Adaptive Grid Refinement Methodology for Unsteady Diffusion Equation on Unstructured Triangular Mesh

Dissertation

Submitted in partial fulfillment of the requirements
for the degree of

Master of Science

By

Parag Vijayraj Pathak

Department of Mechanical Engineering

Abstract

While solving partial differential equations, numerical methods have to be used since analytical solution is typically not available. In many cases, solution given by numerical methods does not have a desired level of accuracy. In such cases, to maintain a certain level of accuracy, the entire domain has to be refined. This significantly increases computational time and hence cost. Thus it becomes necessary to selectively refine the regions in the domain to keep a desired level of accuracy while leaving the rest of the domain unaltered so that a level of computational optimality is obtained.

The work uses a Finite Volume Methodology with an explicit scheme to solve an unsteady diffusion equation which is parabolic differential equation in 2-D on an unstructured triangular conformal mesh. A Taylor series based error analysis is performed to find the parameters on which errors depend on. A curvature based grid refinement indicator is developed to select the region in space domain to be refined. A methodology to refine an unstructured triangular conformal mesh is developed such that the conformal nature of the mesh is retained. The overall methodology working together and its effectiveness on some general test problems has been done.

The method is able to keep a certain level of accuracy while retaining good computational optimality. It is able to refine and coarsen the mesh without much loss of mesh quality.

Table of Contents

Table of figures

Abbreviations

$\bar{T}$	The Approximate or lower order solution of conserved physical quantity being modeled
Δt	Time step
A_b	Length of boundary face f
A_f	Boundary Control Volume or length of face f.
C_ξ, C_η	Co-efficient for fluxes in ξ and η directions
D	The diffusion constant
dV	Deferential area element in 2D
dX	Differential element of Space domain
e	Error
e_b	Bulk error
e_d	Deviation error
e_p	Error at point P
E_δ	Grid refinement indicator
F	Intersection point of line joining nodes and line joining centroids in discretized space.
FDM	Finite Difference equation
FEM	Finite Element equation
FVM	Finite Volume Equation
G	Amplification factor
K	Diffusivity of conserved quantity T
N	For n^{th} iteration
S	Source term in diffusion equation
S_b	Bulk error source term
S_d	Deviation error source term
S_e	Source term in error diffusion equation
T	The exact or higher order solution of conserved physical quantity being modeled
V_P	Area of cell P.
A	The diffusion coefficient
H	Direction along the line joining the nodes of the control volume
Ξ	Direction along the line joining the cell centers of the control volume
ρc	Density of conserved quantity T
Ω	Bounded Space domain

Mesh Generation

1. Grid generation

A good grid generation methodology must generate well shaped grids on domain of any level of complexity. The skewness of the grids must be low. It should also have an adaptive nature to the boundaries. It should be easily programmable.

When every point in a group of points is a node of any triangle of group of non-intersecting triangles then such a group of triangles is called the triangulation of the given set of points. There can be many ways of triangulating a given set of points.

If circum-circle of any triangle does not enclose any other node in the domain, then such a triangulation is called the Delaunay triangulation of the given set of points. This property ensures optimum shapes of the elements generated inside the domain. This is also called the "in-circle" property or criteria to identify a triangulation as Delaunay.

Rupert's Algorithm (Wikipedia) is one such grid generation methodology available. Bowyer 1981 and Watson 1981 have developed a procedure to generate a Delaunay triangulation from n points called Bowyer algorithm.

Bowyer algorithm

Our aim is to generate the Delaunay triangulation of the given set of points P_1 to P_n such that the given triangulation maps a convex domain. Instead of triangulating the points all at once we employ point by point methodology.

Let us say we have Delaunay triangulated points from P_1 to P_m. $4<m<n+1$ A new point P_{m+1} is considered. We then track the changes that need to be done to add P_{m+1} to the existing triangulation such that the new triangulation which includes the P_{m+1} also is Delaunay. As m goes from 5 to n we have Delaunay triangulated the given set of points.

The changes are done according to the bowyer algorithm.

1. If the newly added point lies inside the circum-circle of any triangle, then this triangle will be deleted.
2. A list of all faces common between deleted triangles and undeleted neighbors is obtained.
3. New triangles are created consisting of the newly added point and the points of the edges obtained in step 2.

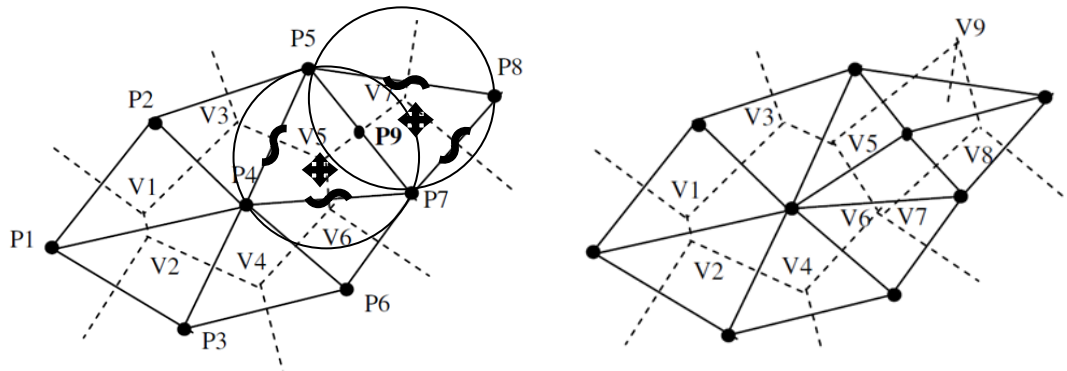


Figure A1.1 A point being inserted in an existing Delaunay triangulation.

In the Figure A1.1, we want generate the Delaunay triangulation of set of points P_1 to P_9 .We have Delaunay triangulated points from P_1 to P_8 .To do this we employ a point by point methodology. A point P_9 needs to be added some where inside an existing Delaunay triangulation such that the new triangulation also has Delaunay in-circle property. This is done by the Bayer algorithm given below. Point P_9 is added such that it lies is the circum-circle of triangles V_5 and V_7. In the above example triangles marked by '+' are deleted while edges marked by '~' satisfy step 2. New triangles are created consisting of the edges marked by '~' and P_9.

Note that if a triangle is to be **deleted** or created, only the data structure of the triangle is deleted or altered. The data structure of the points forming the triangle, remain intact and unaffected by deletion or alteration of the triangle.

Weatherill 1988 and 1992 developed a mesh generation methodology based on the above discussed concept of sequential addition of points in the domain. The method starts with only the boundary mesh and gradually progresses inside the domain by generating new nodes and adding them using the algorithm described above. The method can be easily extended to three dimensional domains. This method appears attractive as it minimizes the inputs from the user and ensures good mesh quality. Hence this method was used in the code developed for triangulation. The steps involved are listed below.

1. Initiating the Delaunay triangulation

We are given a set of points that define a boundary of a 2-D domain that have to be triangulated using the Delaunay procedure. The Bowyer algorithm needs some initial Delaunay triangles to start triangulation such that they cover the entire domain. Hence four temporary points are added to this set such that, a convex quadrilateral is formed from them and the set of boundary points lies inside this quadrilateral. Two Delaunay triangles are created from them by joining appropriate points. The Bowyer algorithm is initiated from these two triangles. The final mesh is unaffected by the initial points.

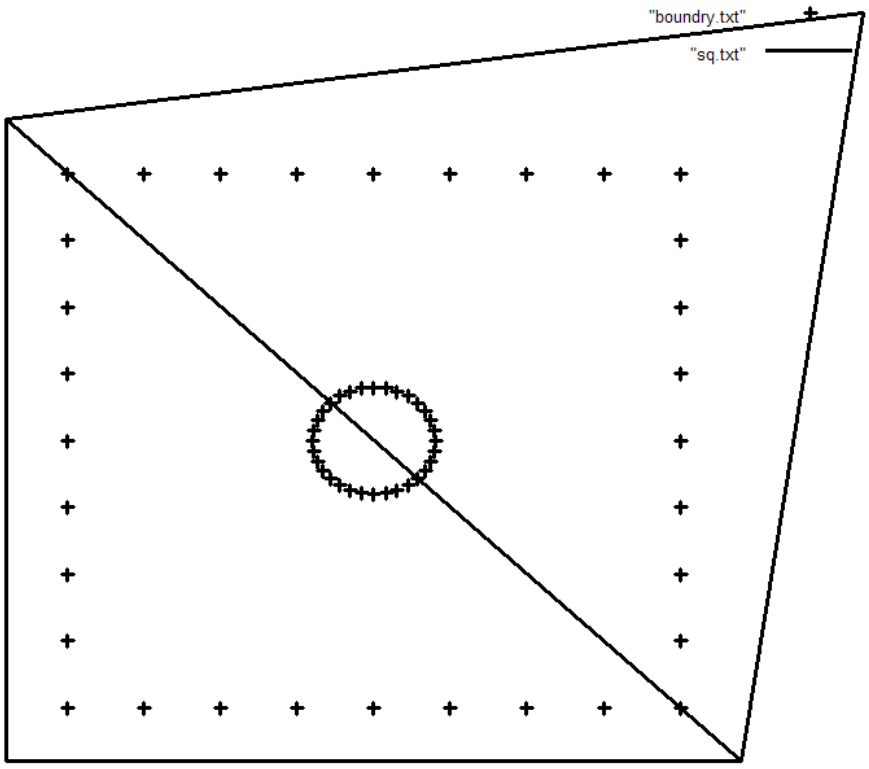


Figure A1.2 Convex quadrilateral around points to be triangulated

2. Generating the boundary mesh

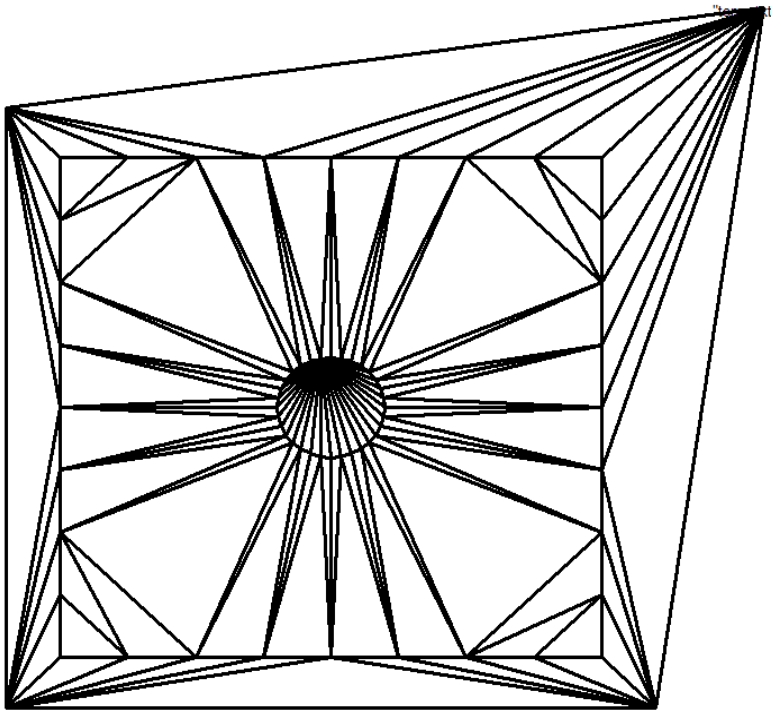


Figure A1.3 Boundary mesh of the triangulated points

The set of boundary points is fed one by one into the initiated Delaunay triangulation. The mesh is modified using the Bowyer algorithm described above. After this step is complete, triangulated

mesh consisting of set of boundary points and the four additional points of the initial quadrilateral is obtained.

3. **Cleaning process to delete triangles outside the actual domain**

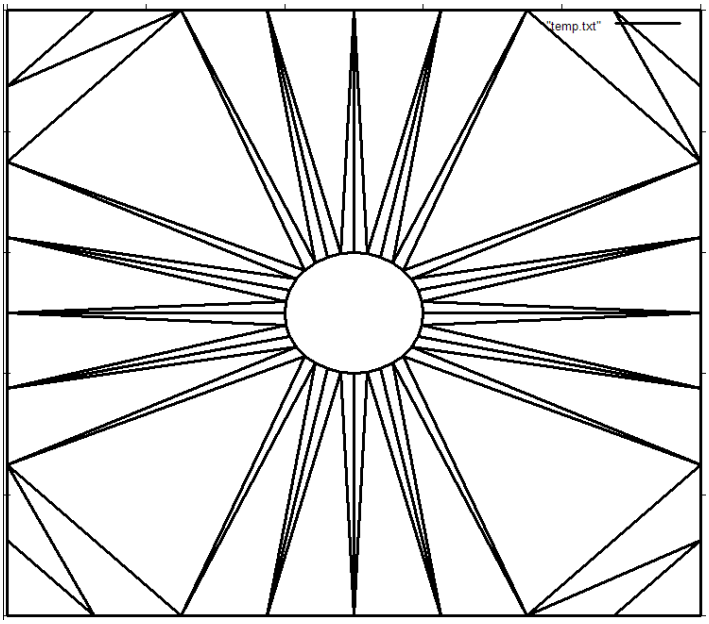


Figure A1.4 Cleaned up boundary mesh of the triangulated points

All those triangles which are outside the domain are deleted. After this step is complete, triangulated mesh consisting of only the boundary points is obtained. This serves as the initial mesh for automatic point insertion step. This triangulation has the Delaunay property because of the Bowyer algorithm.

4. **Automatic point insertion inside the domain**

This step is the most crucial step as it decides the location of the nodes inside the domain, and triangulates it.

Active triangles are those that generate a new set of prospective points at their centroids for next iteration. **Inactive triangles** are those that do not generate new points. For the first iteration all triangles of the initial mesh are initialized to be active.

Point Distribution Function (PDF) is defined for every node, which represents the local mesh scale. For every boundary node it is calculated as average of the lengths of the two adjacent edges sharing this node. PDF for the prospective centroidal point is obtained by taking average of the PDF at the nodes of the triangle. Following are the steps involved in the triangulation.

With this we begin the first iteration of the automatic point insertion algorithm.

1. New prospective points are generated at the centroids of the active triangles for the current iteration.
2. PDF for the new prospective points is obtained for the current iteration.
3. A prospective point must satisfy the following conditions.

- Distances of this centroidal point from the any node of the triangle that created it must be greater than 0.7 times the PDF at the node.

- Consider any other point (that include established points of the previous iteration as well as newly considered non-rejected prospective points) that has a PDF smaller than the prospective point being examined. The distance between these two points must be greater than 0.7 times the PDF of the prospective point

4. A set of prospective points passing step three is obtained. This set is inserted one by one into the existing Delaunay triangulation using the Bowyer algorithm for boundary triangulation.
5. Consider the average PDF of any two nodes of a triangle. If the average PDF is less than 2/3rd the distance between them, then the triangle is made inactive. This step is done for all triangles.

The algorithm converges when either of the following situations arises.

- All the triangles remain inactive after iteration.
- All the prospective points get rejected by step 3 or 4.

Else the next iteration begins from step 1.Figure below shows meshes after number of iterations

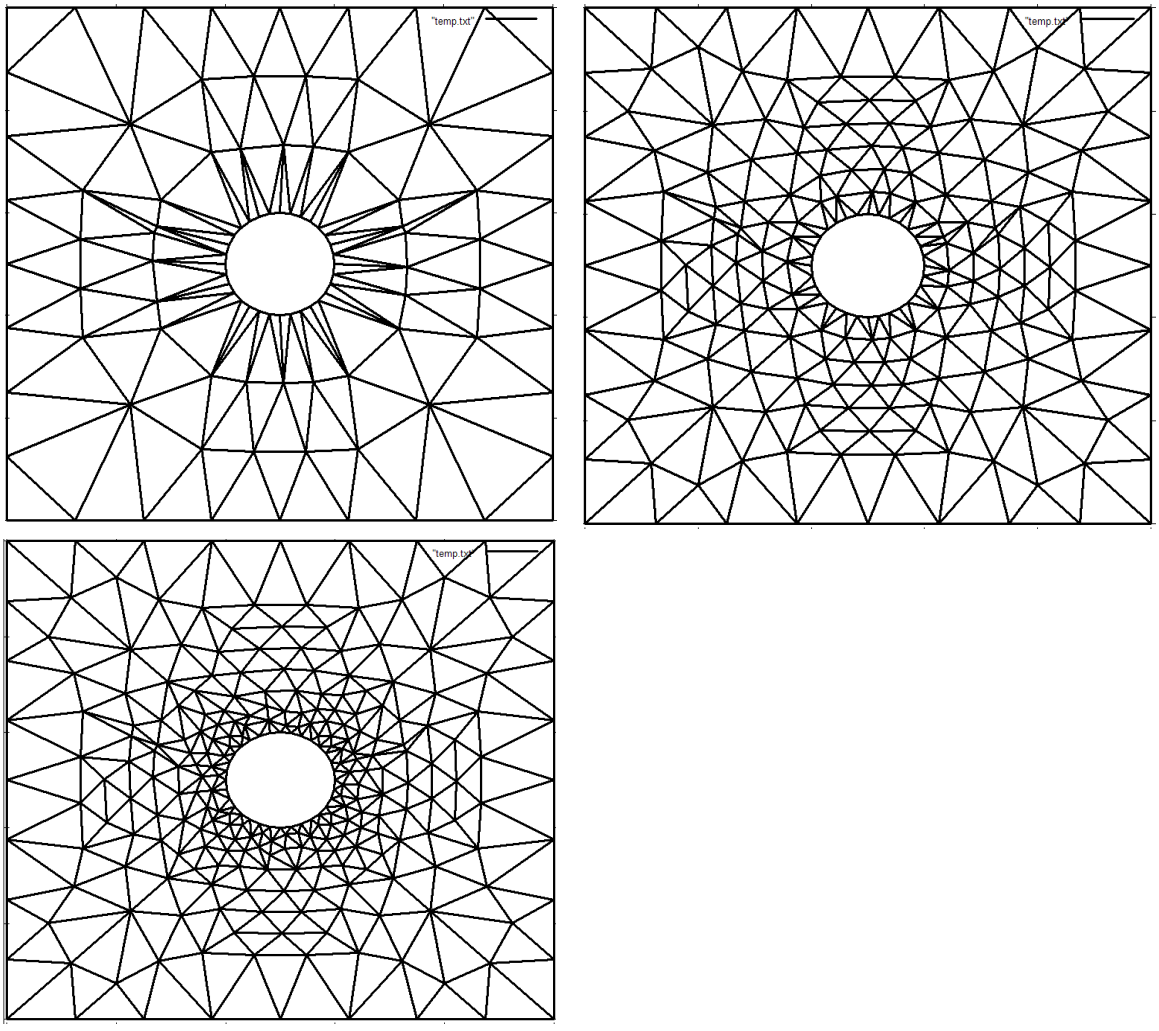


Figure A1.5 A Progressive Iterations of the Automatic point insertion method

5. Smoothing the mesh to improve mesh shape

This above process is followed by a smoothing process (Laplacian smoothing), where every point in the domain is slowly and

iteratively moved to the centroid of its surrounding points using the following method. This improves the mesh quality by stretching the triangles to get them close to an equilateral shape.

$$x_i = x_i + \sum_{NB} (x_{NB} - x_i) \frac{\alpha}{N}$$

Where x_i is the coordinate of the point i α is an under-relaxation parameter, x_{NB} is a surrounding point coordinate, and *N* is the number of surrounding points. A similar equation appears for 'y' coordinate. These iterations are carried out till the difference in the distance moved by any point during the two consecutive smoothing iterations is less than 0.001 times the PDF at that point.

