Finite Volume Solution of Poisson's Equation

Karthik Reddy Lyathakula

North Carolina State University, Raleigh, United States

1. Introduction

Poisson's equation is a partial differential equation of elliptical type which is used to mathematically represent various phenomena of mechanical sciences, electrostatics and theoretical physics. For example flow past a NACA aerofoil can be assumed to be inviscid and this flow can be described by a Poisson's equation. Poisson's equation is a second order equation and most of the physics equations like Navier-Stokes, mass transport, heat transport equation contains second order differential terms. So, it is extremely important to understand various numerical methods that can be used to discretize and solve Poisson's equation before an attempt to solve transport problems.

In this project, Poisson's Equation is solved numerically using combination of spatial discretization techniques like 'thin layer' method, tangent / normal gradient decomposition method and integration methods like Point Jacobi(PJ), Incomplete LU Decomposition(ILU).

2. Discretization Methods

In this section, finite volume method to discretize the Poisson's equation and different methods of estimating the boundary fluxes is discussed. The Poisson's equation is given by

$$\bigtriangledown \cdot \bigtriangledown \phi = f(x, y) \tag{1}$$

Here ϕ is the dependent variable and f(x,y) is the source term. The source term used in this problem is

$$f(x, y) = exp(-35[(x - x0)^2 + (y - y0)^2]) \tag{2}$$

here x0 anf y0 is the area centroid of the domain.

The Poisson's equation is discretized by integrating with respect to volume and gauss divergence theorem is applied to convert the volume integral to surface.

$$\int_V \bigtriangledown \cdot \bigtriangledown \phi \, dV = \int_V f(x, y) \, dV \tag{3}$$

$$\oint_S \bigtriangledown \phi \cdot n \, dA = \int_V f(x, y) \, dV \tag{4}$$

To solve Eq. 4 over the simulation domain, the domain is divided into infinitesimal volumes

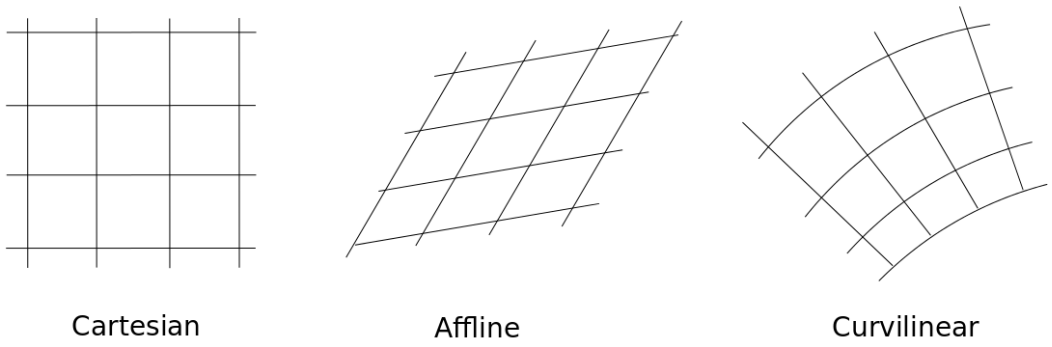


Figure 1: Different types of meshes

called mesh elements. Few examples of meshes are shown in figure 1.

The Eq. 4 is approximated in each cell as follows:

$$\sum_{k=1}^{N} \bigtriangledown\phi \cdot \overrightarrow{n_k} A_k \approx \overline{f(x,y)} V_k \tag{5}$$

Here $\overrightarrow{n}$ is the outward normal. In this project, quadrilateral mesh will be used and the value of N will be 4. From Eq. 5, the residue is defined as:

$$res = -\sum_{k=1}^{N} \bigtriangledown\phi \cdot \overrightarrow{n_k} A_k + \overline{f(x,y)} V_k \tag{6}$$

By applying Eq. 6 on a cell (i,j) as shown in the Fig. 2, the left hand term of the Eq. 5 is reduced to following expression:

$$\bigtriangledown\phi_1 \cdot \overrightarrow{n}_1 A_1 + \bigtriangledown\phi_2 \cdot \overrightarrow{n}_2 A_2 + \bigtriangledown\phi_3 \cdot \overrightarrow{n}_3 A_3 + \bigtriangledown\phi_4 \cdot \overrightarrow{n}_4 A_4 \tag{7}$$

$$F_{i+\frac{i}{2},j} - F_{i-\frac{i}{2},j} + G_{i,j+\frac{i}{2}} - G_{i,j--\frac{i}{2}} \tag{8}$$

Here F and G are the fluxes in x and y directions on the faces of a unite cell (i,j) respectively. These fluxes can be estimated using following methods:

1. Parallel CV method
2. Thin layer method
3. Tangent / normal gradient decomposition method

In this work, second and third methods are used to approximate the fluxes a mesh cell.

2.1. Thin Layer Method

In thin layer method, the fluxes on the cell faces of (i,j) are estimated by the normal flux.

$$\bigtriangledown\phi = (\bigtriangledown\phi \cdot \overrightarrow{n})\overrightarrow{n} \tag{9}$$

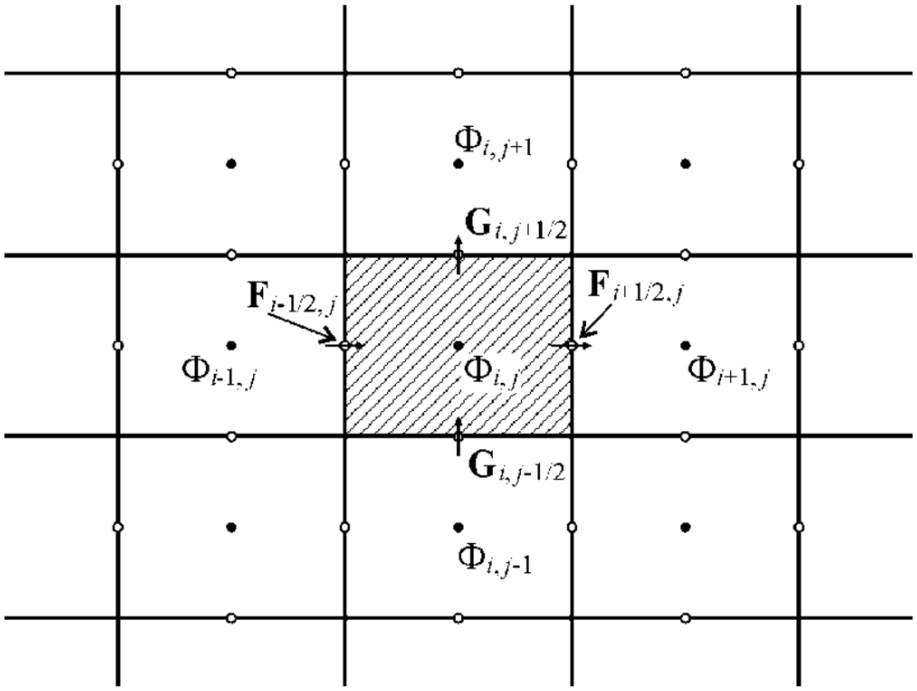


Figure 2: Fluxes on the faces of a mesh cell

$$\bigtriangledown\phi = \frac{\Delta u\phi}{\Delta n}\overrightarrow{n} \tag{10}$$

Here $\Delta\phi$ and Δn is given by:

$$\Delta\phi_1 = \phi_{i+1,j} - \phi_{i,j} \tag{11}$$

$$\Delta\phi_2 = \phi_{i,j+1} - \phi_{i,j} \tag{12}$$

$$\Delta\phi_3 = \phi_{i,j} - \phi_{i-1,j} \tag{13}$$

$$\Delta\phi_4 = \phi_{i,j} - \phi_{i,j-1} \tag{14}$$

$$\Delta n1 = \frac{Vol_{i+1,j} + Vol_{i,j}}{A_{i+1/2,j}} \tag{15}$$

$$\Delta n2 = \frac{Vol_{i,j+1} + Vol_{i,j}}{A_{i,j+1/2}} \tag{16}$$

$$\Delta n3 = \frac{Vol_{i-1,j} + Vol_{i,j}}{A_{i-1/2,j}} \tag{17}$$

$$\Delta n4 = \frac{Vol_{i,j-1} + Vol_{i,j}}{A_{i,j-1/2}} \tag{18}$$

The coefficients of the stiffness matrix 'A' and residue is calculated by substituting Eqn.

