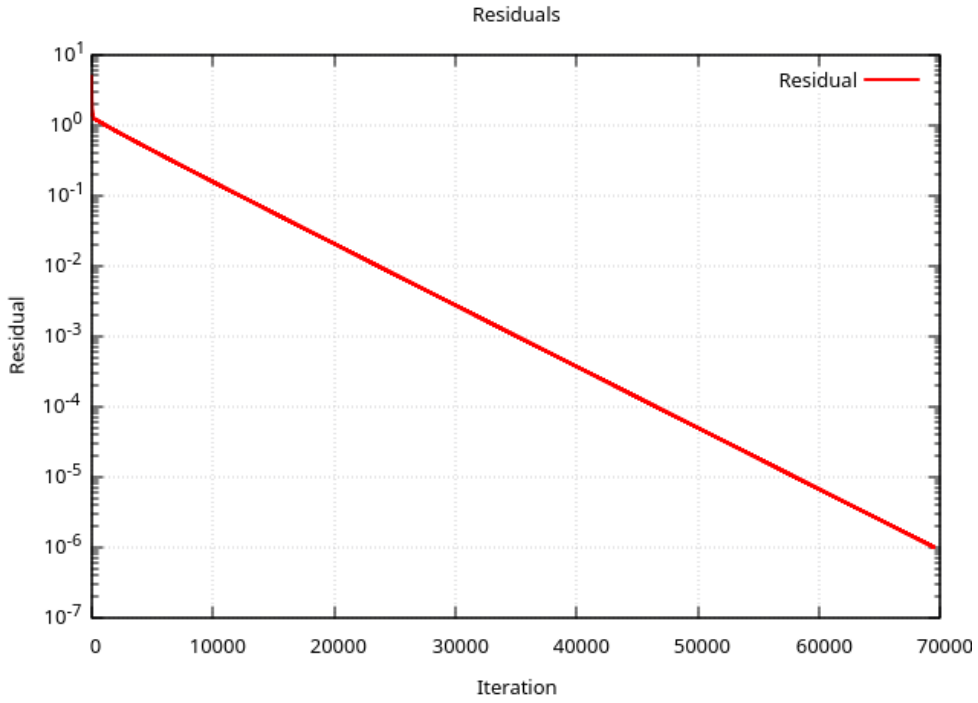


Figure 5: Residuals for a Uniform Mesh of 320×160

4 Miscellaneous

Upon reducing the tolerance the solver takes longer to converge and conversely shorter time to converge upon increasing the tolerance. The results of the modified boundary conditions as follows are presented in this section, these are given by,

$$\begin{aligned} T1 &= 15 \\ T2 &\longrightarrow q = 5000 \\ T3 &= 5 \\ T4 &\longrightarrow q = 5000 \\ S &= -1.5 \\ k &= 16(y/H + 1) \\ L &= 1 \\ H &= 0.5 \end{aligned}$$

The results of the uniform mesh (320×160) for these modified boundary conditions are as as follows. These results are expected given the Neumann boundary conditions that effectively act in the same

direction and the formation of the thermal boundary layer is also a standard phenomena.