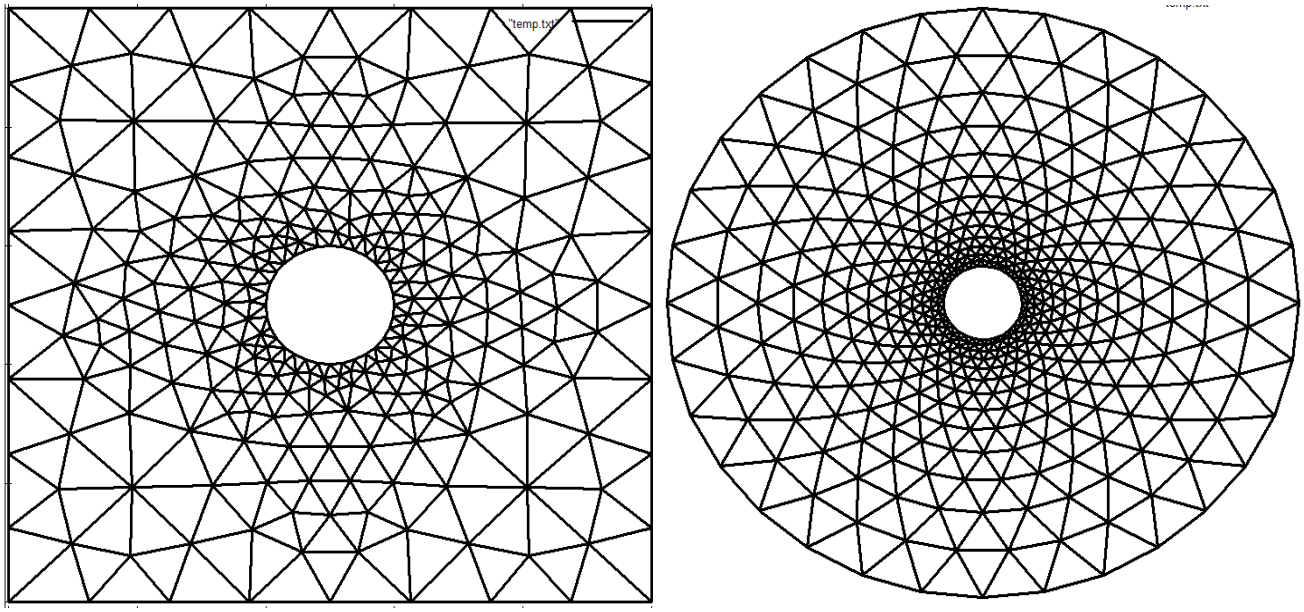


Figure A1.6 Final smoothened meshes

2. Error analysis

Let T be the true solution.

The following approximations were done

1) $T_t \approx \frac{T_P^{n+1} - T_P^n}{\Delta t}$ Linear time step approximation

2) $\nabla^2 T \approx \frac{1}{\Delta V} \int \nabla^2 T dV$ Linear spatial Laplacian approximation.

3) $\oint \vec{\nabla} T . d\vec{A} \approx \sum \vec{\nabla} T_f . \vec{A}_f$ Face flux approximation.

4) $T_\xi \approx \frac{T_F - T_P}{\Delta \xi}$ and $T_\eta \approx \frac{T_2 - T_1}{\Delta \eta}$ Linear face gradient approximation

1) Face flux approximation

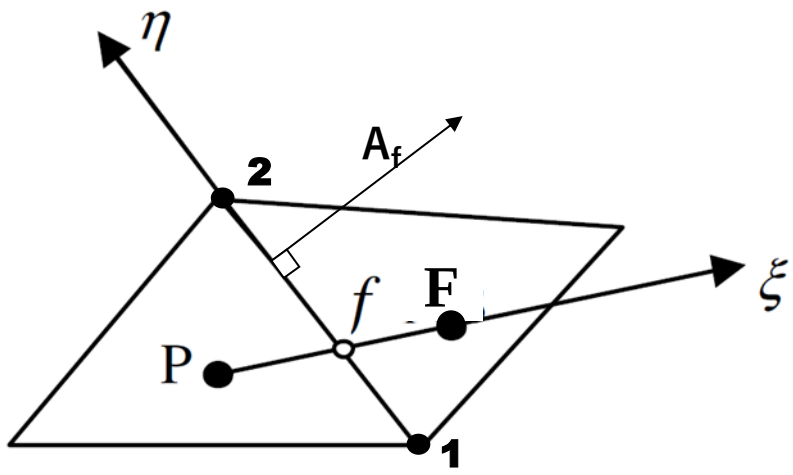


Figure A2.1 Control volume P and F

Our aim is to write

$$\int_1^2 \vec{\nabla} T . d\vec{A}_f = C_\xi \left(\frac{\delta T}{\delta \xi}\right)_f + C_\eta \left(\frac{\delta T}{\delta \eta}\right)_f + E_3 \qquad \text{(A2.1)}$$

Using vector algebra, we can show that

$$\vec{\nabla} T = \frac{T_\xi - w T_\eta}{1 - w^2} \hat{\xi} + \frac{T_\eta - w T_\xi}{1 - w^2} \hat{\eta} \qquad \text{(A2.2)}$$

Where $w = \hat{\xi} . \hat{\eta}$. $T_\xi and\ T_\eta$ are gradients in ξ and η directions

Using gradient equation A2.2 in A2.1

$$\int_{\eta_1}^{\eta_2} \vec{\nabla} T . d\vec{A}_f = \frac{\int_{\eta_1}^{\eta_2} T_\xi d\eta - w \int_{\eta_1}^{\eta_2} T_\eta d\eta}{1 - w^2} \hat{\xi} . \hat{A}_f \Big|_{\xi=0} \qquad \text{(A2.3)}$$

Substituting from

$$Let\ C_\xi = \frac{\hat{\xi} . \vec{A}_f}{1 - w^2}\ and\ C_\eta = -\frac{w \hat{\xi} . \vec{A}_f}{1 - w^2}\ \text{and}\ since\ \hat{\eta} . \hat{A}_f = 0$$

$$\int_{\eta_1}^{\eta_2} \vec{\nabla} T . d\vec{A}_f = C_\xi \int_{\eta_1}^{\eta_2} T_\xi d\eta \Big|_{\xi=0} + C_\eta \int_{\eta_1}^{\eta_2} T_\eta d\eta \Big|_{\xi=0} \quad \text{(A2.4)}$$

Expanding Taylor series at Point 'f' in ξ and η directions

$$T(\xi,\eta) = T_f + \left(\frac{\delta T}{\delta \xi}\right)_f \xi + \left(\frac{\delta T}{\delta \eta}\right)_f \eta + \left(\frac{\delta^2 T}{\delta \xi^2}\right)_f \frac{\xi^2}{2} + \left(\frac{\delta^2 T}{\delta \eta^2}\right)_f \frac{\eta^2}{2} + \left(\frac{\delta^2 T}{\delta \xi \delta \eta}\right)_f \xi\eta + \cdots \quad \text{(A2.5)}$$

Taking gradients in ξ direction of eqn A2.5

$$T_\xi = \left(\frac{\delta T}{\delta \xi}\right)_f + T_{\xi\xi}\xi + T_{\xi\eta}\eta + \cdots \quad \text{(A2.6)}$$

$$\int_{\eta_1}^{\eta_2} T_\xi d\eta \Big|_{\xi=0} = \left(\frac{\delta T}{\delta \xi}\right)_f A_f + T_{\xi\eta} \frac{(\eta_1+\eta_2)}{2} A_f \quad \text{(A2.7)}$$

From figure A2.1 , $A_f = \eta_2 - \eta_1$.Also substituting $\frac{(\xi_F+\xi_P)}{2} = \frac{(d_F-d_P)}{2}$ *and* $\frac{(\eta_1+\eta_2)}{2} = \frac{(d_2-d_1)}{2}$

Similarly equation as A2.6 and A2.7 is for η eqn A2.8 and eqn A2.9.

$$T_\eta = \left(\frac{\delta T}{\delta \eta}\right)_f + T_{\eta\eta}\eta + T_{\xi\eta}\xi + \cdots \quad \text{(A2.8)}$$

$$\int_{\eta_1}^{\eta_2} T_\eta d\eta \Big|_{\xi=0} = \left(\frac{\delta T}{\delta \eta}\right)_f A_f + T_{\eta\eta} \frac{(\eta_1+\eta_2)}{2} A_f \quad \text{(A2.9)}$$

Substitute eqn A2.9 and A2.7 in eqn A2.4 to get

$$\int_1^2 \vec{\nabla} T . d\vec{A}_f = C_\xi \left(\frac{\delta T}{\delta \xi}\right)_f + C_\eta \left(\frac{\delta T}{\delta \eta}\right)_f + E_3 \quad \text{(A2.10)}$$

$$E_3 = \frac{(d_2-d_1)}{2}\left(C_\xi T_{\xi\eta} + C_\eta T_{\eta\eta}\right) \quad \text{(A2.11)}$$

2) **Linear face gradient approximation**

We want to find out the expansions of $\left(\frac{\delta T}{\delta \eta}\right)_f$ *and* $\left(\frac{\delta T}{\delta \xi}\right)_f$.Expanding Taylor series at points P and F in Figure A2.1

$$T_P = T_f + \left(\frac{\delta T}{\delta \xi}\right)_f \xi_P + T_{\xi\xi}\frac{\xi_P^2}{2} + \cdots \quad \text{(A2.12)}$$

$$T_F = T_f + \left(\frac{\delta T}{\delta \xi}\right)_f \xi_F + T_{\xi\xi}\frac{\xi_F^2}{2} + \cdots \quad \text{(A2.13)}$$

Subtracting above equations A2.12 and A2.13

$$\left(\frac{\delta T}{\delta \xi}\right)_f = \frac{T_F - T_P}{\Delta \xi} + T_{\xi\xi}\frac{(\xi_F + \xi_P)}{2} + \cdots \quad \text{(A2.14)}$$

Similarly for η we get

$$\left(\frac{\delta T}{\delta \eta}\right)_f = \frac{T_2 - T_1}{\Delta \eta} + T_{\eta\eta}\frac{(\eta_1 + \eta_2)}{2} + \cdots \quad \text{(A2.15)}$$

Substitute eqn s A2.14 and A2.15 in expression below to get

$$C_\xi\left(\frac{\delta T}{\delta \xi}\right)_f + C_\eta\left(\frac{\delta T}{\delta \eta}\right)_f = C_\xi\frac{T_F - T_P}{\Delta \xi} + C_\eta\frac{T_2 - T_1}{\Delta \eta} + E_4 \quad \text{(A2.16)}$$

$$E_4 = C_\xi T_{\xi\xi}\frac{(d_F - d_P)}{2} + C_\eta T_{\eta\eta}\frac{(d_2 - d_1)}{2} \quad \text{(A2.17)}$$

Total error

By adding E_3 and E_4 we get total error contribution due to approximations by considering Eqns A2.10, A2.11, A2.16 and A2.17

$$\int_1^2 \vec{\nabla}T.d\vec{A}_f = C_\xi\frac{T_F - T_P}{\Delta \xi} + C_\eta\frac{T_2 - T_1}{\Delta \eta} + E_f \quad \text{(A2.18)}$$

$$E_f = E_3 + E_4 = \frac{(d_F - d_P)}{2}C_\xi T_{\xi\xi} + \frac{(d_2 - d_1)}{2}\left(C_\xi T_{\xi\eta} + 2C_\eta T_{\eta\eta}\right) = O(\delta^2) \quad \text{(A2.19)}$$

1) Linear time step approximation

The Taylor equation for time series is

$$T_P^{n+1} = T_P^n + T_t\Delta t + \frac{T_t}{2}\Delta t^2 + \cdots \quad \text{(A2.20)}$$

Rearranging the terms in eqn A2.1 and neglecting higher order terms

$$T_t = \frac{(T_P^{n+1} - T_P^n)}{\Delta t} - \frac{T_{tt}}{2}\Delta t \quad \text{(A2.21)}$$

2) Linear spatial Laplacian approximation.

Our Aim find E_δ such that

$$\frac{1}{\Delta V_P}\int(\nabla^2 T)dV = (\nabla^2 T)_P - E_\delta \quad \text{(A2.22)}$$

Consider the Taylor series expansion of a general function T at cell center P in x and y

$$T(x,y) = T_P + T_x x + T_y y + T_{xx}\frac{x^2}{2} + T_{yy}\frac{y^2}{2} + T_{xy}xy + \cdots \quad \text{(A2.23)}$$

All derivatives are at Centroid P

$$\int_{V_P} T(x,y)dV = T_P \int_{V_P} dV + T_x \int_{V_P} xdV + T_y \int_{V_P} ydV + \\ + \frac{T_{xx}}{2} \int_{V_P} x^2\, dV + \frac{T_{yy}}{2} \int_{V_P} y^2 dV + T_{xy} \int_{V_P} xydV \tag{A2.24}$$

$$By\ defenition\ , \int_{V_P} xdV\ , \int_{V_P} ydV = 0\ since\ P\ is\ centroid$$

$$Put\ \int_{V_P} dV = \Delta V_P, I_{xx} = \int_{V_P} x^2\, dV, I_{yy} = \frac{T_{yy}}{2} \int_{V_P} y^2 dV\ , I_{xy} = T_{xy} \int_{V_P} xydV$$

Making above substitutions in eqn A2.24

$$\int_{V_P} T(x,y)dV = T_P \Delta V_P + T_{xx} \frac{I_{xx}}{2} + T_{yy} \frac{I_{yy}}{2} + T_{xy} I_{xy} \tag{A2.25}$$

Instead of T putting $\nabla^2 T$,Dividing by ΔV_P and rearranging eqn A2.25

$$\frac{1}{\Delta V_P} \int (\nabla^2 T)dV = (\nabla^2 T)_P - E_\delta \tag{A2.26}$$

$$Where\ E_\delta = -\frac{1}{\Delta V_P} \left((\nabla^2 T)_{xx} \frac{I_{xx}}{2} + (\nabla^2 T)_{yy} \frac{I_{yy}}{2} + (\nabla^2 T)_{xy} I_{xy} \right) \tag{A2.27}$$

3. Stability Analysis

While simulating the unsteady diffusion equation or a parabolic equation using the explicit discretization scheme and finite volume methodology the solution can suffer from a stability crisis if ration the time step and the volume are not within a certain limit. The aim is to find this stability limit for the triangular equilateral mesh. A Von Neumon stability analysis is performed for this equation in an explicit scheme on an equilateral triangular mesh. A limit is established for the ratio of time step and cell area for the mesh.

The diffusion equation that is being simulated is

$$T_t = \alpha \nabla^2 T + S$$

The diffusivity is α and the temperature is T with S as the source term. The relation is simulated in two dimensions. Using the explicit scheme on the discretised domain it can be shown that. An explicit scheme is used to discretize the time domain on a triangular equilateral mesh. A finite volume discretization is used to discretize the space domain. The diffusion

$$\frac{\delta \iint T dV}{\delta t} = \alpha \oint \vec{\nabla} T . d\vec{A}_f + S_{avg}$$

After linear approximations across the cell (refer Figure 0.1), the equation is as follows

$$T_P^{n+1} = T_P^n + \frac{\alpha \Delta t}{\Delta V_p} \sqrt{3} \sum_F (T_F^n - T_P^n) + S_p$$

This series is conditionally stable. A condition to ensure the stability of this series and the reduction of errors needs to be met.

Let e be the corresponding error field of T. Since the equation linear, The corresponding error diffusion is.

$$Let\ T = T' + e$$

Where T is actual temperature, T' is calculated and e is the error.

$$T'_t + e_t = \alpha \nabla^2 T' + S + \alpha \nabla^2 e$$

Now as calculated value also has to obey the equation $T'_t = \alpha \nabla^2 T' + S$

$$e_t = \alpha \nabla^2 e$$

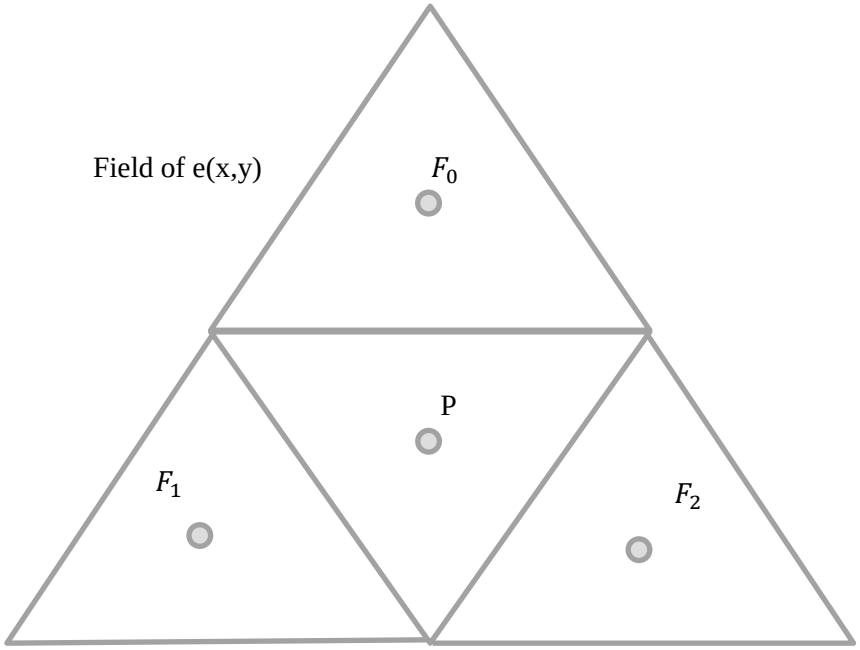


Figure 0.1 Equilateral triangular mesh consisting of cells P,F_0,F_1,F_2,in the field of e(x,y,t)

For an equilateral triangular grid shown above, by discretising the equation using finite volume method and explicit time scheme as done previously for T, it can be shown that

$$e_P^{n+1} = e_P^n + D\sqrt{3}\sum_F (e_F^n - e_P^n)$$

Where $D = \frac{\alpha \Delta t}{\Delta V_p}$ for an equilateral triangular mesh.