7-16 in Eqn. 4. Thin layer discretization is a five point stencil method and the solution on a particular cell (i,j) depends on the value of ϕ in cells (i,j) (i+1,j) (i,j+1) (i-1,j) (i,j-1).

2.2. Tangent/Normal Gradient Decomposition Method

In tangent/normal method, the gradient on the cell faces is estimated by:

$$\bigtriangledown\phi = (\bigtriangledown\phi \cdot \overrightarrow{\tau})\overrightarrow{n} + \overline{\bigtriangledown\phi} - (\overline{\bigtriangledown\phi} \cdot \tau)(\overrightarrow{\tau} \cdot \overrightarrow{n})\overrightarrow{n} \tag{19}$$

Consider the flux expression on the face (i+1/2,j) of the cell (i,j). The Definition of the different terms in the above expression is

$$\overrightarrow{\tau} = (x_c(i+1,j) - x_c(i,j))\overrightarrow{i} + (y_c(i+1,j) - y_c(i,j))\overrightarrow{j} \tag{20}$$

$$\overline{\bigtriangledown\phi} = 0.5 * (\bigtriangledown\phi_{i,j} + \bigtriangledown\phi_{i+1,j}) \tag{21}$$

$$\bigtriangledown\phi_{i,j} = (\phi\overrightarrow{n}A)_{i+1/2,j} + (\phi\overrightarrow{n}A)_{i,j+1/2} - (\phi\overrightarrow{n}A)_{i-1/2,j} - (\phi\overrightarrow{n}A)_{i,j-1/2} \tag{22}$$

In a similar way, the above terms are defined for each face respectively. The coefficients of the stiffness matrix 'A' and residue is calculated by substituting Eqn. 22 in Eqn. 4 for all the faces. This makes the tangent/normal method to be a thirteen point stencil method where solution in a cell depends on the values of ϕ in the cells (i-2,j), (i,j-2), (i-1,j-1), (i-1,j) (i,j-1), (i,j), (i+1,j-1), (i,j+1), (i+1,j), (i+1,j+1), (i,j+1), (i+1,j).

3. Integration Methods

Spatial discretization of the Poisson's equation gives set of algebraic equations as many as mesh cells. These algebraic equation can be arrange into system of matrices.

$$A\overrightarrow{\phi} = \overrightarrow{b} \tag{23}$$

Here, A is called stiffness matrix that contains the spatial discretization terms, $\overrightarrow{\phi}$ is a vector with values of ϕ at center of each mesh cell and $\overrightarrow{b}$ is a vector of source contribution in each cell. The size of the A matrix is NxN, where N is the number of mesh cells and size of $\overrightarrow{\phi}$, $\overrightarrow{b}$ is Nx1. The number of elements in each row of matrix A will depend on the type of discretization used. Thin layer discretization will have five elements and tangent/normal method will have 13 elements in each row of A.

The computation effort required to solve a system of matrices depends on the size and sparcity of the matrix 'A'. To solve this matrix system, the matrix A can be written as A = M+N, where M is the simple matrix and N is not a simple matrix.

Before solving the system of matrices, they are arranged into below form.

$$(M+N)\overrightarrow{\phi}_{k+1} = \overrightarrow{b}_k \tag{24}$$

$$M\overrightarrow{\phi}_{k+1} = \overrightarrow{b}_k - (N)\overrightarrow{\phi}_k \tag{25}$$

$$M\overrightarrow{\phi}_{k+1} - M\overrightarrow{\phi}_k = \overrightarrow{b}_k - (M+N)\overrightarrow{\phi}_k \tag{26}$$

$$M\overrightarrow{\Delta}\phi_{k+1} = \overrightarrow{R}_k \tag{27}$$

here k is the iteration number, $\overrightarrow{R}_k$ is the residue which also represents error measure. The residue is calculated for each method as described in the discretization section.

To solve these set of equations, following iterative methods are used:

1. Point Jacobi
2. Incomplete LU Decomposition

In Point Jacobi method, the matrix M is approximated as M=D, here D is the matrix containing only diagonal elements of the matrix A. In Incomplete LU decomposition, the M is approximated as:

$$M = (\widetilde{D} + L)\widetilde{D}^{-1}(M = (\widetilde{D} + U) \tag{28}$$

here L consists of lower diagonal elements and U consists of upper diagonal elements of the matrix A. The diagonal matrix $\widetilde{D}$ is approximated such that the matrix M is closer to original matrix A.

$$\widetilde{D} + U + L) + L\widetilde{D}^{-1}U = D + L + U \tag{29}$$

$$\widetilde{D} = D - L\widetilde{D}^{-1}U \tag{30}$$

The matrix $L\widetilde{D}^{-1}U$ is not sparse and it is approximated by ignoring its non-sparse elements so that the sparsity of the original matrix is still maintained.

These set of matrices solved iteratively using the Integration methods until the norm of the residue is smaller than prescribed tolerance. The norm of the residue is calculated by the formula:

$$||R|| = (\sum_{k=1}^{N} \overrightarrow{R}_k)^(1/2) \tag{31}$$

4. Results

The Poisson's equation is solved for all combinations of discretization methods and integration methods on three grids: Square grid 50x50, Square grid 100x100 and curvilinear grid as shown in Fig. 3-5. The boundary condition used for all the meshes is no flux on right and left boundaries and phi is prescribed to be 0 on the top and bottom boundary. A tolerance of 1e-4 is used for all the simulations. For this project, a C++ program with various subroutines is used to solve the Poisson's equation and Tecplot, MATLAB is used for plotting the results. The code is attached to the appendix.