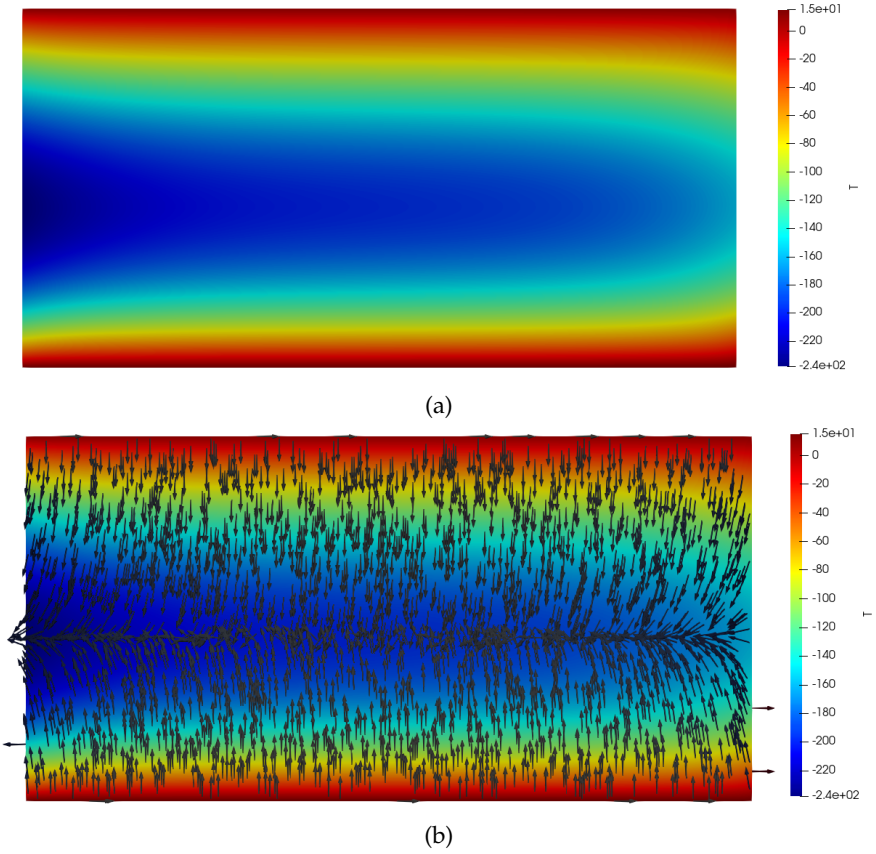


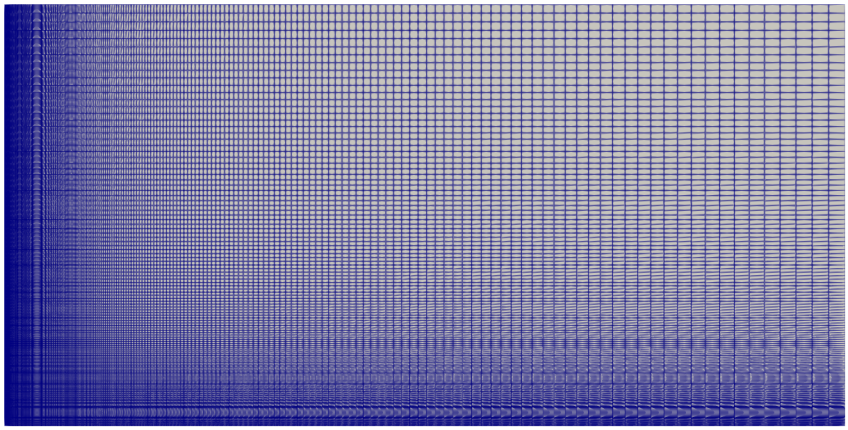
Figure 6: *(a)* Temperature Contour and *(b)* Heat Flux Vectors for the Uniform Mesh with modified Boundary Conditions

5 Stretched Mesh

The following boundary conditions were implemented on a stretched mesh (98%) of resolution 300×150,

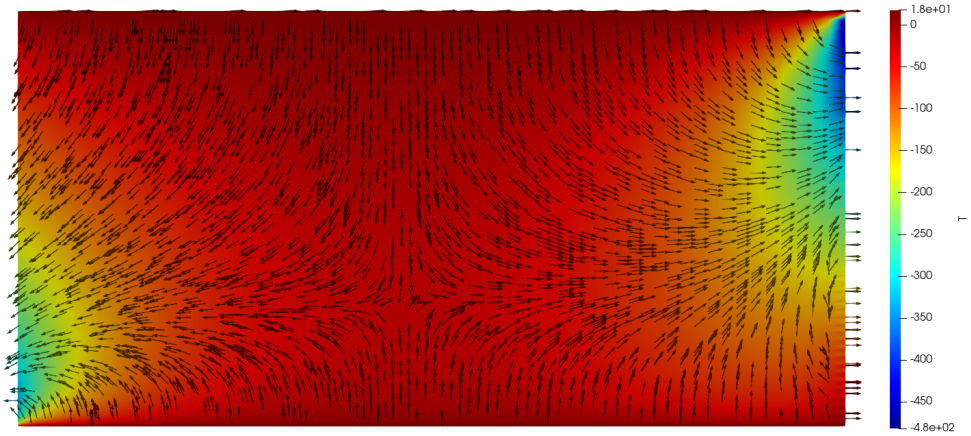
$$
\begin{aligned}
T1 &= 15 \\
T2 &= 5(1 - y/H) + 15\sin(\pi y/H) \\
T3 &= 10 \\
T4 &\longrightarrow q = -5000 \\
S &= -1.5 \\
k &= 16(y/H + 1) \\
L &= 1 \\
H &= 0.5
\end{aligned}
$$

The results vary significantly compared to the uniform mesh and the simulation took longer to converge. The maximum temperature seems to be the same, but there is a significant reduction in the minimum temperature in the domain. This can be attributed to the increased resolution of the large gradients introduced by the negative heat flux boundary condition.



(a)

(b)



(c)

Figure 7: *(b)* The Mesh *(b)* Temperature Contour and *(c)* Heat Flux Vectors for the Stretched Mesh

6 References

- H. Versteeg and W. Malalasekera. *An Introduction to Computational Fluid Dynamics - The Finite Volume Method*, Longman Scientific & Technical, Harlow, England, 1st edition, 1995.
- Ferziger, J.H. and Peric, M. (2002) *Computational Methods for Fluid Dynamics*, 3rd Edition, Springer, Berlin.