General approach

The error *e*(x,y,t) field is converted into its fourier transformation in two dimensions. If the gain of each component of the Fourier transform of the error is less than 1 than the series will be stable. A condition involving D is derived and this becomes the condition for stability of the explicit scheme.

Take the Fourier transform of the error field.

$$Let\ e(x,y,t_n) = \sum_{m,n=-\infty}^{m,n=\infty} C_{mn}(t_n) e^{\frac{im\pi x}{L_x}} e^{\frac{in\pi y}{L_y}} = \sum_{m,n} C_{mn}^{t_n} E_{m,n}$$

$$\text{where} E_{m,n} = e^{\frac{im\pi x}{L_x}} e^{\frac{in\pi y}{L_y}}$$

$$Substitue\ e_P^n = \sum_{m,n} C_{mn}^n E_{mn_P}\ and\ e_F^n = \sum_{m,n} C_{mn}^n E_{mn_F}\ and\ e_P^{n+1} = \sum_{m,n} C_{mn}^{n+1} E_{mn_P}$$

$$e_P^{n+1} = \sum_{m,n} C_{mn}^{n+1} E_{mn_P} = \sum_{m,n} C_{mn}^{n} \left((E_{mn_P} + D\sqrt{3} \sum_{F} (E_{mn_F} - E_{mn_P}) \right)$$

Let gain of m[th] and n[th] term be $G_{mn} = 1 + D\sqrt{3} \sum_F \left(\frac{E_{mn_F}}{E_{mn_P}} - 1 \right)$

$$\sum_{m,n} C_{mn}^{n+1} E_{mn_P} = \sum_{m,n} C_{mn}^{n} E_{mn_P} G_{mn}$$

Since this equation is linear (the behaviour of each term of the series is the same as series itself), it is enough to consider the growth of error of a typical term. With no loss of generality consider the m,n the term. As E_{mn_P} is independent of time, the series will converge at any point P, if gain of individual terms $|G_{mn}|$ is less than one.

$$|G_{mn}| < 1 \forall (m,n)$$

Since above equation has no co-ordinate system of its own we construct our 2-D coordinate system on a triangular structured grid to capture the field.

$$\frac{E_{mn_F}}{E_{mn_P}} = \frac{e^{\frac{im\pi x_{F_i}}{L_x}} e^{\frac{in\pi y_{F_i}}{L_y}}}{e^{\frac{im\pi x_P}{L_x}} e^{\frac{in\pi y_P}{L_y}}} = e^{i\frac{m\Delta x_{F_i P}\pi}{L_x}} e^{i\frac{n\Delta y_{F_i P}\pi}{L_y}}$$

dropping m,n from Gm,n

$$G_{mn} = G = 1 + D\sqrt{3} \sum_{F_i} \left(e^{i\frac{m\Delta x_{F_i P}\pi}{L_x}} e^{i\frac{n\Delta y_{F_i P}\pi}{L_y}} - 1 \right) \ldots (1)$$

Let $A_0 = \frac{\Delta x_{F_0 P}\pi}{L_x}$ *and* $B_0 = \frac{\Delta y_{F_0 P}\pi}{L_y}$ similarly for A1 & *A2*

(1) Transforms to

$$G = 1 + D\sqrt{3}\left(e^{i(mA_0 + nB_0)} + e^{i(mA_1 + nB_1)} + e^{i(mA_2 + nB_2)} - 3 \right)$$

For a given coordinate system, grid and domain A_0,B_0… are constants

Where A_1,B_1,… are constants for given inclination of the coordinate system with respect to the grid. Let

$$mA_0 + nB_0 = P_0$$
$$mA_1 + nB_1 = P_1$$

$$mA_2 + nB_2 = P_2$$

Since P is centroid

$$A_0 + A_1 + A_2 = 0 \; \forall \; (m, n)$$
$$B_0 + B_1 + B_2 = 0 \forall \; (m, n)$$

$$\Rightarrow \; P_0 + P_1 + P_2 = 0 \; \forall \; (m, n)$$

$$G = 1 + D\sqrt{3}\left(e^{iP_0} + e^{iP_1} + e^{iP_2} - 3\right) \tag{1}$$

$$\text{Let } G = 1 + D\sqrt{3}(-a + ib) \tag{2}$$

Comparing (1) and (2)

$$a = 3 - (\cos P_0 + \cos P_1 + \cos P_2) \; and \; \; b = (\sin P_0 + \sin P_1 + \sin P_2)$$

Applying condition of convergence that the gain of errors be less than one on (2).

$$|G| < 1 \Rightarrow \left|1 - D\sqrt{3}a + iD\sqrt{3}b\right|^2 < 1$$

$$\Rightarrow (D\sqrt{3}a)^2 - 2D\sqrt{3}a + 1 + (D\sqrt{3}b)^2 < 1$$

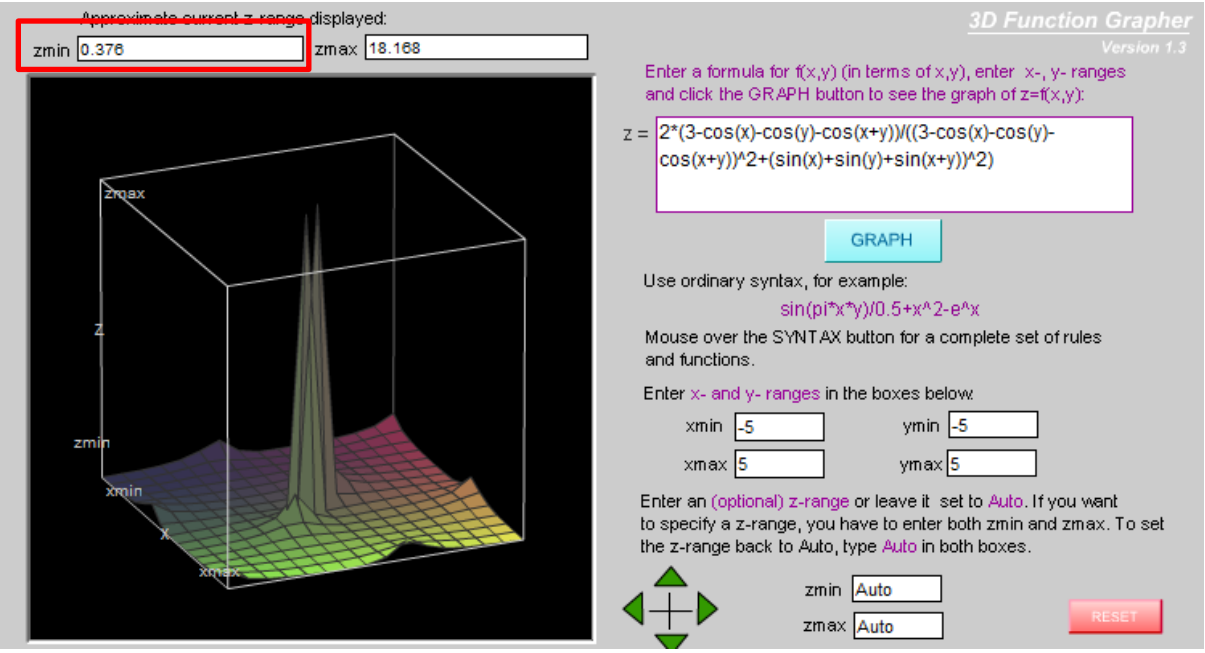
$$\Rightarrow |G| < 1 \Rightarrow 0 < D\sqrt{3} < \frac{2a}{a^2 + b^2} = \; f(P_0, P_1, P_2)(say)$$

$$Where \; f(P_0, P_1, P_2) = \frac{2a}{a^2 + b^2}$$

$$= \frac{2(3 - \cos P_0 + \cos P_1 + \cos P_2)}{((\cos P_0 + \cos P_1 + \cos P_2 - 3)^2 + (\sin P_0 + \sin P_1 + \sin P_2)^2)}$$

$$for \sum_{i=0}^{2} P_i = 0, \qquad min\left(f(P_0, P_1, P_2)\right) = 0.376$$

Using web tool to plot http://www.math.uri.edu/~bkaskosz/flashmo/graph3d2/



From eqn A3.17 and A3.15 we get stability criteria.

$$|G| < 1 \Rightarrow 0 < D\sqrt{3} < 0.376 < \frac{2a}{a^2 + b^2}$$

$$D < \frac{0.376}{\sqrt{3}} = 0.217 \forall\ cells\ \Rightarrow D_{max} = \frac{\alpha \Delta t}{\Delta V_{min}} < 0.217$$

Conclusion

A Von Neuman stability analyses was performed and a theoretical limit was obtained for ratio of the time step and cell volume. For diffusion equation which is a parabolic equation and simulated with explicit time discretization scheme in a triangular equilateral mesh, the theoretical ratio of the product of diffusivity α and the time step Δt with respect to the minimum allowed volume $\Delta V_{min}$of the cell is approximately 0.217. This condition is a necessary condition for stability. A simulation was performed for the initial mesh. Observed value of D_{max} = 0.22 closely matched the experimental value.

$$D_{max} = \frac{\alpha \Delta t}{\Delta V_{min}} < 0.217$$

Chapter 1 Introduction

1.1 Motivation

Differential equations appear in all nearly all branches of engineering. Many times they are mostly partial multi-variable differential equations. Numerical Methods are most of the times used to solve multi-variable partial differential equations, since analytical solution is not readily available. They are created by discretizing the domain and the partial differential eqn. They often do not exactly match analytical solution and difference between them is called error in the solution which can be predicted but not exactly found. The errors are dependent four factors the method, the problem being modeled, local discretized domain size and the region in the domain. Out of these, the local discretized domain size is the only parameter that can be modified as suited. For a particular problem if the errors are high in a particular region of the domain, they can be reduced if the local discretized domain size in the entire region of the domain is reduced. This is called refinement of the domain. However this refinement significantly increases computational time and hence cost. Hence it is wiser to selectively refine the regions of the domain where the errors are high although even this results in some increase in computational time. But this increase, in computational time relatively less than that which would have been required of the entire domain had been refined for the same level of accuracy. Thus, we need an adaptive grid refinement methodology that can refine the domain wherever required and ensuring minimum use of computational time without compromising on accuracy of the solution. This work is about developing such a methodology.

1.2 Description of problem

In this part, we find out a methodology for a unsteady diffusion equation 1.1 which is a parabolic differential equation. The problem is that we are going to find out the solution this equation within the accuracy limit of (say) x%. While direct observation of computational optimality is not done in this work, but when accuracy is hovering around the accuracy limit in most of the domain of the problem, we qualitatively conclude that the method is computationally optimum to a good degree.

Let T(x,y,t) be a conserved quantity S be a source term. α is the diffusion coefficient. We are given a domain, its boundary condition and initial condition on it. We have to find the solution to this parabolic differential equation with the error just hovering just below x% of accuracy.

$$T_t = \alpha \nabla^2 \mathrm{T} + \mathrm{S}(\mathrm{x}, \mathrm{y}, \mathrm{t}) \qquad 1.1$$

To arrive at a methodology that can solve the unsteady diffusion problem using a numerical method with adaptive grid refinement strategy, the important steps are

1. Discretization

   Here the equation and domain are converted into a from that is numerically solvable.

2. Error analysis

   To reduce the errors we must know how they are dependent on the problem being modeled, the discretization of equation and domain, local discretized domain size and the region in the domain.

3. Refinement Indicator

   This is the parameter that decides whether to refine or coarsen a particular region in the domain such that the accuracy is maintained while retaining computational optimality.

4. Domain refinement methodology

   When a particular region in the domain is to refined or coarsened, this part performs those operations on that particular region such that the new discretized domain is acceptable to the discretized equation.

1.3 Layout of the work

- An extensive Literature survey is done in Chapter 2 .
- The numerical method for solving 2-D diffusion equation is presented Chapter 3 .
- The discretization and the data structure of the space domain is described here in Chapter 3 .
- An error analysis and of the numerical method is done chapter 4.
- An indicator to modify the discretized domain is developed in Chapter 4 .
- A grid modifying methodology is developed here in Chapter 5 .
- The overall procedure to solve the diffusion equation numerically with adaptive grid refining capability is described here in chapter 6.
- The result of the methodology is described in Chapter 7 .

Chapter 2 Literature survey

2.1 Introduction

The goal of the project is to solve the diffusion equation over any domain while keeping a certain level of accuracy and also maintaining good computational speed. To solve the equation we have to resort to numerical methods since analytical solution will not be readily available. To do this we have to bring the equation into a form which is numerically solvable. This is called discretization. Since numerical methods are modeled on a domain, domain also has to be discretized. No numerical solution can exactly match the analytical solution. If the numerical method is not being accurate enough, some appropriate changes will have to be made to it so that it reaches a certain minimum level of accuracy. These changes are made to the domain. At the same time, some modifications can be made to the domain such that computational time is reducible without reducing accuracy. To do this, some parameters are created that will tell us whether any of these circumstances are present in the method. These are the most important steps in the Adaptive refinement strategy.

2.2 Discretization of the equation and domain

In this part, the differential equation is brought into a form which is numerically solvable. This is called discretization of the equation. The equations encountered in engineering normally involve variables that are functions of space and time. The various ways of Discretizing parabolic differential equation and space-time domain that are found in literature are given below.

2.2.1 Temporal Discretization of equation

The equation 2.1 is discretized, such that the equation is numerically evaluable in the time domain. The original equation is

$$T_t = \alpha \mathrm{O}(T) + S(t) \tag{2.1}$$

where O is some operator independent of the variable t. Integrate on both sides with respect to t.

$$\int_{n}^{n+1} T_t dt = \int_{\mathrm{n}}^{\mathrm{n+1}} (\alpha \mathrm{O}(T) + S) dt \tag{2.2}$$

$$T^{n+1} - T^n = (\alpha \mathrm{O(T)} + \mathrm{S})^{n+1} f + (\alpha \mathrm{O(T)} + \mathrm{S})^n (1 - f) \tag{2.3}$$

$$where\ f \in [0,1]$$

There are three well developed schemes in literature, explicit (f=0), Crank Nicolson (f=0.5) and Implicit (f=1). We will be using explicit scheme in this work.

The time domain is divided into discrete time steps of Δt_n where n is iteration number.

2.2.2 Spatial discretization of equation

This part deals with the Discretizing the equation so that the equation is numerically evaluable in the spatial domain. These discretization are broadly classified into three types

1) Finite Element methods (FEM)
2) Finite Difference methods (FDM)
3) Finite Volume methods (FVM)

All three of these methods have been developed in detail. The Finite volume method is good for CFD applications since there is a requirement of large number of cells. Also this method maps the conservation of physical properties in realistic way. We will be using Finite Volume Method because the long term goal of this approach is to head towards Naviers Stokes. Works by Jasak 1996, Muzaferija and Gosman 1997, Prabhudharwadkar 2007 have used Finite Volume method for solving the equations.

In this method, the equation is solved for a given control volume. Only interactions across the boundaries of the control volume are taken into account. We integrate the differential equation 1.1 across the control volume or we can solve for a conserved quantity such as T across any control volume V_P with surface area A_f.

$$\frac{\delta(\rho c T)_{avg}}{\delta t}\Delta V_p = \oint_{A_f} k\vec{\nabla}T.d\vec{A}_f + S_{avg}\Delta V_p \qquad 2.4$$

T_{avg} is the any conserved quantity and S_{avg} volumetric source term both of which are averaged out over the control volume, k is the diffusion coefficient, ρc is the density of the conserved quantity,ΔV_p is the volume of the control volume. Equation 2.4 transforms to 2.5 where α is the diffusion co-efficient.

$$\frac{\delta T_{avg}}{\delta t} = \frac{\alpha}{\Delta V_p}\oint_{A_f} \vec{\nabla}T.d\vec{A}_f + S_{avg} \qquad 2.5$$

2.2.3 Discretization of the space domain

The space domain is discretized by dividing it into a number of sub-domains that have well defined geometrical shapes in the Cartesian coordinate system. This discretized space is called a grid or mesh. The corners of the geometrical shape are called nodes. Each of them, have centroidal point which is called cell center. The grids that are commonly used in literature are triangular and quadrilateral. In this work a triangular unstructured mesh has been used in this work. The various methods of generating triangular unstructured meshes over a general domain are Wetherill's algorithm (Appendix 1 Grid generation) and Rupert's algorithm (Wikipedia 2011).

2.3 Indicators

Indicators are parameters that indicate whether the domain divisions are good enough so that the errors in the numerical method are below a certain limit. A good refinement indicator is extremely vital to accurately capture the zones that require more attention. It has to indicate whether an element is to be refined or coarsened. There are different types of indicators that are available in literature. Jasak 1996 has discussed some of these in detail.

2.3.1 Gradient based indicators

The concept behind using this indicator is to refine regions where large gradients are occurring. In order to identify the regions that require grid refinement, a sensor must be defined. The sensor used is based on gradients of flow properties. Its general definition could be expressed as

$$(sensor)_i = \frac{|\nabla\zeta|}{|\zeta_{max} - \zeta_{min}|} \qquad 2.6$$

where ζ_{max} and ζ_{min} are the maximum and the minimum values on the whole flow-field and $\nabla\zeta$ is the magnitude of the gradient of the ζ property in a control volume which is function of space and time. The refinement in the space domain at point i is a function of sensor. This indicator has been used by Walter et al. 2005, Silva et al 2000.

2.3.2 Richardson extrapolation based error indicators.

The solution is solved twice once on a coarse domain and fine domain. The difference between the two numerical solutions is taken as the error and normalized error is found out which is treated as the refinement indicator. This indicator has been used by Coeiho and Argain 1997

2.3.3 Taylor series approximation based error indicators

The objective of adaptation is to maintain a desired accuracy throughout the domain by varying the mesh and time step size. Errors in the solution result from the approximations done in the governing equations. These errors can be predicted by assuming the solution to be second order Taylor series and solving the exact governing equation for the control volume for this solution. These errors represent the leading truncation error terms in the Taylor series. Thus they will be zero for a linear solution. These errors are then used as refinement indicators. This type of indicator has been used in the work done by Prabhudharwadkar 2007, Muzaferija and Gosman 1997.

2.3.4 Moment error indicator

The exact equation is

$$T_t = \alpha\nabla^2\mathrm{T} + \mathrm{S(x,y,t)} \qquad 2.7$$

Let $m_T = \frac{T^2}{2}$ be the second moment of T. Above eqn 2.7 must satisfy even for higher moments T in the equation 2.8.

$$m_{T_t} = \alpha\nabla^2 m_T - \alpha\vec{\nabla}T.\vec{\nabla}T + \mathrm{S(x,y,t)T} \qquad 2.8$$

An estimate of the error is derived from the properties of the exact solution. The lower (eqn 2.7) and a higher (eqn. 2.8) moment solution of the equation are compared and the difference between them is used to calculate the error indicator.

We are restricting to second moment of T. The diffusion equation is solved for m_T and T and the difference between them is taken as the moment error. This is then normalized and an error indicator is formed. The details of this approach have been given by Jasak 1996.