Figure 3: Square Grid 50x50

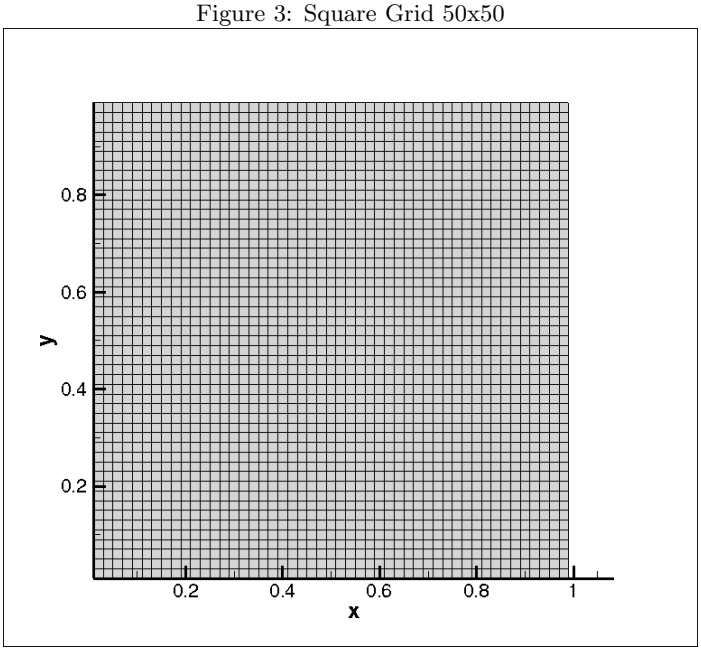
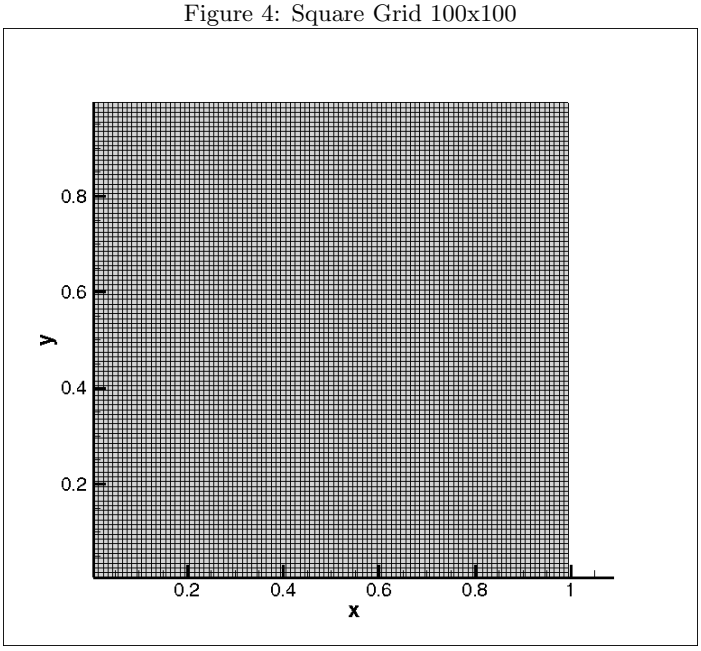


Figure 4: Square Grid 100x100



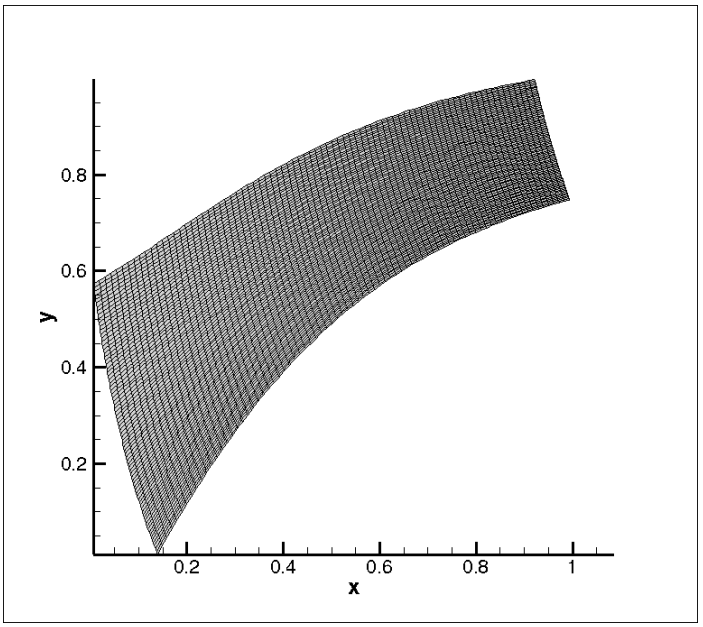


Figure 5: Curvilinear Grid

4.1. Initial Convergence Issues

Initially, the C++ program is executed using variables with float precision and it is found that the simulations are diverging for few cases like "square grid 100x100 - LU Decomposition" and "curvilinear grid- Point Jacobi". The convergence issue is investigated by executing the program with different tolerance values and it is found that the convergence problem is because of the truncation error induced by float precision makes the integration schemes to diverge. Fig. 6 shows the relative residue vs iteration number for two diverging cases. The relative residue for the "square grid 100x100 - LU Decomposition" and "curvilinear grid- Point Jacobi" is reaching the values of 1.3e-4 and 1e-3 respectively before stabilizing. To resolve this, the precision of all the variables is changed to double and further convergence issues are not observed.

4.2. Square grid 50x50

The subsequent simulations are conducted using double precision. Fig. 7 shows the convergence plot and Table. 1 shows the computational values for four different cases. The results shows that the ILU method converges quicker than PJ and takes less number of iterations. This is because the matrix M is closer to the original matrix A in ILU method and so the solution is arrived with less number of iterations.

Table 1: Number of Iterations and simulation time for square grid 50x50

Method	N: PJ	t(s):PJ	N: ILU	t(s):ILU
Thin Layer	8742	1.96	1347	0.68
Normal Tangent	8742	14.92	1347	2.66