2.3.5 Residual Error indicator

The FVM discretization is derived from the integral form of the governing equation over the control volume. The numerical solution consists of cell center and cell face values that satisfy the original equation over each control volume. Cell face values are determined using an interpolation practice consistent with discretization. But depending on where you interpolate from, there can be inconsistencies that arise in the interpolated values. The difference in these interpolated values is exploited to estimate the possible error in the solution. In the present work ϕ can represent the gradient of T. The details of this indicator have been given by Jasak 1996.

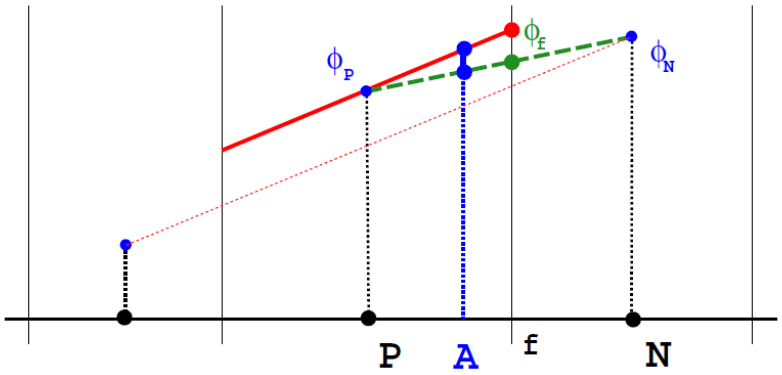


Figure 2.1 Inconsistency interpolation of ϕ_f at A (Jasak, 1996)

2.4 Mesh modification methods

To control the error, the indicator recommends some changes be done to the mesh. This part performs those changes and ensures that the new mesh can be accepted by the numerical method. There are a variety of ways one can modify the mesh which are presented as follows.

2.4.1 Node movement method

In this method, the mesh points are moved closer in the region where a smaller mesh size is required. The figure below shows how the mesh is getting distorted. Work done by Baines 1998 uses this.

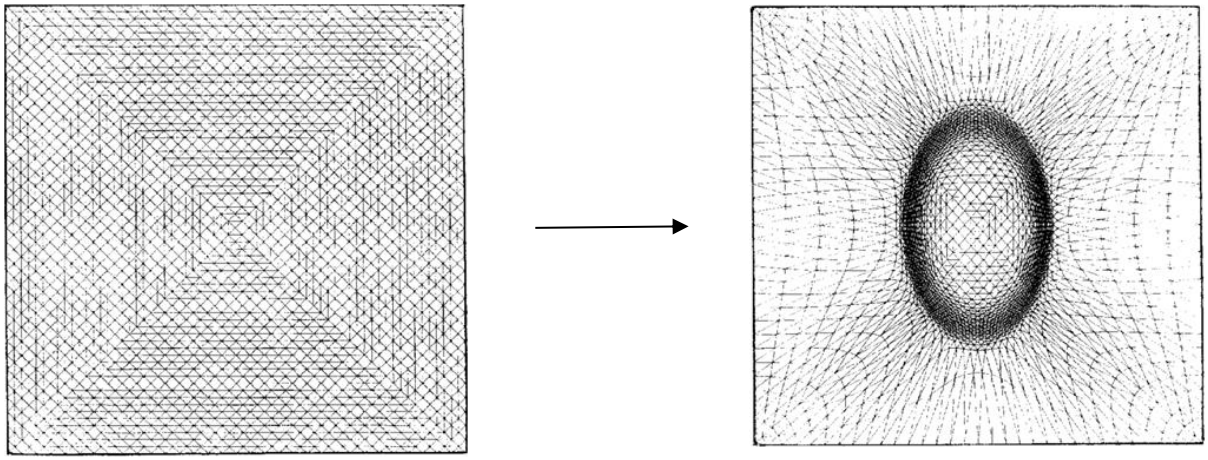


Figure 2.2 Node movement method (Baines, 1998)

2.4.2 Node and edge addition methods.

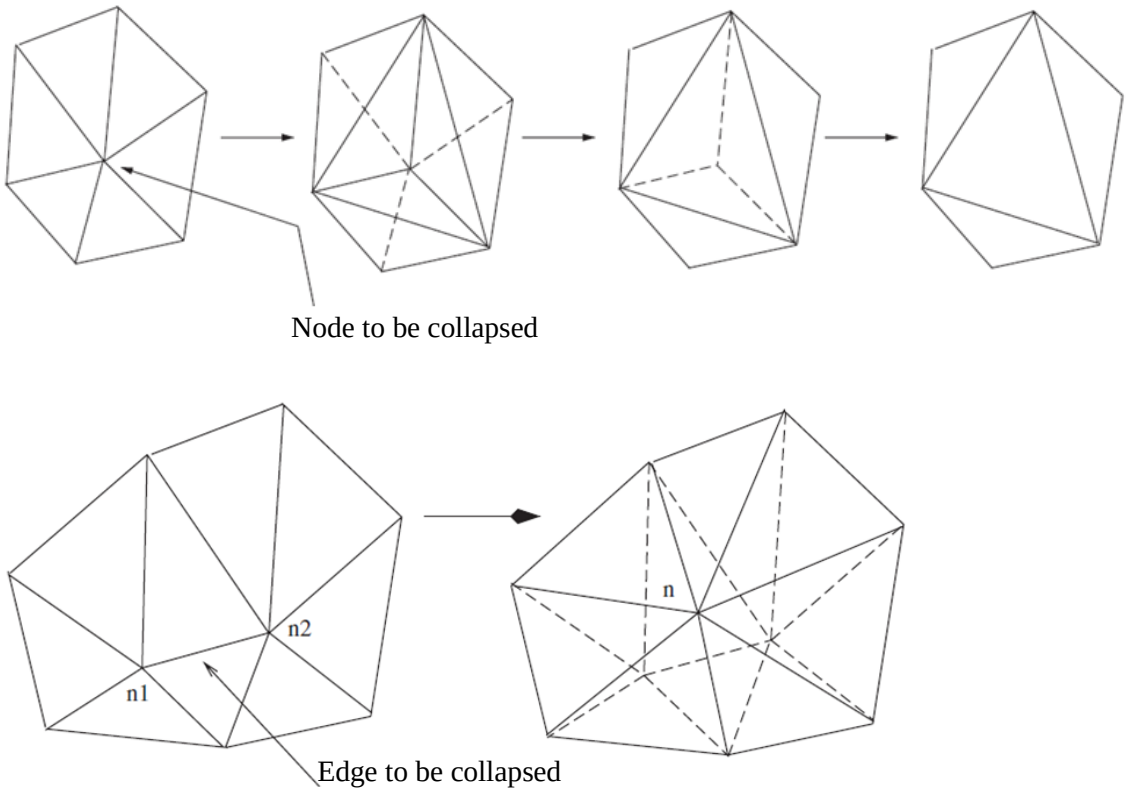


Figure 2.3 Node and edge addition methods (Walter et al. 2005)

In this class of methods, nodes are added using the *Bowyer* algorithm (Apendix 1) so that the triangulation has the Delaunay property (Appendix 1). Walter et al. 2005 has used this method to modify the grids.

2.4.3 Element based refinement methods

This method has been used by Waanders and Carnes 2010, Prabhudharwadkar 2007, Oden et al. 1986, Wang and Ragusa 2011, Jones and Plassmann 1997, Tomlin et al. 2000. In this class of methods, individual control volumes of the mesh are refined or coarsened in certain predetermined ways. More control volumes of similar shape are added into existing control volumes. During the refinement control volumes that are not to be refined may get affected in the process due to requirements of additional faces. Conversely all control volumes that have to be coarsened may not get coarsened due to constraining conditions.

In the method below by each triangle is split into 4 triangles by bisecting each side of the original triangle. The neighbors of the original triangle will now have two neighboring cells. This method has been used by Waanders and Carnes 2010, Wang and Ragusa 2011, Ripley and Yovanovich 2003.

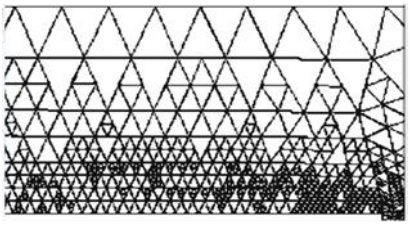


Figure 2.4 Element based method resulting in Non-conformal triangular mesh (Waanders and Carnes 2010)

Oden et al 1986, Coeiho and Argain 1997, Muzaferija and Gosman 1997, Jasak 1996 have used this method for a square structured domain.

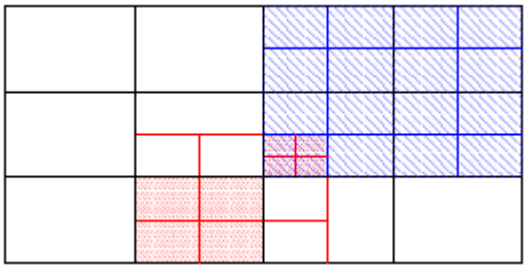


Figure 2.5 Square non-conformal mesh (Jasak, 1996)

The above methods result in non conforming meshes (refer Figure 2.4 and Figure 2.5). The mesh can be made conforming if the neighbors of the original triangle can further be bisected into two triangles (refer Figure 2.6) such that each triangle as at max three neighbors. Thus, no refined mesh angle can be less than half the smallest initial mesh angle. This has been done by Silva et al. 2000, Prabhudharwadkar 2007, Tomlin, et al. 2000.

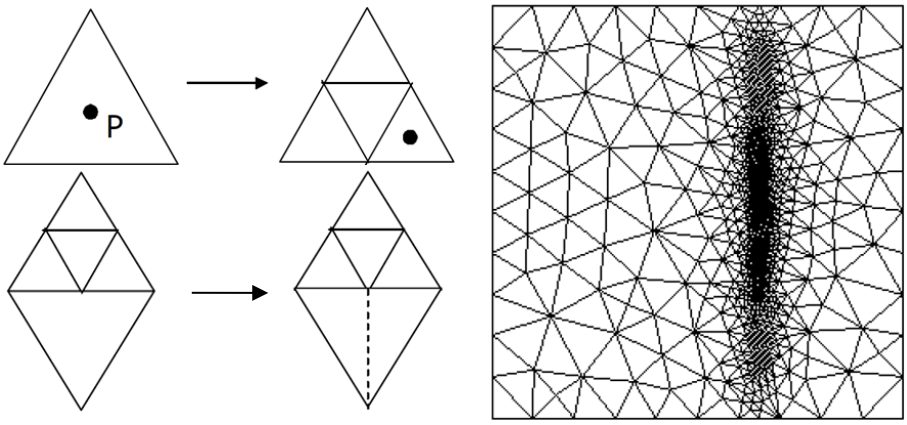


Figure 2.6 Procedure to avoid non conformal mesh (Prabhudharwadkar, 2007)

2.4.4 Bisection Method

Maria and Rivara 1984, Jones and Plassmann 1997 have used an algorithm that can refine mesh by bisecting the triangles as shown in figure below. In this algorithm, any triangle to be refined is bisected into two triangles. In the Figure 2.7 marked triangles are similar. This can again result in non-conforming meshes. Like the previous method (refer Figure 2.6). Thus additional refinements may have to be performed to obtain conforming mesh.

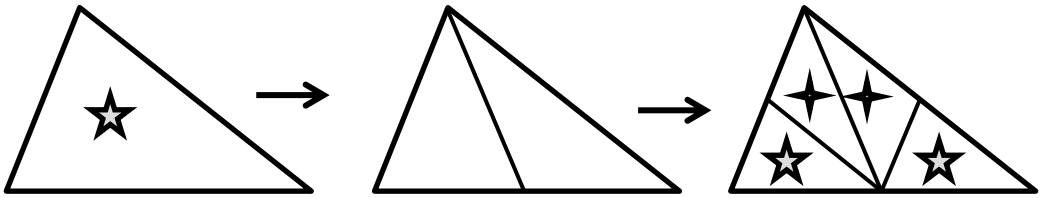


Figure 2.7 Bisection method

2.5 Conclusions of the Literature Survey

1. The work has to be done on three main fronts viz discretization, refinement indicator and domain modification.
2. In the discretization part (2.2), we chose the explicit scheme since it does not require simultaneous equations to be solved at every time step. We chose Finite Volume Method for discretization of equation because it directly modeled the conservation of the conserved physical quantity (T). We chose unstructured triangular domain for solving the differential equation since it could capture domain of any complexity.
3. A brief over view of the various error indicators has been done in section 2.3.
4. We chose the element based method as our grid modification methodology (2.4) due to its simplicity, mesh quality and low error generation.

2.6 Objectives of the report

1. The diffusion equation is to be solved on unstructured triangular space domain. The space domain is often a complicated domain to map so a good methodology to generate triangular grid on any domain has to be studied and coded (Appendix 1). Also the data structure and the connectivity of the points have to be defined (chapter 3.5).
2. A discretization scheme for the discretized Finite Volume diffusion equation has to be created on unstructured triangular domain and its boundary(Chapter 3.2 and 3.3)
3. One of the main objectives of the work is to reduce errors in the solution of the diffusion equation. Thus an error analysis has to be done in order to find out what the errors depend on. (Chapter 4).
4. An indicator has to be developed to tell whether a particular region in time and space domain needs more grid points or not (Section 4.6).
5. Since the space domain is triangular unstructured domain, a separate grid modification methodology has to be developed that can refine or coarsen the space domain wherever required (Chapter 5).
6. The overall methodology (6.1) working together and its effectiveness on some test problems has to be done (Sections 6.2,6.3,6.4).

Chapter 3 Discretization

3.1 Motivation for discretization

Consider a parabolic differential equation

$$T_t = \alpha \nabla^2 \mathrm{T} + \mathrm{S}$$

Our aim is to find a solution to the differential equation in a give domain of any shape. Its analytical solution in general will not be available or very difficult to find. In such cases we have to numerically solve the equation. The differential equation in its present form is not numerically solvable. We have to bring it into a form which numerically solvable. This is called discretization of the equation.

3.2 Evaluation of Space integrals

We had chosen the FVM method of discretization for the parabolic equation

$$T_t = \alpha \nabla^2 \mathrm{T} + \mathrm{S} \tag{3.1}$$

The FVM equation for the above parabolic differential equation is

$$\frac{\delta T_{avg}}{\delta t} = \frac{\alpha}{\Delta V_p} \oint \vec{\nabla} T . d\vec{A}_f + S_{avg} \tag{3.2}$$

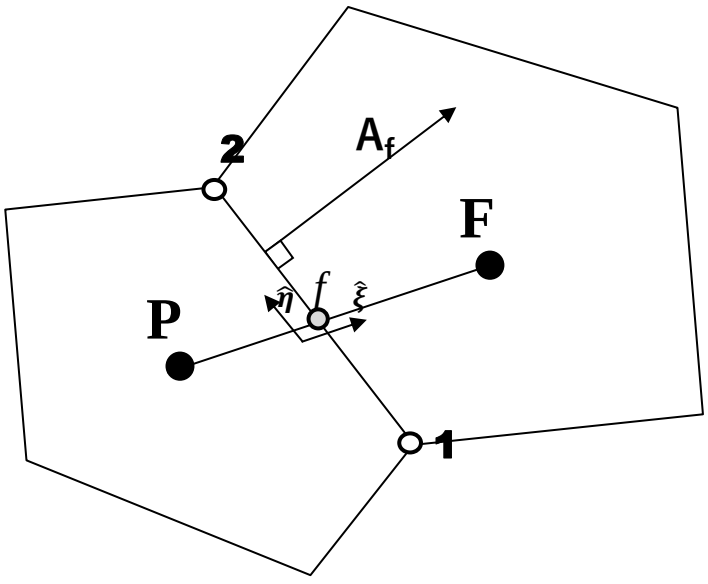


Figure 3.1 Control volumes P and F

We make our first approximation about space integrals in eqn 3.3.

$$\oint_f \vec{\nabla} T . d\vec{A} \approx \sum_f \vec{\nabla} T_f . \vec{A}_f \tag{3.3}$$

Using vector algebra, we can show that

$$\vec{\nabla} T = \frac{T_\xi - wT_\eta}{1 - w^2} \hat{\xi} + \frac{T_\eta - wT_\xi}{1 - w^2} \hat{\eta} \tag{3.4}$$

Where $w = \hat{\xi} . \hat{\eta},\ T_\xi \ and\ T_\eta$ are gradients in those directions $\hat{\xi}\ and\ \hat{\eta}$ respectively.

Putting eqn 3.4 in 3.3

$$\vec{\nabla} T_f . \vec{A}_f = \frac{T_\xi - wT_\eta}{1 - w^2} \hat{\xi} . \vec{A}_f + \frac{T_\eta - wT_\xi}{1 - w^2} \hat{\eta} . \vec{A}_f \tag{3.5}$$

$$Let\ C_\xi = \frac{\hat{\xi} . \vec{A}_f}{1 - w^2}\ and\ C_\eta = -\frac{w\hat{\xi} . \vec{A}_f}{1 - w^2}\ \text{ and put } \hat{\eta} . \vec{A}_f = 0 \tag{3.6}$$

We linearly approximate gradients in ξ and η directions

$$T_\xi \approx \frac{T_F - T_P}{\Delta \xi}\ and\ T_\eta \approx \frac{T_2 - T_1}{\Delta \eta} \tag{3.7}$$

Substitute eqn 3.7 and 3.6 in 3.5. eqauqte it to 3.3 to get

$$\oint_f \vec{\nabla} T . d\vec{A} \approx \sum_f \vec{\nabla} T_f . \vec{A}_f = \sum \left(C_\xi \frac{T_F - T_P}{\Delta \xi} + C_\eta \frac{T_2 - T_1}{\Delta \eta} \right) \tag{3.8}$$

Using stokes relations for Laplacian

$$\nabla^2 T \approx \frac{1}{\Delta V_P} \oint_f \vec{\nabla} T . d\vec{A} = \frac{1}{\Delta V_P} \sum \left(C_\xi \frac{T_F - T_P}{\Delta \xi} + C_\eta \frac{T_2 - T_1}{\Delta \eta} \right) \tag{3.9}$$

The use eqn 3.8 in eqn 3.2 to get final discretized equation 3.10.

$$\frac{T_P^{n+1} - T_P^n}{\Delta t} = \frac{\alpha}{\Delta V_P} \sum \left(C_\xi \frac{T_F^n - T_P^n}{\Delta \xi} + C_\eta \frac{T_2^n - T_1^n}{\Delta \eta} \right) + S \tag{3.10}$$

3.3 Boundary treatment

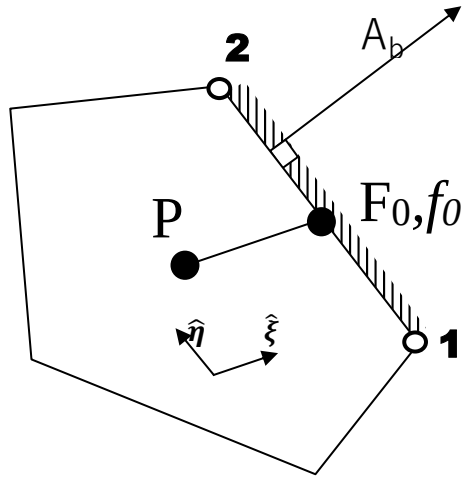


Figure 3.2 Control Volume P as a boundary cell

The same control volume equations 3.2 applies for the boundary.