Figure 6: Simulation using float precision

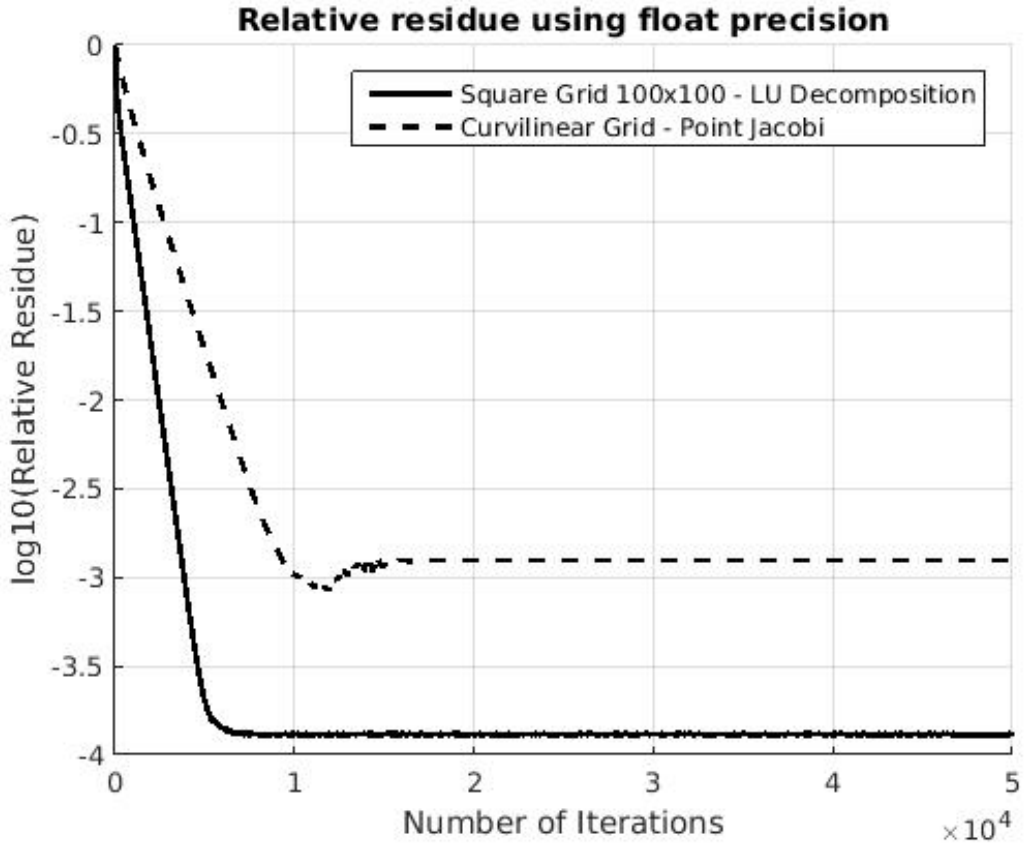


Figure 7: Convergence plot for Square grid 50x50

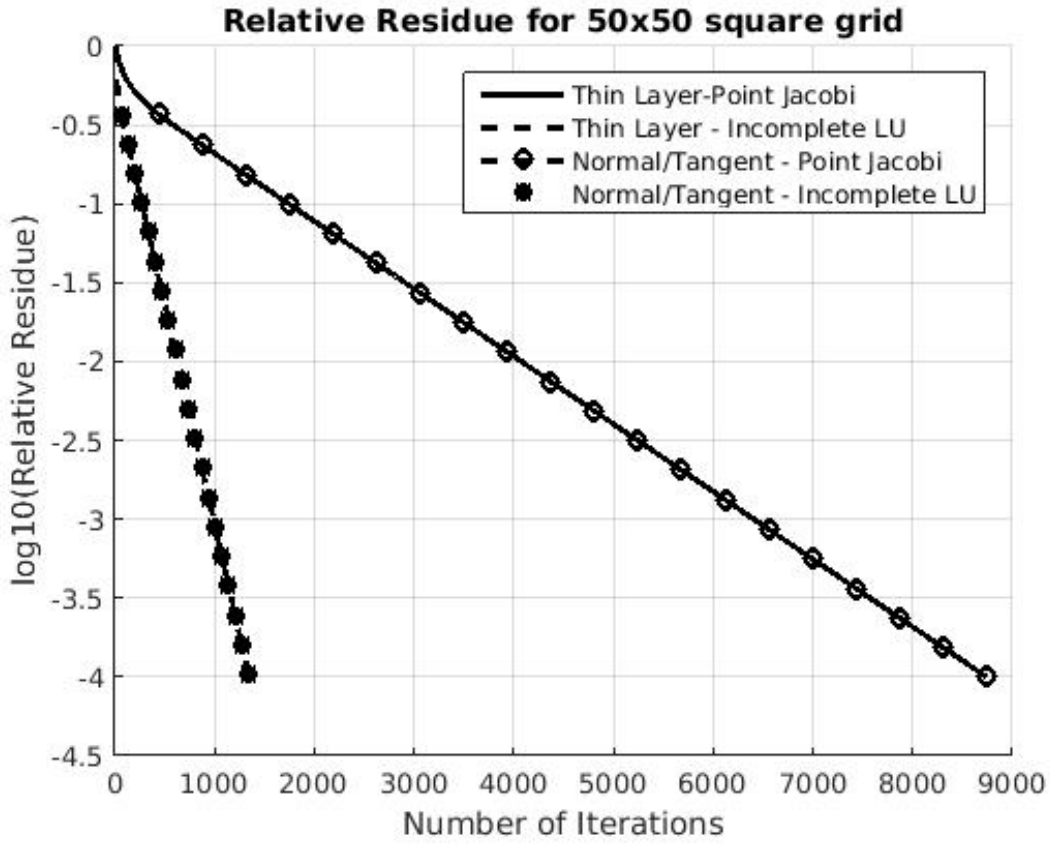


Figure 8: ϕ distribution using square grid 50x50

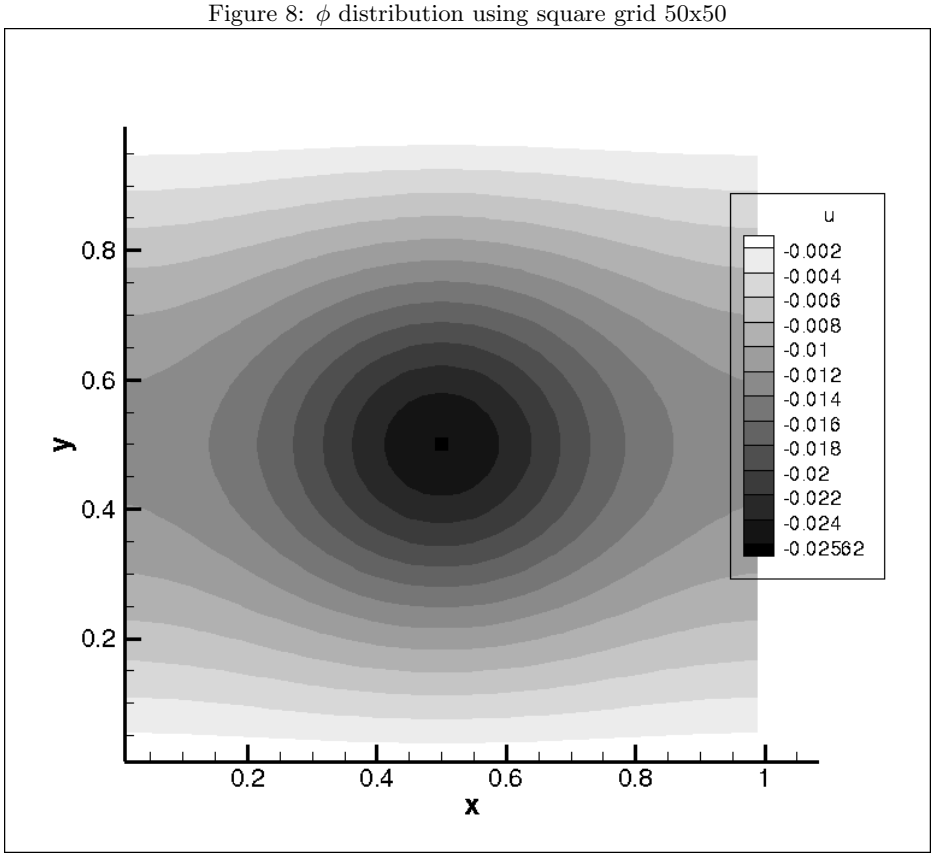
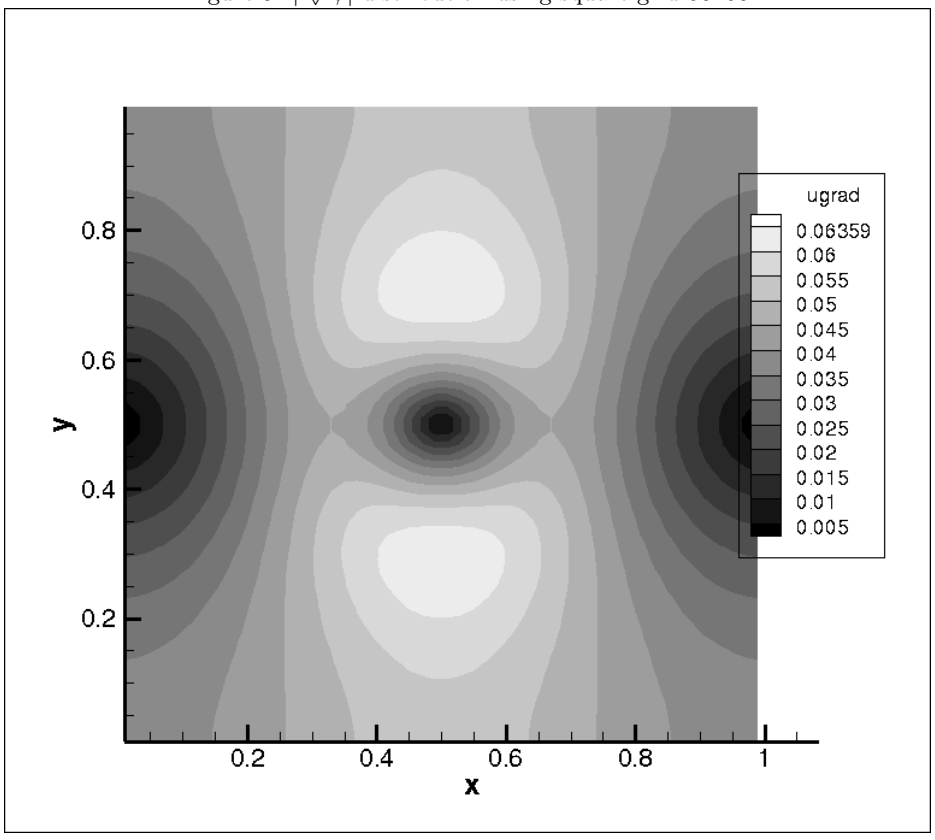


Figure 9: $|\bigtriangledown \phi|$ distribution using square grid 50x50



It is observed that thin layer and normal/tangent methods converges in same number of iterations using ILU and PJ methods. This is because the approximation expression used in normal/tangent and thin layer method reduces to same form after applying the finite volume method (Eq.5). Consider the flux on one of the faces of a cell. For a square grid cell, the $\overrightarrow{n}$ and $\overrightarrow{\tau}$ are equal. Using this expression, the Eq. 19 reduces to following:

$$\bigtriangledown\phi = (\bigtriangledown\phi \cdot \overrightarrow{n}_1)\overrightarrow{n}_1 + \overline{\bigtriangledown\phi} - (\overline{\bigtriangledown\phi} \cdot n_1)(\overrightarrow{n}_1 \cdot \overrightarrow{n}_1)\overrightarrow{n}_1 \tag{32}$$

$$\bigtriangledown\phi = (\bigtriangledown\phi \cdot \overrightarrow{n}_1)\overrightarrow{n}_1 + \overline{\bigtriangledown\phi} - (\overline{\bigtriangledown\phi} \cdot n_1)\overrightarrow{n}_1 \tag{33}$$

By applying the finite volume aproximation(Eq.6), the term $\overline{\bigtriangledown\phi} \cdot n$ on face 1 is given by.

$$\bigtriangledown\phi = (\bigtriangledown\phi \cdot \overrightarrow{n}_1)\overrightarrow{n}_1 + \overline{\bigtriangledown\phi} - (\overline{\bigtriangledown\phi} \cdot n_1)\overrightarrow{n}_1 \tag{34}$$

$$\bigtriangledown\phi \cdot \overrightarrow{n} = (\bigtriangledown\phi \cdot \overrightarrow{n}_1)(\overrightarrow{n}_1 \cdot \overrightarrow{n}_1) + \overline{\bigtriangledown\phi} \cdot \overrightarrow{n}_1 - (\overline{\bigtriangledown\phi} \cdot n_1)(\overrightarrow{n}_1 \cdot \overrightarrow{n}_1) \tag{35}$$

$$\bigtriangledown\phi \cdot \overrightarrow{n} = (\bigtriangledown\phi \cdot \overrightarrow{n}_1)(\overrightarrow{n}_1 \cdot \overrightarrow{n}_1) \tag{36}$$

The Eq. 36 will be same as the equation derived when the finite volume approximation (Eq.6) is applied on a face of a cell with thin layer method. This concludes that results for a mesh with mutually perpendicular sides will be same using thin layer and tangent/normal discretization methods. However, the CPU time is more using tangent/normal method compared to thin layer method due to more number of operations involved in tangent/normal method.

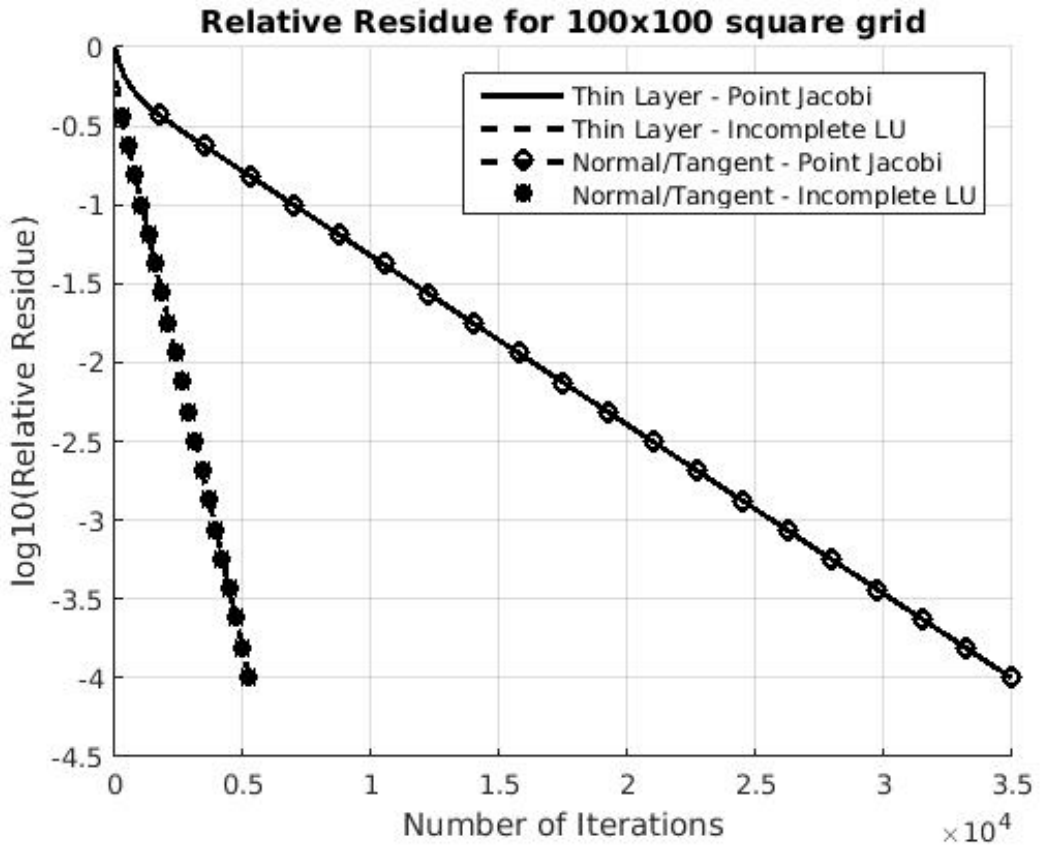
Fig. 7 and Fig. 8 shows the distribution of ϕ and the magnitude of gradient on a square grid. The minimum value of phi is -0.02562 and the contours are perpendicular to the right and left boundary due to no flux boundary condition. The value of ϕ is negative and this can be explained by assuming ϕ to be temperature in centigrade scale. The expression f(x,y) in the Eq.1 is positive and now the Poisson's equation will be same as that of heat conduction equation with a sink. This causes decrease of temperature and hence the value of ϕ is negative. The top and bottom boundaries are maintained at constant value and flux of ϕ will diffuse into the domain because the values of ϕ is lower than boundary value. Further it is obsorved that results are same for all the cases on a square grid for both 50x50 and 100x100 elements. Hence only one set of soultion is shown in this report.

4.3. Square grid 100x100

Table 2 shows the number of iterations and CPU time for square grid 100x100 and Fig. 10 shows the comparision of the convergence rate. Convergence rate and the CPU time required to run the simulation follow similar trend as that of square grid 50x50. The ILU integration method converges in less number of iterations than compared to PJ integration method for a particular discretization technique and the number of iterations are same when comparing thin layer and tangent/normal for a particular integration methods.