T_{F_0} is interpolated between points 1 and 2.

$$T_{F_0} \approx \frac{T_1 + T_2}{2} \tag{3.11}$$

Flux integral across the boundary face A_b is approximated at f which is same as F_0 for the boundary.

$$\int_1^2 \vec{\nabla} T . d\vec{A}_b \approx \vec{\nabla} T_{f_0} . \vec{A}_b \tag{3.12}$$

Using vector eqn 3.4 in 3.12

$$\vec{\nabla} T_{f_0} . \vec{A}_b = \frac{T_\xi - wT_\eta}{1 - w^2} \hat{\xi} . \vec{A}_b + \frac{T_\eta - wT_\xi}{1 - w^2} \hat{\eta} . \vec{A}_b \tag{3.13}$$

$$\text{Let } C_\xi = \frac{\hat{\xi} . \vec{A}_b}{1 - w^2} \; and \; C_\eta = -\frac{w\hat{\xi} . \vec{A}_b}{1 - w^2} \quad \text{and put } \hat{\eta} . \vec{A}_f = 0$$

We linearly approximate gradients in ξ and η directions

Flux for the boundary is

$$\int_1^2 \vec{\nabla} T_{f_0} . d\vec{A}_b \;\; for \; boundary \approx \vec{\nabla} T_f . \vec{A}_b = C_\xi \frac{T_{F_0} - T_P}{\Delta\xi} + C_\eta \frac{T_2 - T_1}{\Delta\eta} \tag{3.14}$$

$$\sum \vec{\nabla} T_f . \vec{A}_f = \vec{\nabla} T_{f_0} . \vec{A}_b + \vec{\nabla} T_{f_1} . \vec{A}_{f_1} + \vec{\nabla} T_{f_2} . \vec{A}_{f_2} \quad 3.15$$

Put the above eqn in the finial discretized eqn

$$\frac{T_P^{n+1} - T_P^n}{\Delta t} = \frac{\alpha}{\Delta V} \sum \left(C_\xi \frac{T_F^n - T_P^n}{\Delta \xi} + C_\eta \frac{T_2 - T_1}{\Delta \eta} \right) + S \quad 3.16$$

For a general Robin boundary condition

$$aT + b \frac{\delta T}{\delta n} = C \quad 3.17$$

We implement this condition at F_0

$$aT_{F_0} + b \left(\frac{\delta T}{\delta n} \right)_{F_0} = c \quad 3.18$$

T_{Fo} is approximated in eqn 3.11

$$\frac{\delta T}{\delta n} = \vec{\nabla} T_{f_0} . \hat{A}_b = \frac{1}{A_b} \vec{\nabla} T_{f_0} . \vec{A}_b \quad 3.19$$

According to eqn 3.13 and 3.19

$$\frac{\delta T}{\delta n} = \vec{\nabla} T_{f_0} . \vec{A}_b = C_\xi \frac{T_{F_1} - T_P}{\Delta \xi} + C_\eta \frac{T_2 - T_1}{\Delta \eta} \quad 3.20$$

Substitute eqn 3.20 in 3.18 to get

$$aT_{F_0} + \frac{b}{A_b} \left(C_\xi \frac{T_{F_0} - T_P}{\Delta \xi} + C_\eta \frac{T_2 - T_1}{\Delta \eta} \right) = c \quad 3.21$$

3.4 Discretization of time domain

The time domain can be discretized independently of any other variable. The time domain is a single variable domain and its length is independent of the space domain. Thus discretization of time domain is very straight forward. It will be divided into a number of time steps (Δt_n) of different size which will be a function of iteration number.eg tϵ(t_1,t_2). t_1,t_2 are real constants.

3.5 Discretization of Space domain

The space domain most of the times has variables(x,y,z) whose domain boundaries are function of one another. Elliptic and trapezoidal domains are examples of such 2-D domains. Thus, the space domain is discretized by dividing it into a number of sub-domains that have well defined geometrical shapes in the Cartesian coordinate system. This is called a grid.

The various data structures that are used do characterize the field T(x,y) are given

A 2-D Cartesian co-ordinate system is described.

Point-

Identification index i →Every point is given a unique integer index for identification purposes.

x → it gets the 'x' coordinate value of the point with respect to the 2-D Cartesian system.

y → it gets the 'y' coordinate value of the point with respect to the 2-D Cartesian system.

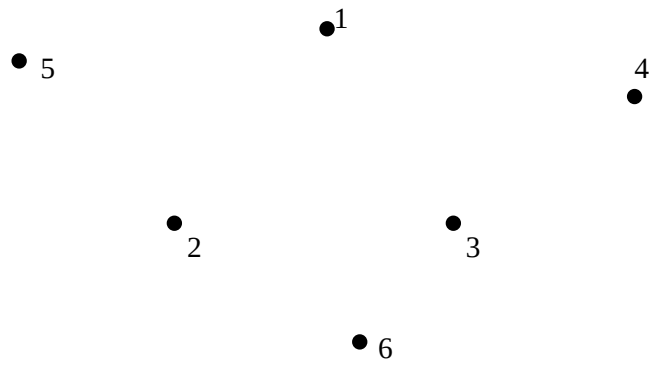


Figure 3.3 Set of discretized points of space domain

T → it gets the field value of at that point which is T(x.y).

Example

identification index	x label	y label	field value T
1	0	0	100
2	-10	-10	200
3	10	-10	300

Cells- It is triangular element which is characterized by three points.

Data structure

Cell identification index (j) →

Each cell is given a unique integer index for identification purposes.

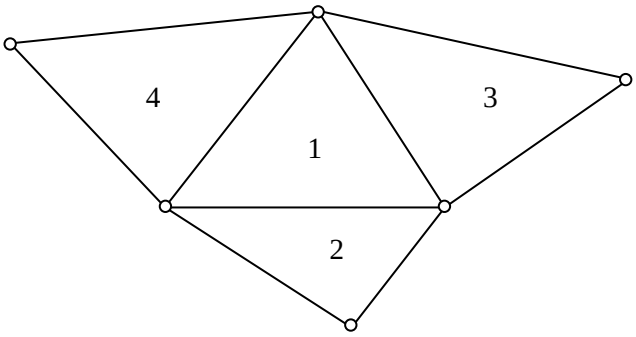


Figure 3.4 Cells connectivity and their indices

Point index P_0, P_1, P_2→These are variables indices that get the value from the indices of the points that make up triangle. In this case P_0=1, P_1=2, P_2=3.

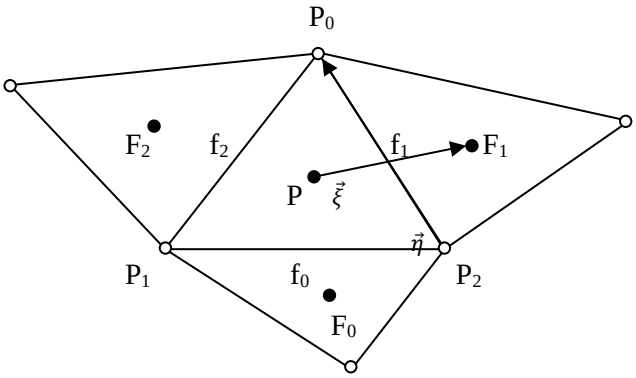


Figure 3.5 Showing the data structure of cell and its local labels.

Two triangles with unique identification indices can have at most two point indices that are same. Such triangles are identified as **neighbors** of each other.

Neighboring cell index F_0, F_1, F_2,→ These are variables indices that get the value of the indices of the cell that are neighboring to cell under consideration. The indices are given such that cell N_0 is opposite to point P_0, N_1 is opposite to point P_1 , N_2 is opposite to point P_2. In this case F_0=2, F_1=3, F_2=4.

cell centroidal point P →This is centroid of the points represented by indices P_0, P_1, P_2. Here P is a point of cell 1 and not a point index.

Face opposite to P_0 is labeled f_0. Cell reached by traversing face f_0 is labeled N_0. Face in 2-D is defined by two points. f_0,f_1,f_2 are just local labels that are used for illustration. They have no relevance in data structure.

Note that these are local labels with respect to a particular triangle.

For Triangle PQR→j=1, P_0=1, P_1=2, P_2=3, N_0=2, N_1=3, N_2=4,

Chapter 4 Error analysis and Indicators

4.1 Motivation for error analysis

In every discretization procedure some approximations are done to the differential equations to numerically solve them. These approximations result in generation of errors called discretization errors. Our goal is that the errors should be below a certain limit set by the user. To do that we must understand on what these errors depend on. Also once the errors are created, they propagate throughout the domain as iterations proceed. This error propagation needs to be thoroughly studied.

4.2 FVM approximations

Let T(x,y,t) be the true solution and the solution due to the approximations be $\bar{T}(x_i, y_i, t_n)$

Original eqn is

$$T_t = \alpha \nabla^2 T + S \tag{4.1}$$

Discretized eqn according to eqn 3.10 is

$$\frac{\bar{T}_P^{n+1} - \bar{T}_P^n}{\Delta t} = \frac{\alpha}{\Delta V_P} \sum_f \left(C_\xi \frac{\bar{T}_F - \bar{T}_P}{\Delta \xi} + C_\eta \frac{\bar{T}_2 - \bar{T}_1}{\Delta \eta} \right) + S \tag{4.2}$$

The following approximations were done to get eqn 4.2 from eqn 4.1

1) $T_t \approx \frac{T_P^{n+1} - T_P^n}{\Delta t}$ Linear time step approximation
2) $(\nabla^2 T)_P \approx \frac{1}{\Delta V_P} \int_{V_P} \nabla^2 T dV = \frac{1}{\Delta V_P} \oint_f \vec{\nabla} T . d\vec{A}_f$ Linear spatial Laplacian approximation.
3) $\oint_f \vec{\nabla} T . d\vec{A}_f \approx \sum \vec{\nabla} T_f . \vec{A}_f$ Face flux approximation.
4) $T_\xi \approx \frac{T_F - T_P}{\Delta \xi}$ and $T_\eta \approx \frac{T_2 - T_1}{\Delta \eta}$ Linear face gradient approximation

To find out the lowest order error we make use of Taylor series expansions till the second order. Direct expressions of above approximations are presented here. Detailed mathematical proofs are shown in Appendix 2 *Error analysis*.

Also the order of magnitudes with respect to time step and domain size of each of the errors is given below. Let δ be the discretized local length scale and Δt be the discretized time step at given iteration. When evaluating order of magnitude, all variables, integrals or expressions involving lengths or time steps are substituted in some powers of δ or Δt.

1) Linear time step approximation

$$T_t = \frac{T_P^{n+1} - T_P^n}{\Delta t} - E_t \tag{4.3}$$

$$E_t = T_{tt}\frac{\Delta t}{2} = O(\Delta t) \quad 4.4$$

$$T_{tt} = \alpha\nabla^2 T_t + S_t = \alpha^2\nabla^4 T + S_t + \alpha\nabla^2 \mathrm{S} \quad 4.5$$

2) Linear spatial Laplacian approximation

$$(\nabla^2 T)_P = \frac{1}{\Delta V_P}\oint_{V_P} \vec{\nabla} T.d\vec{A} + E_\delta = \frac{1}{\Delta V_P}\int_{V_p} (\nabla^2 T)dV + E_\delta \quad 4.6$$

$$E_\delta = -\frac{1}{\Delta V_P}\left((\nabla^2 T)_{xx}\frac{I_{xx}}{2} + (\nabla^2 T)_{yy}\frac{I_{yy}}{2} + (\nabla^2 T)_{xy} I_{xy}\right) \quad 4.7$$

Eqn 4.6 and 4.7 imply

$$(\nabla^2 T)_P = \frac{1}{\Delta V_P}\oint_{V_P} \vec{\nabla} T.d\vec{A} + E_\delta \quad 4.8$$

All derivatives in E_δ are evaluated at P (refer Figure 4.1) I_{xx} I_{yy} I_{xy} are constants that depend only on the shape and size of the triangular element in the mesh. Where centre of the coordinate system is at P.

$$I_{xx} = \int x^2 dV =, I_{yy} = \int y^2 dV\,, I_{xy} = \int xydV$$

And $I_{xx}, I_{yy}, I_{xy} = O(\delta^4)$

$$E_\delta = \frac{O(\delta^4)}{\Delta V_P} = O(\delta^2) \quad 4.9$$

3) Face flux approximation and 4)Linear face gradient approximation

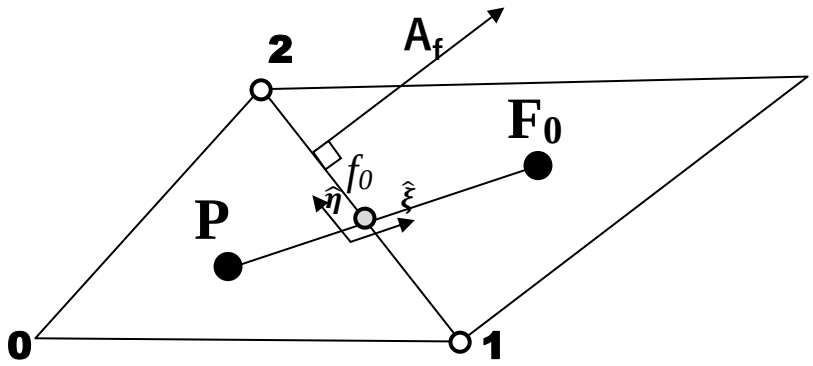


Figure 4.1 Control volumes P and F

$$where\ C_\xi = \frac{\hat{\xi}.\vec{A}_f}{1-w^2} = O(\delta)\ and\ C_\eta = -\frac{w\hat{\xi}.\vec{A}_f}{1-w^2} = O(\delta)\ and\ w = \hat{\xi}.\hat{\eta}$$

$$\oint_{V_P} \vec{\nabla}T.d\vec{A}_f \approx \sum_f \int_1^2 \vec{\nabla}T.d\vec{A}_f \qquad 4.10$$

$$\int_1^2 \vec{\nabla}T.d\vec{A}_f = C_\xi \frac{T_F - T_P}{\Delta\xi} + C_\eta \frac{T_2 - T_1}{\Delta\eta} + E_f \qquad 4.11$$

Where

$$E_f = \frac{(d_F - d_P)}{2} C_\xi T_{\xi\xi} + \frac{(d_2 - d_1)}{2}\left(C_\xi T_{\xi\eta} + 2C_\eta T_{\eta\eta}\right) = O(\delta^2) \qquad 4.12$$

All derivatives of E_f are evaluated at point f. All distances are from point f

Substituting approximations 1,2,3,4 in original parabolic eqn 4.1.

$$\frac{T_P^{n+1} - T_P^n}{\Delta t} = \frac{\alpha}{\Delta V_P} \sum_f \left(C_\xi \frac{T_F - T_P}{\Delta\xi} + C_\eta \frac{T_2 - T_1}{\Delta\eta}\right) + S_e + S \qquad 4.13$$

$$Where\ S_e = \frac{\alpha}{\Delta V_P} \sum_f E_f + \alpha E_\delta + E_t \qquad 4.14$$

In terms of order of magnitude dependence putting eqn 4.4, 4.9, 4.12 in 4.14

$$S_e = \sum_f \frac{O(\delta^2)}{O(\delta^2)} + O(\delta^2) + O(\Delta t) \qquad 4.15$$

4.3 Error propagation equation

Let T(x,y,t) be the true solution and the solution due to the approximations be $\bar{T}(x_i, y_i, t_n)$

Original eqn is. S is the source term.

$$T_t = \alpha\nabla^2 T + S \qquad 4.16$$

The discretized form of above equation 4.16 with higher order approximation is eqn 4.13.

$$\frac{T_P^{n+1} - T_P^n}{\Delta t} = \frac{\alpha}{\Delta V} \sum \left(C_\xi \frac{T_F^n - T_P^n}{\Delta\xi} + C_\eta \frac{T_2^n - T_1^n}{\Delta\eta}\right) + S + S_e \qquad 4.17$$

$$where\ S_e = \alpha \sum_f \frac{E_f}{\Delta V_P} + \alpha E_\delta + E_t\ .S_e\ is\ the\ higher\ order\ term$$

The discretized form of eqn 4.16 with lower order linear approximation is

$$\frac{\bar{T}_P^{n+1} - \bar{T}_P^n}{\Delta t} = \frac{\alpha}{\Delta V_P} \sum_f \left(C_\xi \frac{\bar{T}_F - \bar{T}_P}{\Delta\xi} + C_\eta \frac{\bar{T}_2 - \bar{T}_1}{\Delta\eta}\right) + S \qquad 4.18$$

Subtracting the lower order (4.18) from the higher order (4.17) gives us the error (e_p) propagation equation (4.19).

$$T = \bar{T} + e \Rightarrow e = T - \bar{T}$$

$$\frac{e_P^{n+1} - e_P^n}{\Delta t} = \frac{\alpha}{\Delta V} \sum \left(C_\xi \frac{e_F^n - e_P^n}{\Delta \xi} + C_\eta \frac{e_2^n - e_1^n}{\Delta \eta} \right) + S_e \qquad 4.19$$

4.4 Stability

In case of explicit scheme, as the time steps proceed, errors that are generated due to discretization may blow up due to error propagation in the equation.

We want to find out the condition when above series in eqn 4.19 diverges or converges.

In general for 2-D grids $A_f = \Delta\eta$. Let $D = \frac{\alpha \Delta t}{\Delta V_p}$.

For a special case of a **triangular equilateral grid**. Substituting the terms below in 4.19

$$\omega_f = \hat{\xi}.\hat{\eta} = 0, C_\xi = \frac{\hat{\xi}.\vec{A}_f}{1 - \omega_f^2} = A_f \, , C_\eta = -\omega_f \frac{\hat{\xi}.\vec{A}_f}{1 - \omega_f^2}, C_\xi = -A_f \, , \frac{A_f}{\Delta \xi_f} = \sqrt{3}$$

Eqn 4.19 transforms to series eqn

$$e_P^{n+1} = e_P^n + D\sqrt{3} \sum_F (e_F^n - e_P^n) + S_e \Delta t \qquad 4.20$$

Let us consider some initial error profile and that the error source term (S_e) is zero. To find out a condition for which the series does not diverge for any initial condition of error distribution, we find out the condition for which the amplification factor (G) per iteration for the errors is less than one. Eqn 4.20 transforms to

$$G = \left| \frac{e_P^{n+1}}{e_P^n} \right| = \left| 1 + D\sqrt{3} \sum_F (e_F^n - e_P^n) \right| \qquad 4.21$$

This is done through a von neumon Stability Analysis (Appendix 3)

$$G < 1 \Rightarrow D\sqrt{3} < 0.34 \Rightarrow D_{max} = \frac{\alpha \Delta t}{\Delta V_{min}} < 0.1964$$

Instability oscillations observed at $D_{max} \approx 0.21$ for equilateral grids.

$$\frac{\alpha \Delta t}{\Delta V_p} < 0.15 \qquad 4.22$$

Thus, $D_{max} \approx 0.15$ to keep a with factor of safety, on reasonably equilateral triangular grids.