Number of Iterations and simulation time for square grid 100x100				
Method	N: PJ	t(s):PJ	N: ILU	t(s):ILU
Thin Layer	34967	33.80	5249	10.79
Normal Tangent	34967	456.83	5249	46.8283

Figure 10: Convergence plot for Square grid 100x100



However, for any combination of discretization and integration method, the number of iterations and simulation time is more for 100x100 grid than compared to that of square grid 50x50. In particular, it is observed that number of iterations for simulating on 100x100 grid is approximately four times as that of 50x50 grid. This is because the number of dependent variables solved using 100x100 grid is four times as that of 50x50

Fig. 11 and Fig. 12 are the distribution of ϕ and $|\bigtriangledown\phi|$ over the square domain respectively. The results shows that the values of ϕ and $|\bigtriangledown\phi|$ using 100X100 elements are similar to that of 50X50 grid and there is negligible difference in the absolute of maximum and minimum values. Hence, it can be concluded that the results are mesh independent.

Figure 11: ϕ distribution using square grid 100x100

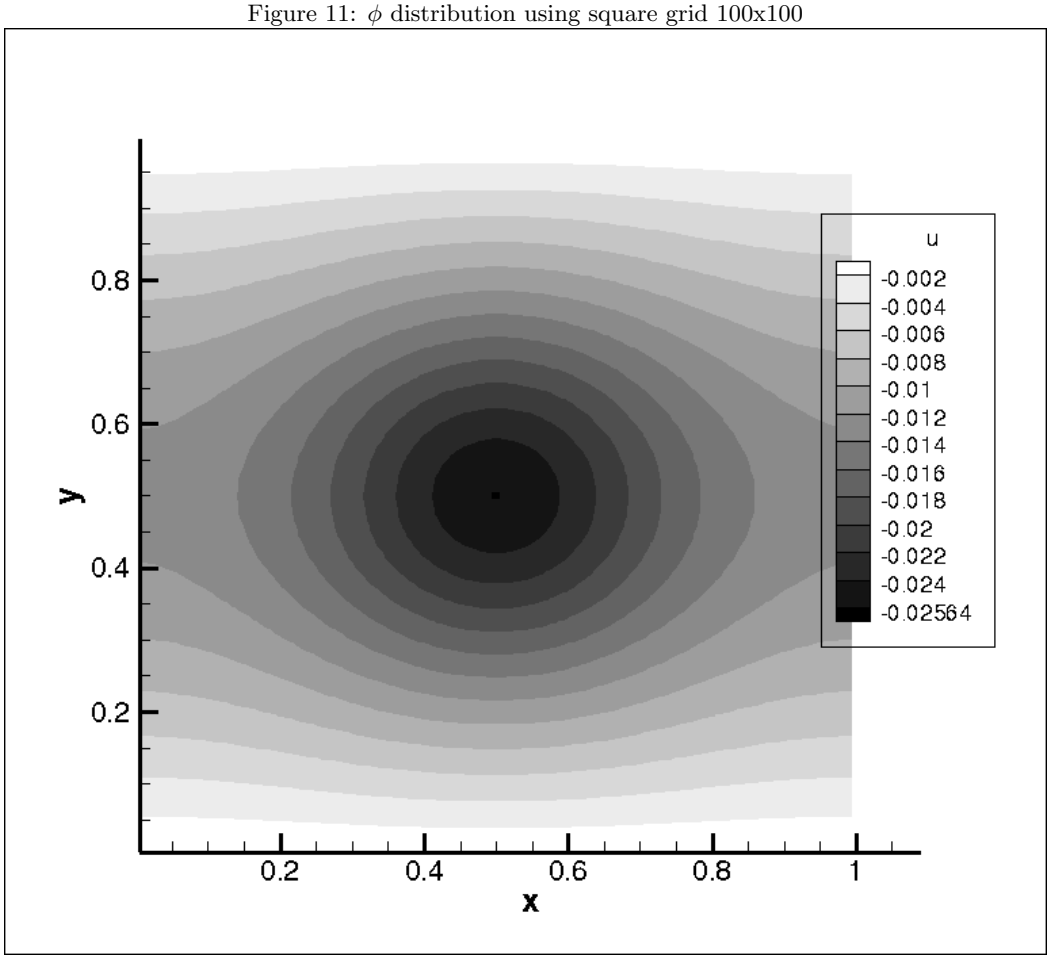
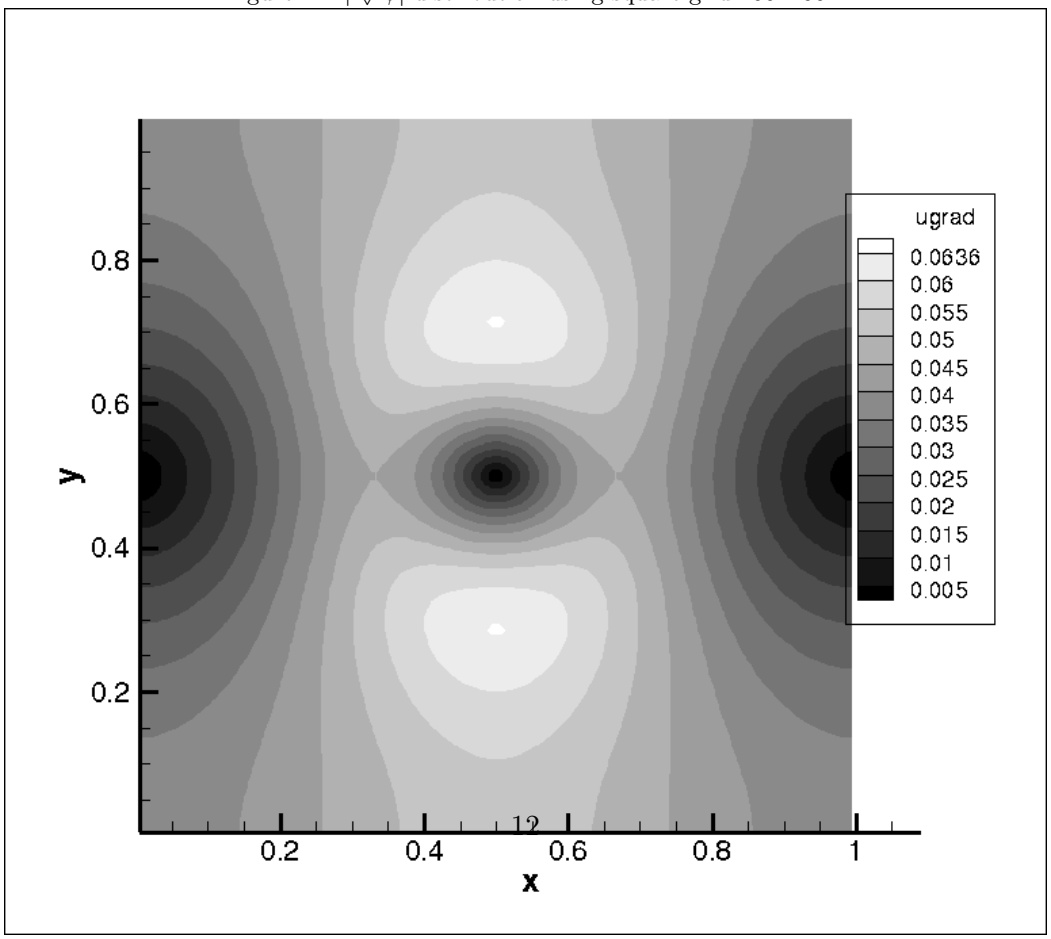


Figure 12: $|\bigtriangledown \phi|$ distribution using square grid 100x100



4.4. Curvilinear grid

Fig. 13 compares the convergence plot for all the cases and the Table 2 lists the maximum number of iteration and CPU time for different cases. The results shows that the convergence rate using thin layer method and tangent/normal method is almost similar for a particular integration method. However, it can be observed that convergence rate using normal/tangent method is slightly more than thin layer method for PJ integration and the scenario is reversed for ILU integration.

The trend in CPU time and maximum number of iterations follow similar trend as that of above cases. ILU integration converges in less number of iterations and time compared to PJ method for a particular discretization technique and tangent/normal method takes more time than compared to thin layer discretization technique for a particular integration method.

Table 2: Number of Iterations and simulation time for curvilinear grid

Method	N: PJ	t(s):PJ	N: ILU	t(s):ILU
Thin Layer	12416	7.17	853	1.17
Normal Tangent	11381	59.01	1331	7.91

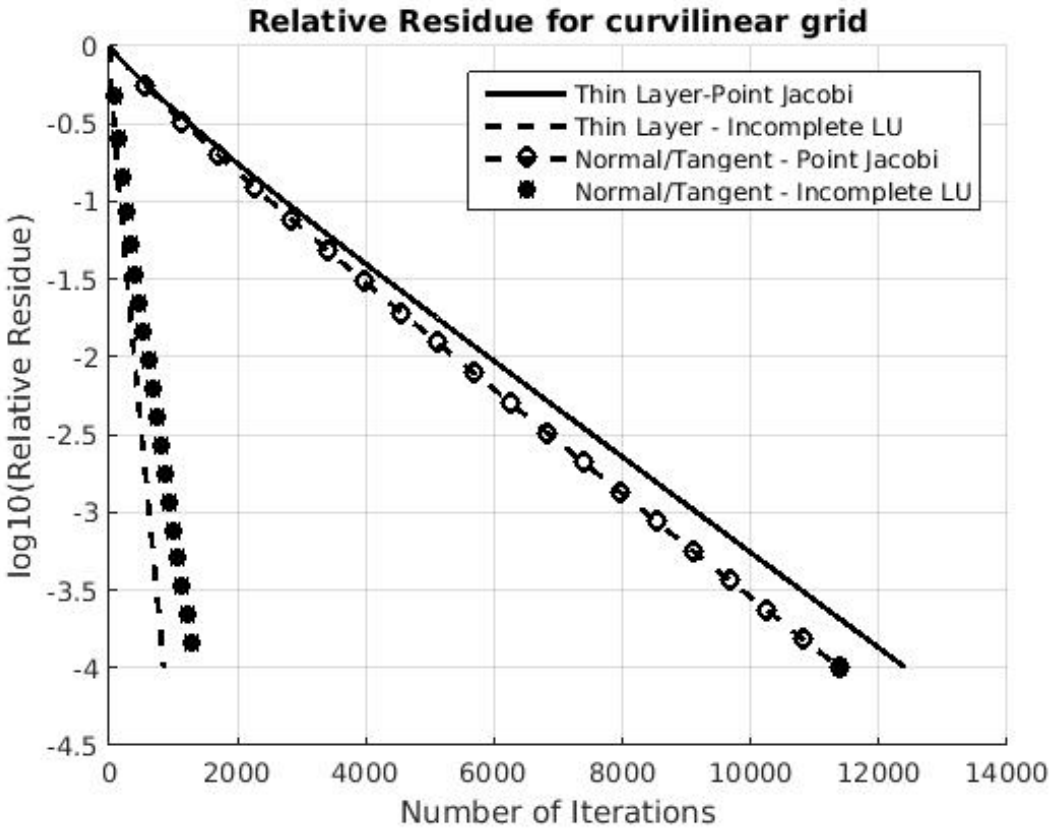


Figure 13: Convergence plot for curvilinear grid

Fig. 14, Fig. 15 shows the distribution of ϕ and $|\bigtriangledown\phi|$ using thin layer method and Fig. 16, Fig. 17 shows distribution ϕ and $|\bigtriangledown\phi|$ using tangent/normal method. The plots shows that the results are different using thin layer and tangent/normal methods. The minimum

value of ϕ and maximum value of its gradient is -7.8e-3, 0.095 respectively using thin layer and -8.3e-3, 0.109 using normal/tangent method. The maximum value of the gradient is observed on the center point of lower boundary because the cross section is lower at that cell and hence flux will be more. The minimum value of ϕ is obsorved at the center because the sink value is more at the center.

The distribution plots of ϕ and $| \bigtriangledown \phi|$ shows that the tangent/normal method resolves the gradient more accurately than the thin layer. This is because thin layer considers only normal flux when discretizing the Poisson's equation whereas tangent/normal considers the flux along both tangent and normal direction. However the solution is similar using PJ or ILU for a particular discretization method and hence only two sets of solutions are presented.

Figure 14: ϕ distribution on a curvilinear grid using thin layer discretization method

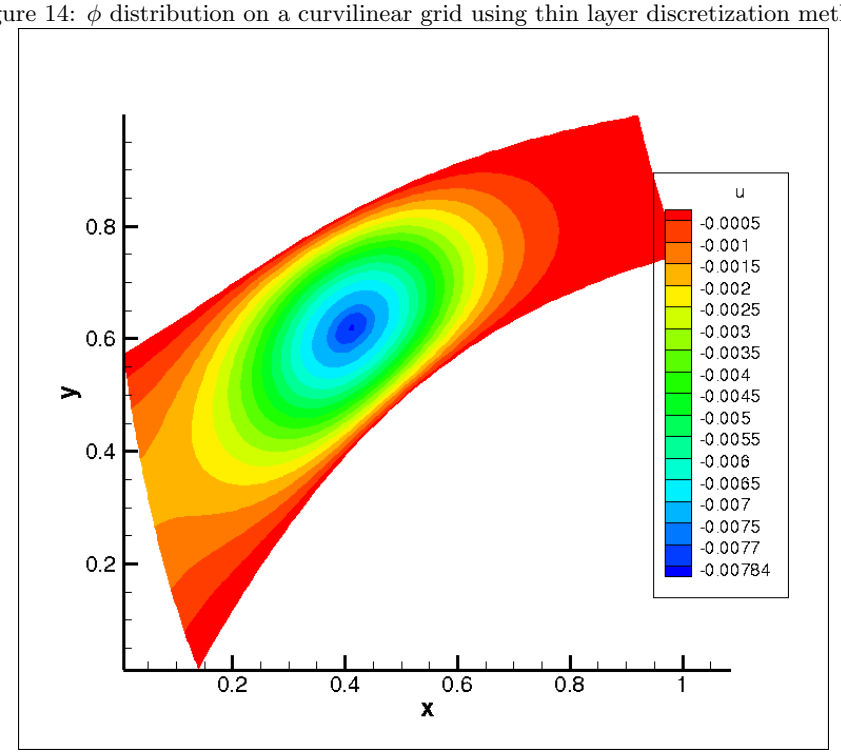


Figure 15: $|\bigtriangledown \phi|$ distribution on a curvilinar grid using thin layer discretization method