4.5 Steady state error

4.5.1 Definition

Error propagation eqn 4.19 suggests that errors are constantly getting produced at every iteration. Errors are constantly getting dissipated also as they diffuse in the medium due to the boundary conditions. When the error production rate and the error diffusion rate for a control volume are equal that is called the steady state error for the source term. The Steady state error is the error distribution at that will, when errors are getting produced at a constant rate.

4.5.2 Significance

Let us say the current distribution is e(x,y) and E(x,y) is the steady state distribution for a error source term S_e(x,y). That means that e(x,y) will tend to E(x,y) in the next iteration at (x,y). It gives us an upper or lower bound for the given source term. It gives us a way to relate error production rate (S_e) and actual error (e). We can convert the discretized form of the error propagation (4.19) into a differential from of the equation (4.23) by assuming infinite precision.

$$e_t = \alpha \nabla^2 e + S_e \tag{4.23}$$

For a square domain of length L with constant error source term S_e which is independent of x,y,t error eqn is and Dirichlet boundary condition of zero.

$$\frac{\alpha e_{max}}{L^2 S_e} \approx 0.07 \tag{4.24}$$

The eqn 4.24 gives the maximum steady error for a given source term.

This error (e_{max}) depends upon

1. Domain length and shape.
2. Type of boundary condition.
3. The error source term and its distribution which depend on the length scale and time step.
4. The heat diffusion constant.

4.5.3 Bulk errors

Consider a equilateral grid. Let us say that each control volume in the grid has error source term of constant magnitude S_d but is surrounded by three neighbors of opposite sign of the error source term, like an equilateral checker board. Now there will be a steady state error distribution (e_d) whose mean will be approximately zero and deviation will be the value of the steady state error. Let us say that a constant error source term S_b is considered the corresponding error distribution will be (e_b) Let us they are considered together i.e. the error source term is

$$new\ error\ source\ term\ S_e = S_b + S_d \tag{4.25}$$

Since the equation 4.23 is linear net error is simply the superimposition of the two errors.

$$Net\ error\ e = e_b + e_d \tag{4.26}$$

Thus the net error e will have two components the bulk (e_b) and the second deviation (e_d) component. These arise from the superimposition of the corresponding source terms error source terms S_b ,S_d. The bulk errors are normally the major contributors to the errors since they contribute to error built up more than deviation component. So we would like to consider the bulk component (S_b) of the error source term as the major contributor to the total error.

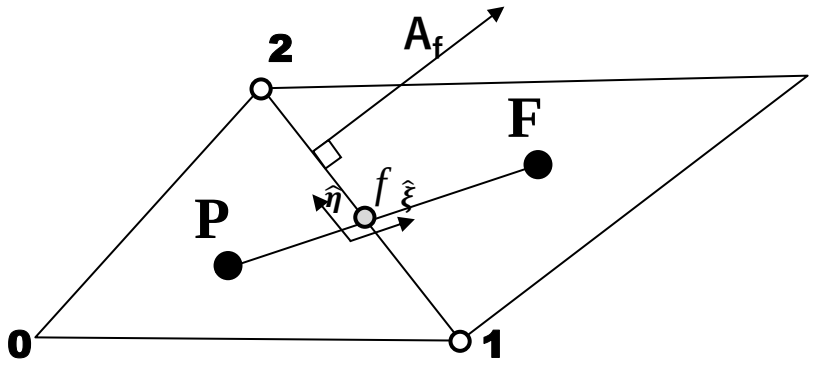


Figure 4.2 Control volumes P and F

Consider the error e_f due as error in measurement of flux for face f. Now the error in flux due to face f in control volumes P and F will have opposite sign because flux on one side will be entering the control volume and that on the other side will be leaving. Thus net error generated. Thus one side will be generating an error source term and the other side will be sinking the error source term just like the checkerboard example. But in this case their magnitudes are different.

Consider a control volumes P and F as one bulk control volume. Net bulk error source term per unit volume (S_b) due to face f in the Control volume P and F taken together is

$$S_b = \frac{(S_f)_P \Delta V_P + (S_f)_F \Delta V_F}{\Delta V_P + \Delta V_F} = \frac{\frac{E_f}{\Delta V_P}\Delta V_P - \frac{E_f}{\Delta V_F}\Delta V_F}{\Delta V_P + \Delta V_F} = 0 \tag{4.27}$$

Thus the net bulk contribution due to the flux errors is zero. The net bulk error is

$$S_b = E_\delta + E_t = O(\delta^2) + O(\Delta t) \tag{4.28}$$

4.6 Indicators as error Controllers

From steady state error equation solution (eqn 4.24) it can be seen that the maximum error (e_{max}) can be controlled if the error source term (S_e) is controlled. Empirically, the major contributors to maximum error (e_{max}) are the bulk error source terms (S_b). Error source term S_b depends on shape of the discretized space domain, curvature of the field T with respect to space and time, and discretized length

scales (δ) and time scale Δt (eqn 4.27). Out of these properties, only length (δ) and time scales Δt are in direct control of the user. Thus the order of dependency is $\delta, \Delta t \rightarrow S_b \rightarrow e_{max}$. To control S_b through δ and Δt we make use parameters called indicators. The idea is that if these indicators are within a prescribed limit then S_b is controlled and thus e_{max} can be controlled. If indicators go beyond prescribed limit, appropriate modifications of δ or Δt will have to be done to bring them back into the limit.

Thus indicators are parameters that tell us whether a cell or time step is to be refined, coarsened or unaltered so that the given level of accuracy is maintained throughout the domain without compromising on computational speed.

4.7 An ideal indicator

A good indicator is that which indicates the correct amount refinement required for a cell or time step. It should do this task with minimum number of computational operations. It should be stable i.e there must be no explosive refinement in any part of the domain.

4.8 Grid refinement indicators

4.8.1 Curvature based indicators

The discretization approximation made in a curve which is perfectly mapped by the discretized points will be very good and correspondingly errors will be low. Curves which area poorly mapped have a high curvature. Thus, the errors in the equation are a result of the curvatures effects. Wherever there is high curvature more points need to be added to capture the surface accurately. Thus this methodology has been done with a curvature based grid refinement indicator.

As shown in Figure 4.3 , the neighbor beside the cell P is linearly extrapolated to reveal the extent of the curvature. The difference (E_δ) between the extrapolation and the actual value is taken as the indicator for curvature. This is then normalized with a reference value T_{ref} and taken as the grid refinement indicator. T_{ref} is the scale of the variable T and can be taken as the difference between T_{max} and T_{min}.

$$T_{ref} = T_{max} - T_{min} \tag{4.29}$$

In Figure 4.3, P is the cell whose E_δ we are trying to find. E_δ with respect to each neighbor is found out and maximum is taken as the E_δ of the cell. The plane P_0, P_1, P_2 is extended to and the difference in the T co-ordinate and P is taken as E_δ.

$$E_\delta = T_P - \left\{T_F + (T_x)_F\,(x_P - x_F) + \left(T_y\right)_F (y_P - y_F)\right\} \tag{4.30}$$

$$where\ T_F = \frac{T_{P_0} + T_{P_1} + T_{P_2}}{3}\ and\ T_C = \frac{T_P + T_{P_1} + T_{P_2}}{3}$$

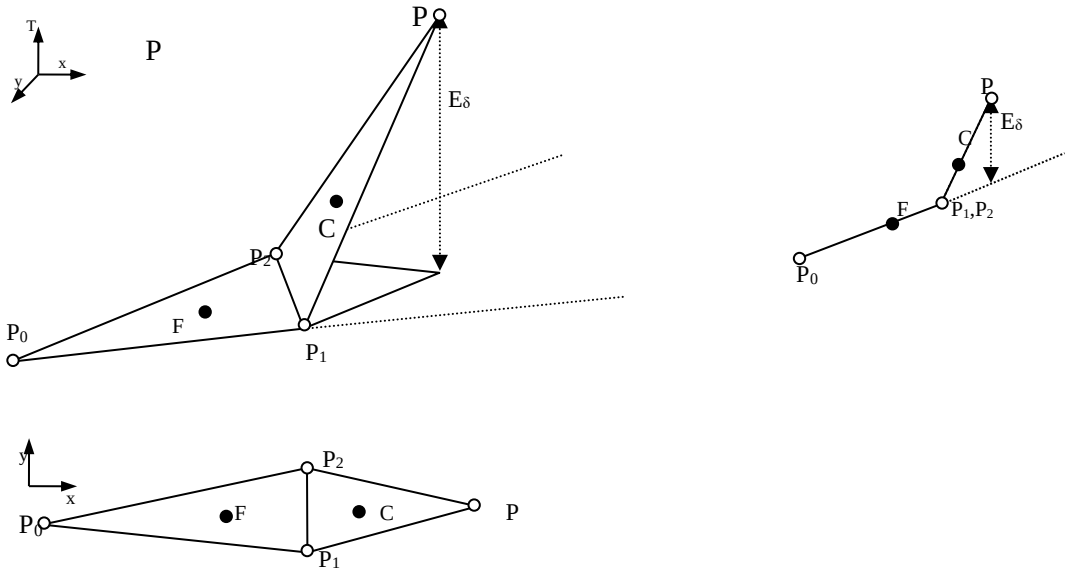


Figure 4.3 Curvature based grid refinement indicator

This parameter gives us an idea of the curvature of the field .It tells us how well the nodes have map the field. The limits have been designed for a error limit of 2% but can be extrapolated to any percentasge by multiplying the limits by a factor of x/2 where x% is the desired error limit. Also observations show that E_δ is $O(\delta^2)$. Therefore limit ratio must be greater than 4. We shall keep it 5 to avoid over refinement. Observations show that indicator limits of 1% and 5% work for the error limit of 2%.

$$(cosrsen)1\% \times \frac{x}{2} < \frac{E_\delta}{T_{ref}} < 5\% \times \frac{x}{2}(Refine) \qquad 4.31$$

4.9 Time step refinement indicator

This parameter is based on two limits.Δt_{max} should be such that.

3) The stability limit of the mesh should not be exceeded.
4) Change in temperature per time step should not exceed permissible error.

$$\frac{\alpha \Delta t}{\Delta V_P} \leq 0.2 \ \& \frac{\left|T_P^{n+1} - T_P^n\right|}{T_{ref}} < 2\% \times \frac{x}{2} \ \forall \ cells \qquad 4.32$$

Chapter 5 Mesh Modification Algorithm

In the previous chapter, the indicator has already indicated that the some cells in the mesh need to be altered so that the errors are kept below a limit. This part performs those alterations like refining and coarsening operations on the grid. Its input is a list of cells that have to be refined and coarsened.

A good grid modification methodology must modify the grids in such a way that original mesh quality is retained or improved. It should consume minimum computational time. It should be easy to program. It should have minimum impact on other places on the grid that were not necessary to be altered.

The grid refinement method chosen is an element based method. In this method the mesh is adapted by performing a local refinement by division of the elements. The major advantage of this approach is that it does not need the mesh generator once the initial mesh is generated. All the mesh changes are accommodated on the initial mesh, which is preferably a coarse one.

The additional data structure required is explained below. Their significance will come in the section

Point-Label 'coarse' for coarsening.
p1,p2 →they store indices of the parent points out of which this point is derived from.

Cells- Three additional point indices(midpt[0,1,2]) are given for storing mid points indices of sides.
Besides these ,some of the other labels it has are
Split→-1,0,1,2,3
Type→0,1,2
Booleans include, dM (double modified), midpt-allocation, coarse→all are 0 or 1.

5.1 Cell Refinement

The cells marked with '+' are the cells marked for refinement. Once a cell is marked for refinement, its edges are split by adding a node at its mid point. This cell is then split into four cells (refer Figure 5.1a). Now, when an element is marked for refinement and its neighbor is not, then this may cause a problem. The central cell in (refer Figure 5.1b) now has two neighbors on its same edge since the lower cell is split. In such situations this cell is split into two as shown (refer Figure 5.1c). All the cells on the boundary of the adapted zone generally have such configuration. The intermediate and final mesh configurations are shown for various cases under which other cells may get marked for division.

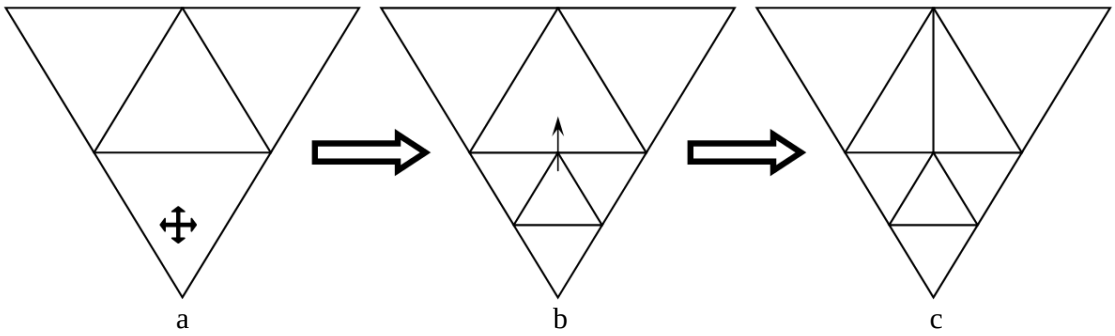


Figure 5.1 Cell marked for refinement

As the adaptation proceeds, if the cell which has already been split into two cells (refer Figure 5.2) is again marked for further refinement then it may lead to a much distorted cell shape. Direct refinement of such cells is avoided and some intermediate operations are performed to prevent mesh distortion. These are shown in the figure below.

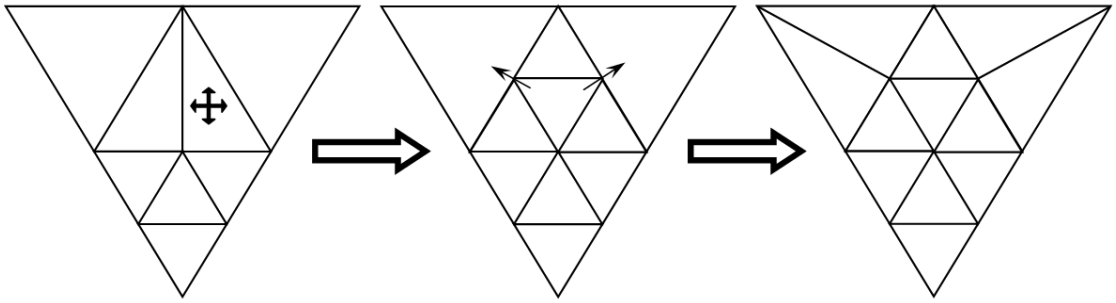


Figure 5.2 A half split cell is marked for further refinement

It can be seen that the moment such cell is marked for division, it first collapses back to its parent cell and then fully split into four cells. This procedure helps minimizing the number of distorted cells during adaptation. Such distortions may happen even if the neighbor of this cell is marked for refinement (refer Figure 5.3 and Figure 5.4).

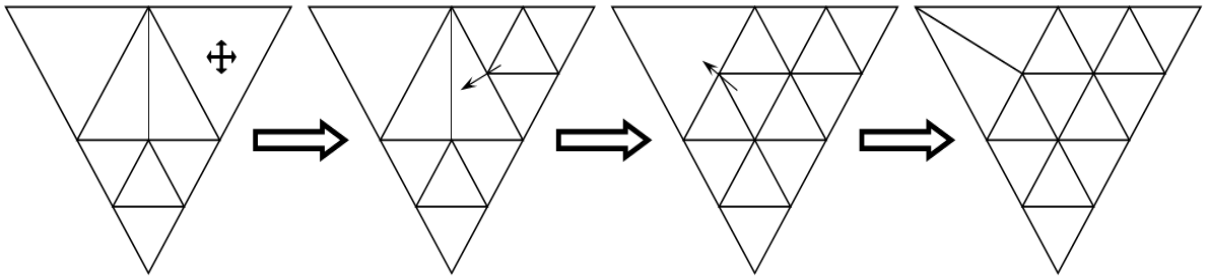


Figure 5.3 Longest side neighbor of a half split cell is marked for further refinement.

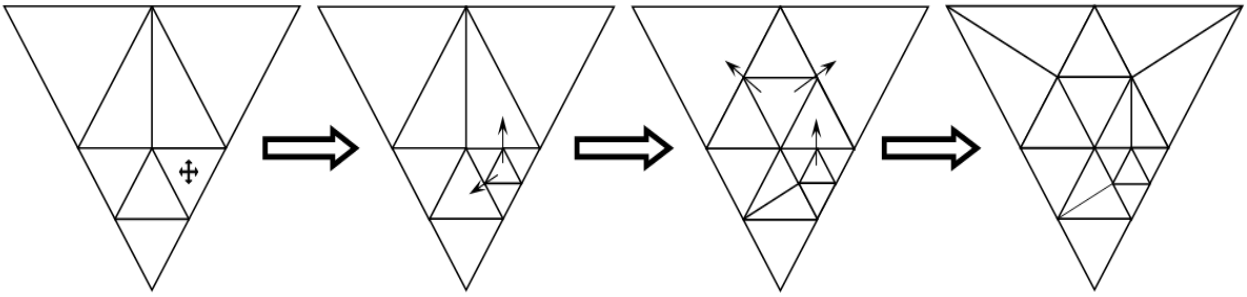


Figure 5.4 Shortest side neighbor of a half split cell is marked for further refinement.

If two neighbors of a non half divided cell (central cell in Figure 5.5) are marked for refinement, then that cell is marked for fully divided into four cells and modifications continue by the logic described earlier.

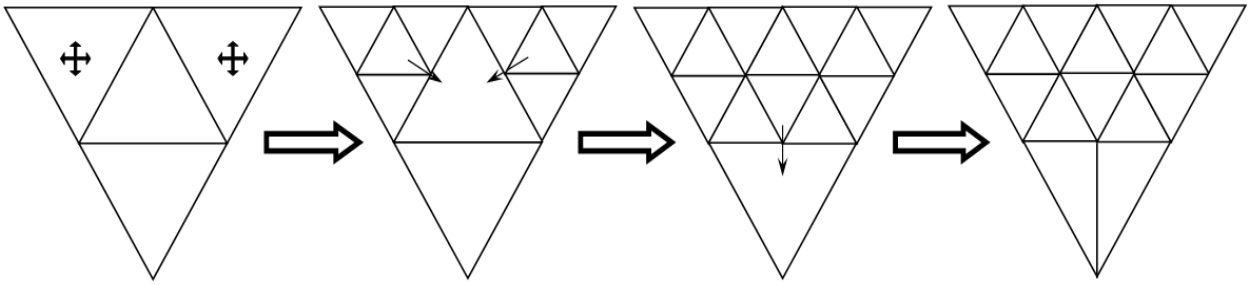


Figure 5.5 Two neighbors of a non half split cell are marked for refinement

Some refinement operations on cells used in the code development are listed below.