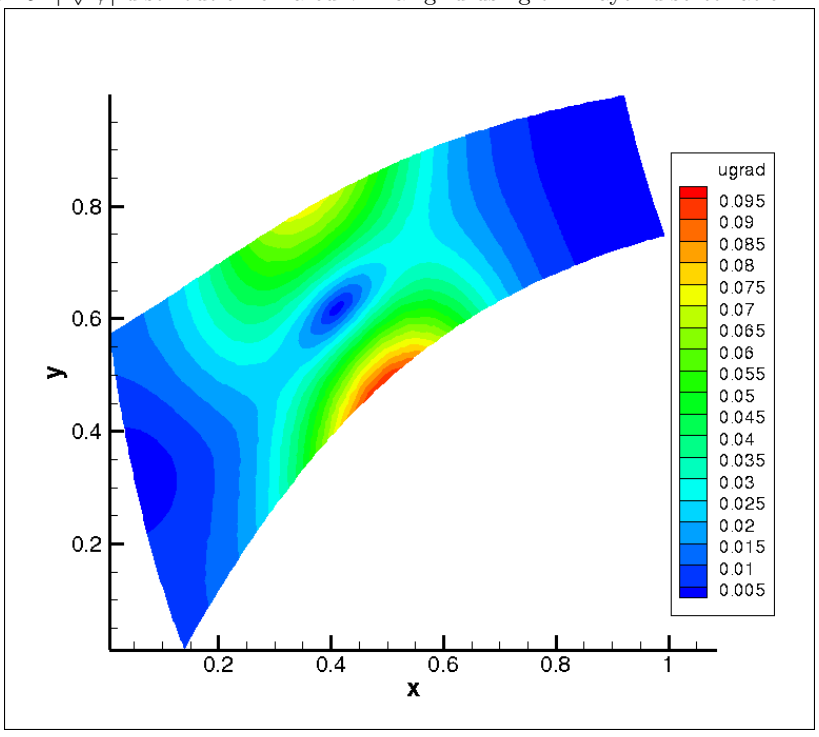


Figure 16: ϕ distribution on a curvilinear grid using tangent/normal method

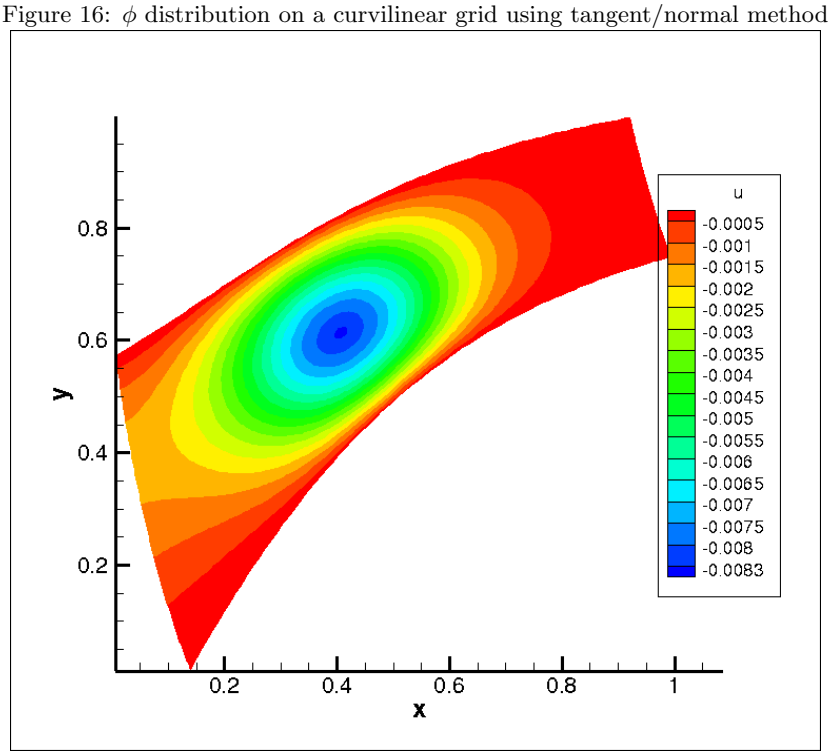
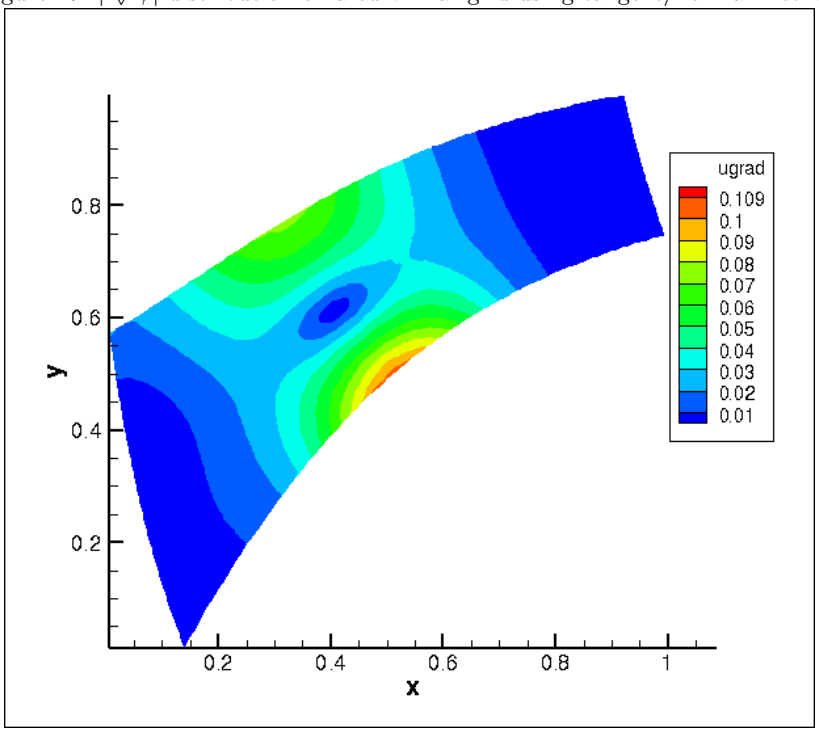


Figure 17: $|\bigtriangledown \phi|$ distribution on a curvilinar grid using tangent/normal method



5. Conclusion

In this work, the Poisson's equation is successfully solved for all combinations of discretization techniques and integration methods on three types of grids. In general, it is observed that the ILU method converges in less number of iterations and takes less time than PJ method for any combination of discretization technique and mesh. It is also observed that the results are same using PJ and ILU integration method for any combination of discretization techniques and meshs.

For a particular square grid, the number of iterations to converge is exactly same with thin layer and tangent/normal discretization as the finite volume approximation reduces to the same expression for rectangular mesh elements. However, the simulation time is more using tangent/normal method because of more number of operations in using the code developed for a generalized domain. In general, for a square grid, solution of ϕ is same for all possible cases. This also concludes that the solution achieved on the square domain is mesh independent.

In case of curvilinear grid, the number of iterations and CPU time follows similar trend as that of other cases but the solution is not same with thin layer and tangent/normal method. The results shows that the tangent/normal method resolves the gradients much better than thin layer method because only normal gradients on a cell face are considered when using thin layer method.

6. Appendix

The C++ program to solve the Poisson's equation is arranged into subroutine with an input file. The input file is used by the user for providing the mesh file name, boundary conditions, discretization method, integration method, maximum number of iterations and tolerance to the program. The subroutine names used in the program is listed below. The program also returns output files of the phi value, gradient values in x and y direction, CPU time and convergence data.

1. make
2. functions.h
3. input.dat
4. main.cpp
5. readinput.cpp
6. readinggrid.cpp
7. trianglearea.cpp
8. initialconditions.cpp
9. boundaryconditions.cpp
10. source.cpp
11. thinlayer.cpp
12. tangentnormal.cpp
13. results.cpp

References