Full split

A triangular cell is locally divided by first bisecting each of the sides of the triangles by adding nodes as shown. These nodes are then joined to obtain four smaller triangles. The local mesh size (δ) is reduced to approximately half of the original size.

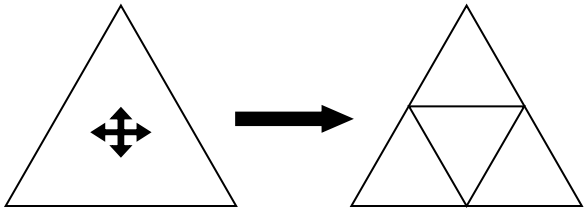


Figure 5.6 Full split cell refinement operation

Half split

When a cell is marked for refinement and its neighbor is not (Figure 5.7a), then this may cause a problem. In such situations, the upper cell is marked for half refinement (Figure 5.7b).. The cell is said to be half split from that particular side (Figure 5.7c). All the cells on the boundary of the adapted zone generally have such configuration. .

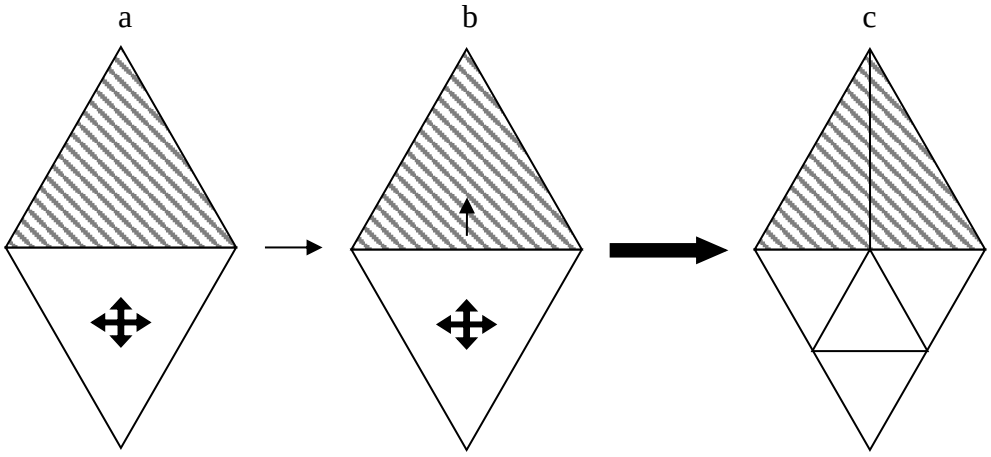


Figure 5.7 Half split cell refinement operation

Some terminologies used in the code development are listed below.

Corresponding to the refinement operations, a cell can be either *full marked* or *half marked or unmarked* depending upon the value of the label 'split'.

Full mark (+)

This indicates that the cell is marked for full splitting. Refining operations that come from the input are all *full marked*.

Half mark(→) This means the cell is marked for half splitting. The direction of the arrow indicates the side it is half marked from. It is to be half split along this direction.

For the purpose of clarity, a cell classified as equilateral cell or a half cell.

Equilateral cell(star marked)

All cells of the initial mess are by default labeled 'equilateral cells'. The only way new equilateral cells can be formed is when an existing equilateral cell is full split (i.e sub-divided into four new cells) (Figure 5.8c).The resulting four cells are called equilateral cells. A equilateral cell can be either full marked or half marked (Figure 5.8b). Depending on the mark, it can be further sub-divided in four new cells by performing a full split or two new cells by performing a half split. They will be exactly half the size of the parent triangle.

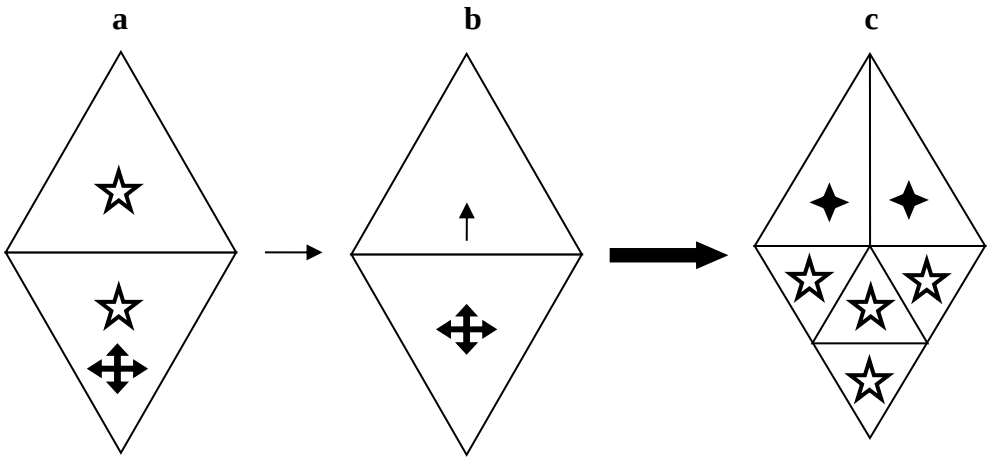


Figure 5.8 Illustrating equilateral cells and half cells

Half cell (◊ marked)

Initial mess has no half cells. The only way new half cells can be formed is when an existing equilateral cell is half split (i.e. sub-divided into two new cells). The resulting two cells are called 'half cells'(refer Figure 5.8b and Figure 5.8c). Half cells generally have an oblong shape adding non-orthogonality to the mesh. If we attempt to split a half cell, we end up getting obtuse triangles further adding non-orthogonality to the mesh. Hence half cells are not be further sub-divided using any splitting operations.

Like a equilateral cell, a half cell can also be either full marked or half marked. In either case, it neither full split nor half split. A new splitting operation called as 'half cell splitting' is performed on it is.

Half cell split (refinement operation)

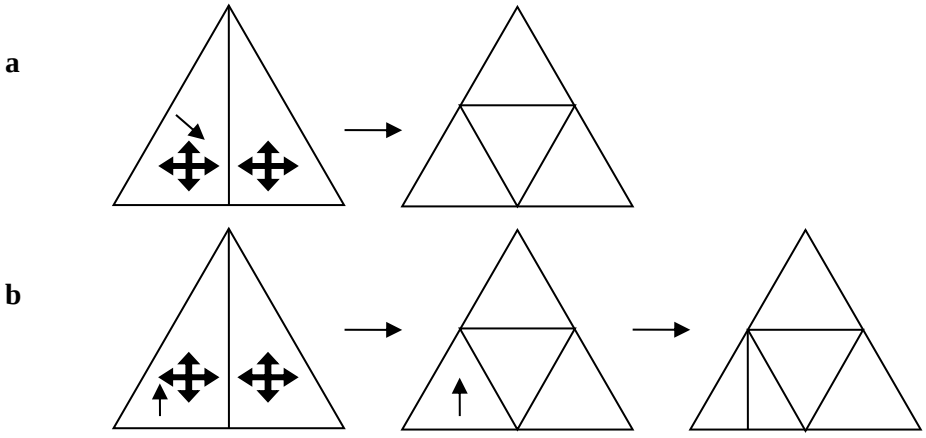


Figure 5.9 Half cell refinement operation

This refinement operation(Figure 5.9a) is performed on a pair of right half cell and left half cell together if they have a non zero split value. The line joining the right half cell and the left half cell is deleted. The equilateral cell left is full split.

If a half cell is split marked from the shortest side (Figure 5.9b), then it is given a new label called ‘double modified’ (dM) which is only used during the its splitting operation. This label is only given to half cells. Because of this label, the new equilateral cell formed from such cell is further half split from that face.

All the various terminologies and splitting operations used in code development have been discussed. A systematic algorithmic procedure for refinement is constructed. The refinement procedure is done in three stages.

Stage 1. Split propagation algorithm.

Stage 2. Face Midpoints creation.

Stage 3. Cell modification

Only the first stage and the second are described here. The last stage is common for refinement and coarsening and will be dealt later.

Stage 1. Split Propagation algorithm

In this stage, cells are only marked for splitting and no other modification is done. If a cell has already been split marked, some of its neighbors might also be split marked. This split marking of the neighbors is called as Split propagation.

For equilateral cells

1) For equilateral cells that are full marked (+), all its neighbors are split marked(Figure 5.10)

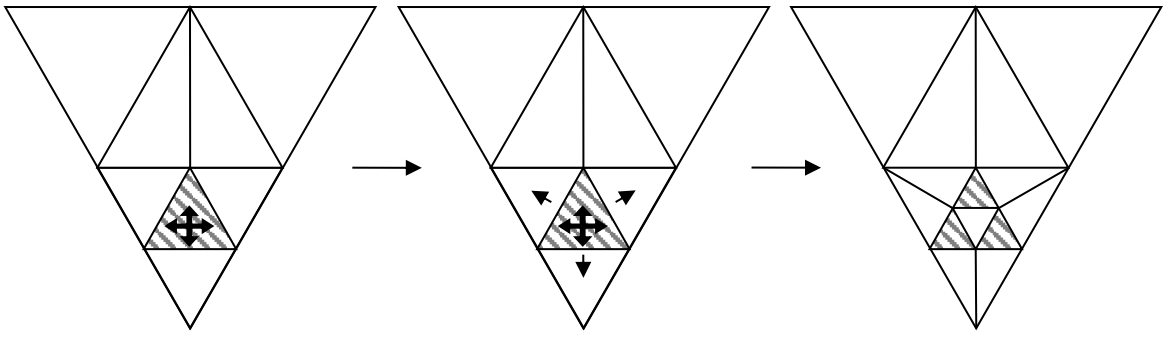


Figure 5.10 Split marking by an equilateral cell.

2) For equilateral cells that are half marked (→), no split propagation is done. Thus the split propagation stops at such cells(Figure 5.11).

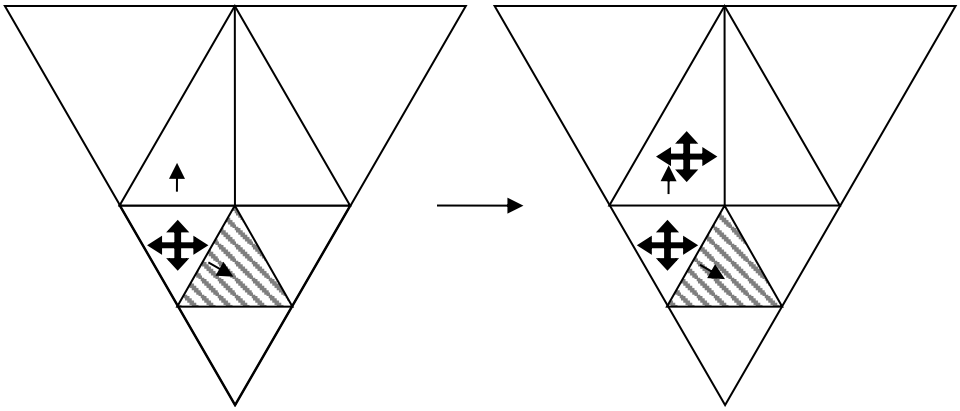


Figure 5.11 Equilateral cell being half marked.

3) If an unmarked equilateral cell can be half marked (→), if cell from only one side wants to split mark (refer Figure 5.11).

4) If an unmarked equilateral cell can be full marked (+) if cells at least two sides want to split mark it. (refer Figure 5.12)

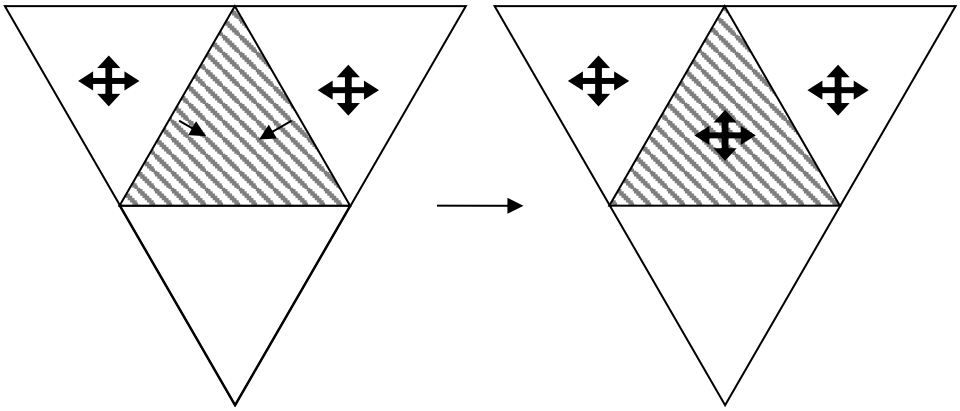


Figure 5.12 Equilateral cell split marked from two sides

For half cells

5) For half cells, that are split marked (non zero split value), all its neighbors except the cell across the shortest side, are split marked (Figure 5.13).

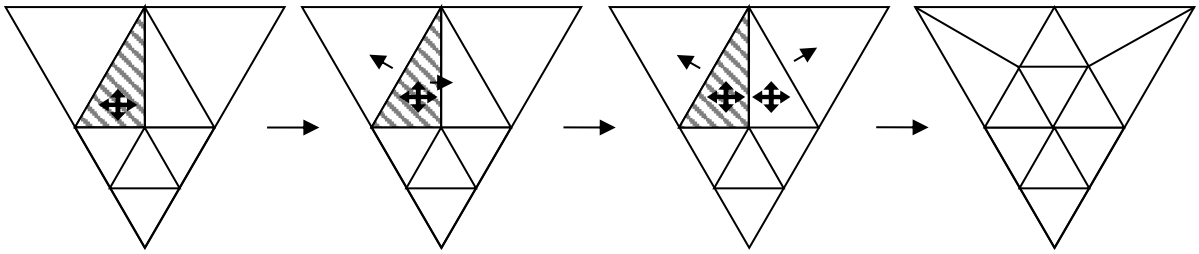


Figure 5.13 Split propagation of a split marked half cell.

6) If a half cell is half marked from any side, it is modified and treated as a full marked cell (Figure 5.14).

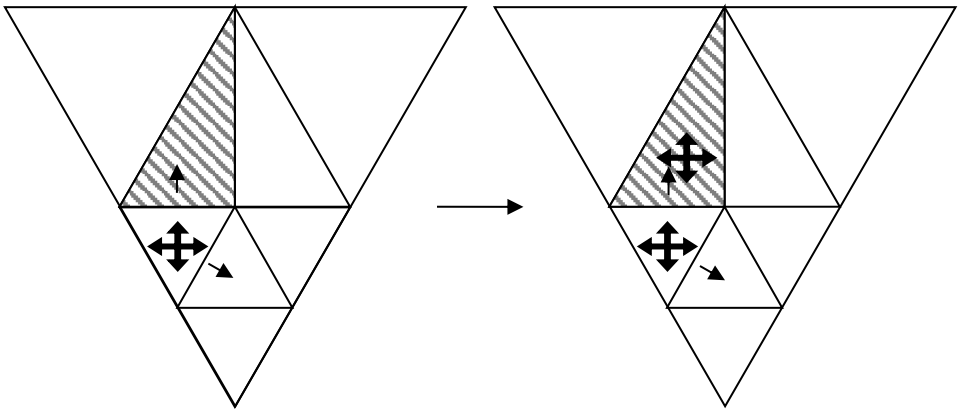


Figure 5.14 Half mark is unconditionally modified to full mark for a half cell

7) If a half cell is split marked from shortest side, it is given a tag ‘double modified’ dM (Figure 5.15). Denoted by an arrow from the shortest side.

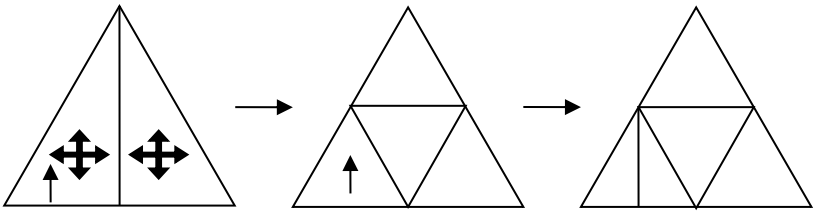


Figure 5.15 Half cell split marked from the shortest side

The algorithm continues iteratively until no change is seen. Although iterative, each cell is allowed to split mark another cell only once. All the cells which are to be split are decided in this algorithm and no more changes are done to the split marking.

Stage 2. Face Midpoints creation

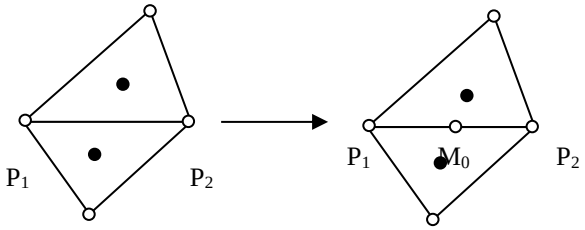


Figure 5.16 Face midpoint creation

Till this stage, no cell modification has been done, so cell variable values of neighboring cells are available. Depending on the cells marking (half mark, full mark ,dM mark) and type of cell (half cell or equilateral cell), allocation of mid points indices for the faces to the deserving sides of the cells is done. The algorithm here creates a midpoint of the face(Figure 5.16), it stores its indices in the value of the appropriate label 'midpt[0 ,1, 2]. For eg, if the midpt is on face F_0 it stores it in label 'midpt[0]' of the cell. The algorithm also stores the parent indices in the data structure of the newly created point. If the neighbor across the face has already created the midpoint for the face, the cell grabs its point index and stores its indices in the value of the appropriate label 'midpt[0,1, 2] in the cell's data structure, similar to the previous step. After allocating midpoint's to the deserving sides, the cell is marked 'midpoint allocated', to inform other cells about the availability of new mid point indices.

5.2 Cell Coarsening.

In coarsening, the cell marked for coarsening are collapsed into their original parent cells. If a given cell is marked for coarsening, some of its sides and points have to be removed. Consider any other cell that has some of its sides or points in common with the given cell. If that cell is not marked for coarsening than the coarsening of the given cell is also obstructed. Such cells cannot be coarsened.
Thus there is need for some algorithm that can come up with a coarsen marking that all cells agree upon. Some terminologies used in the code development are listed below.

Coarse mark for point →A coarse mark on a point is represented by a black dot on it. Such points are potential candidates for removal.

Coarse mark for Cell (Star mark)→ A coarse mark on a cell is represented by a black star on it. These are cells that are marked for coarsening. Such cells are potential candidates for removal.

If a cell is marked for coarsening, coarsening operations are performed on cells. There are three coarsening operations used in the code development which are are listed below.

1) Full collapse

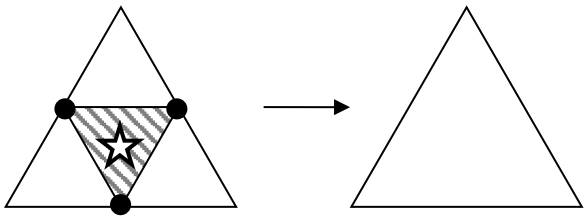


Figure 5.17 Full collapse coarsening operation

This operation is the inverse of the full split operation in refinement. In this operation all the sides and points of the cell are deleted. The neighbors of this cell and the cell itself are all fully collapsed into their original parent cell from which they were originally created. This operation can only happen under the following conditions

1. The cell should be an equilateral cell.
2. The cell has to be coarse marked.
3. All its three of its points must marked for coarsening.
4. Any two of its points must have one common parent point.

Condition 4 ensures that the function acts only on **central cells** as shown by dotted line in the figure. No other type of cells can satisfy this condition.

2) Half collapse

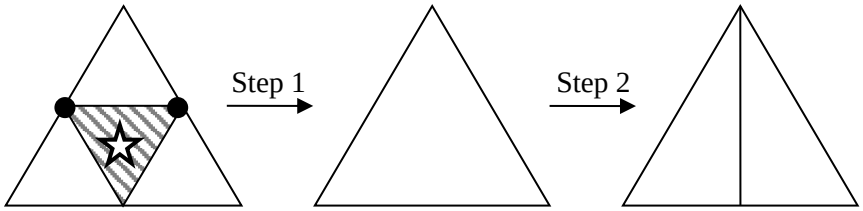


Figure 5.18 Half collapse coarsening operation

This operation is the inverse of the half cell split operation. All the sides and two of the points of the cell are deleted (step 1). The new cell then is half split by a line drawn from the unmarked point to point opposite to it (step 2).Thus four cells are collapsed into two cells.

The conditions for operation to modify a cell are identical to those for full collapse, except condition three which is modified. Instead of three points, two points of the cell must be marked for coarsening.

3) Half cell collapse

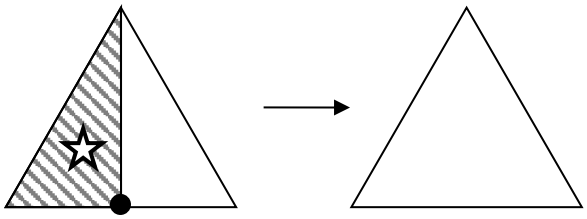


Figure 5.19 Half cell collapse coarsening operation

This operation is the inverse of the half split operation. In this operation, a pair of right and left half cells is collapsed into their original parent cell. The side joining them is deleted. The point of the deleted side that does not belong to the parent cell is also deleted.

This operation can only happen under the following conditions

1. The cell should be a left half cell.
2. The point opposite the longest side must be marked for coarsening.

Like refining coarsening is also divided into three stages.

Stage 1. Coarse marking of Points.

Stage 2. Cells and points filtering.

Stage 3. Cell modification (common for refinement and coarsening).

Stage 1. Coarse marking of Points

Points that are candidates for coarsening are identified (Figure 5.20). A point is marked for coarsening if and only if all the cells that are attached to this point are marked for coarsening. A point that is not a created point is unmarked. This stage is non-iterative.

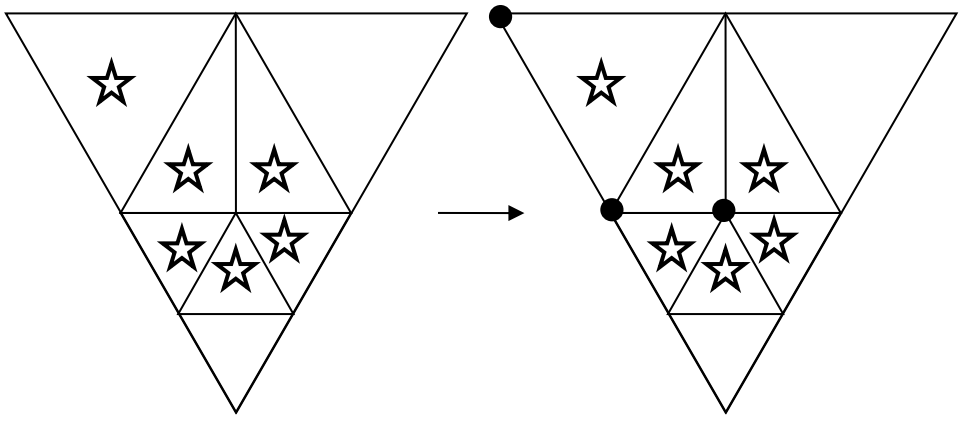


Figure 5.20 Coarse marking of Points

Stage 2. Cells and points filtering

In this stage, the cells and points that are obstructed from coarsening are filtered out of the coarse marking process. The point filtering obstructs some of the points from being released by unmarking them. This affects the coarsening of the some other cells that are ready to release this point. They in turn unmark their point and the cycle goes on until all cells that are to be coarsened agree on a common coarse marking of points. Also some of the coarsening operations also restricts the cells that can and cannot be marked. This makes the stage iterative.

Cell Filtering

If a given cell is marked, it can be filtered out if

1. The cell is not a central equilateral cell or a left half cell (restricted by coarsening operation's input).(refer Figure 5.21)

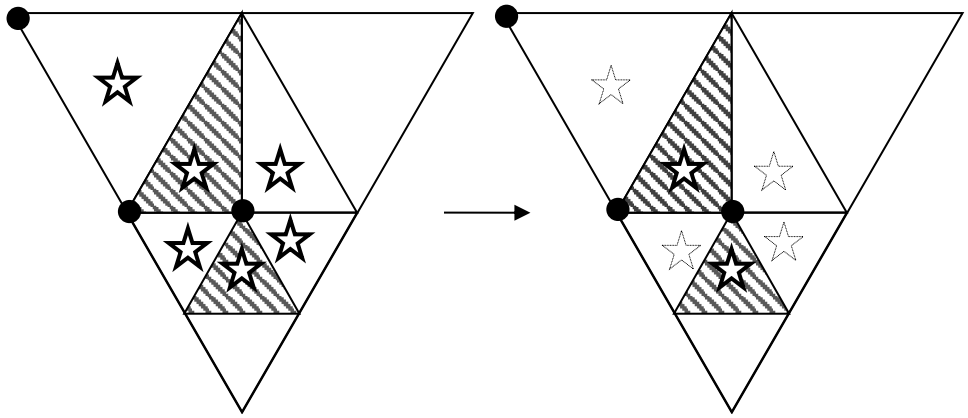


Figure 5.21 Cell filtering of non left half and non-central equilateral cells

2. It is a equilateral cell and only one marked point remains. In this case, the remaining point is also unmarked.(Since at lest two coarse marked points are needed for the cell to collapse)Figure 5.22

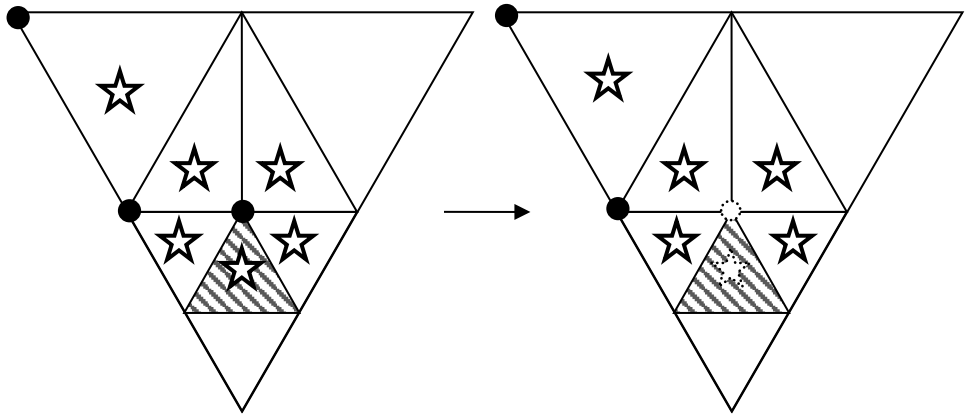


Figure 5.22 Cell filtering of equilateral cells

3. It is left half cell and has its point opposite the longest side (refer Figure 5.23) unmarked.(Since the only point that can be removed in a half cell is itself not there)

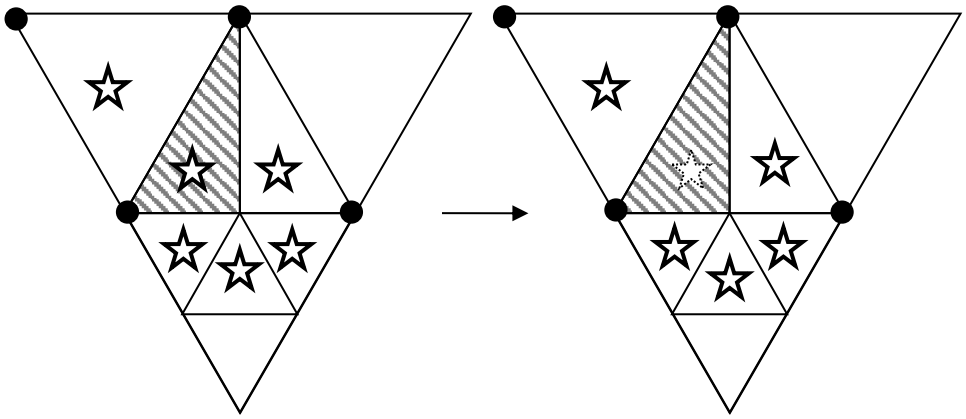


Figure 5.23 Cell filtering of half cells

Points filtering

Points are of a marked cell are unmarked, if there is no possibility for their deletion.

1. For a half cell all points except points opposite the to longest side are marked (refer Figure 5.24)

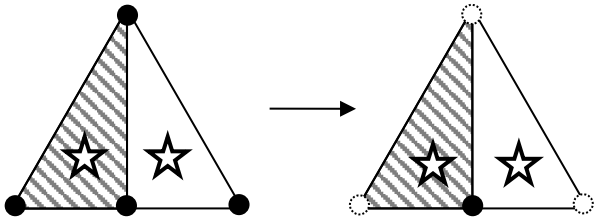


Figure 5.24 Unmarking of points of a half cell

2. For a equilateral cell, all points not belonging to central equilateral cells are unmarked. (refer Figure 5.25).

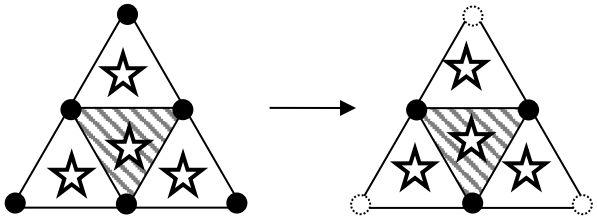


Figure 5.25 Unmarking of points not belonging to the central equilateral cells

3. If only one marked point is left on central equilateral cell, than it is also unmarked (Figure 5.26).

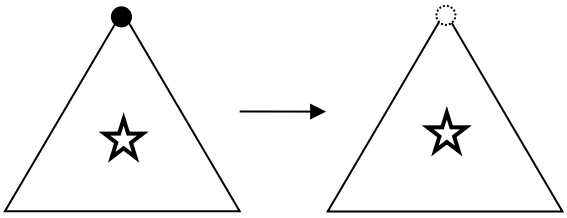


Figure 5.26 unmarking of points of an equilateral cell

Stage 3:Cell modification

This is the Final stage common for refinement and coarsening. In this stage the cell is redirected to appropriate refining and coarsening operations described earlier. This redirection depends upon split or coarse values and type value of cells. Each of these operations blindly modifies the cell in a specific way as they are meant to do. At this stage we lose access to neighbors as neighboring cell's numbers and points may have been changed.

5.3 Redistribution of solution variables

When ever a refinement or coarsening is done on a cell, we have to find field value at a newly created point where there was no value available. To do this we have to resort to some kind of interpolation. Any interpolation always has some errors. These errors increase if local curvatures and mesh size are high.

5.3.1 Nodal interpolation

Values at cell centers like C_i are known. We have to find value at nodes like N that are surrounded by cells.

$$T_N = \frac{\sum w_{C_i} T_{C_i}}{\sum w_{C_i}} \; where \; w_{C_i} = \frac{1}{d_{C_i}} \qquad 5.1$$

This is done by distance averaging out the surrounding points. refer Figure 5.27

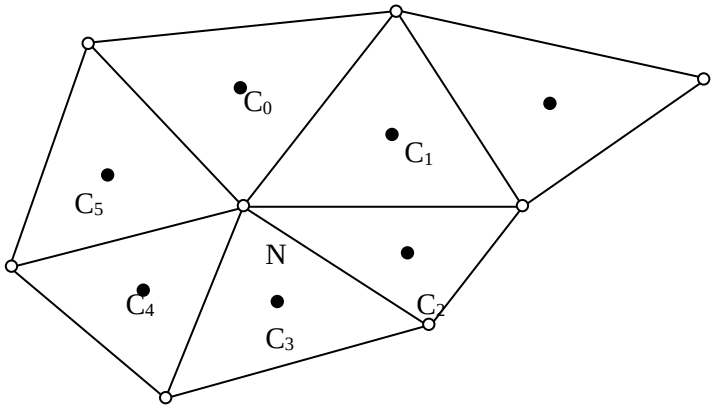


Figure 5.27 Nodal interpolation

5.3.2 Face mid point creation

We take the general case of splitting a face. Linear interpolation is performed. (refer Figure 5.28)

$$T_{M_0} = \frac{T_{P_1} + T_{P_2}}{2} \qquad 5.2$$

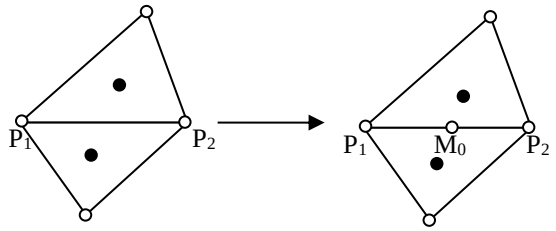


Figure 5.28 Face mid point creation

The interpolations listed below consist of cell centers values of new cells created by the refining and coarsening operations. Although refining operations are shown below, interpolations of inverse operations are give in the same place. We know value at all mid point's indices (M_0, M_1, M_2) through eqn 5.2.

5.3.3 Full split

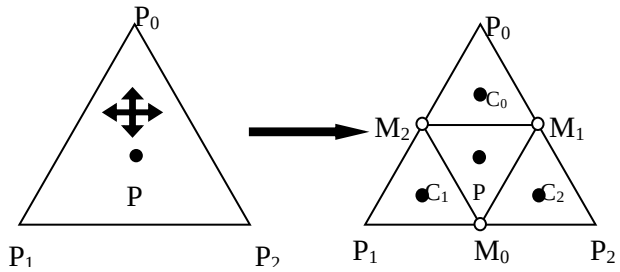


Figure 5.29 Full split

We have to find values at cell centres C_i. To do this (refer Figure 5.29), we interpolate between the original cell centre and the node across the original cell center. The equation is given below.

$$T_{C_i} = \frac{T_P + T_{P_i}}{2} \qquad 5.3$$

5.3.4 Half split

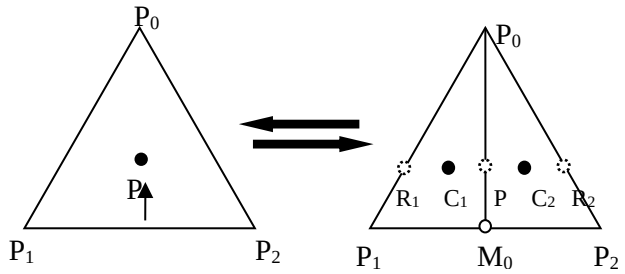


Figure 5.30 Half Split

Since centroid divides the median in the ratio 2:3. Interpolation is performed between point P_0 and point R_1,R2.(refer Figure 5.30)

$$T_{R_1} = \frac{2T_{P_1} + T_{P_0}}{3} \; and \; T_{R_2} = \frac{2T_{P_2} + T_{P_0}}{3} \quad 5.4$$

Interpolation is performed between point P and point R_1,R2

$$T_{C_1} = \frac{T_P + T_{R_1}}{2} \; and \; T_{C_2} = \frac{T_P + T_{R_2}}{2} \quad 5.5$$

For inverse operation *half cell collapse,* interpolation is performed between points C_1 an C_2 to get P

$$T_P = \frac{T_{C_1} + T_{C_2}}{2} \quad 5.6$$

5.3.5 Boundary split

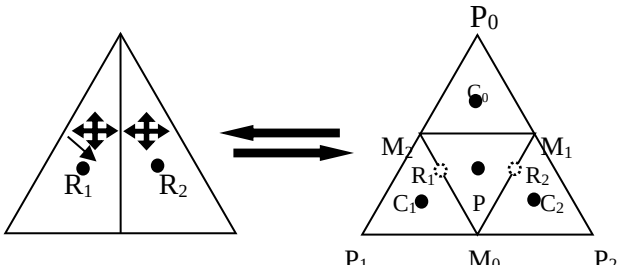


Figure 5.31 Boundary split

Interpolation is performed between point $R_1$and point R_2 to get P. (refer Figure 5.31).

$$T_P = \frac{T_{R_1} + T_{R_2}}{2} \quad 5.7$$

Interpolation is performed between point P and point P_0,P_1,P_2 to get C_0,C_1,C_2

$$T_{C_1} = \frac{T_P + T_{P_1}}{2}, T_{C_2} = \frac{T_P + T_{P_2}}{2}, T_{C_0} = \frac{T_P + T_{P_0}}{2} \quad 5.8$$

For inverse operation, Interpolation is performed between point C_0 and point C_1,C_2 to get R_1,R_2

$$T_{R_1} = \frac{2T_{C_1} + T_{C_0}}{3} \; and \; T_{R_2} = \frac{2T_{C_2} + T_{C_0}}{3} \quad 5.9$$

Chapter 6 Results

6.1 General Procedure

The general procedure to solve the diffusion equation with adaptive grid refinement methodology is give below.

- Generate the Mesh (Appendix 1).
- Put the initial condition on the grid.
- Start the time step loop.
  1. Start the Grid refinement loop.
     a. Find grid refinement indicator (E_δ) based on current time step mark for refinement or coarsening. According the condition below. (eqn 4.31).For the present work x is taken as 2%.

$$(cosrsen)1\% \times \frac{x}{2} < \frac{E_\delta}{T_{ref}} < 5\% \times \frac{x}{2}(Refine)$$

     b. Use the grid refinement methodology to modify the grid (Chapter 5).
     c. Redistribute the solution variable T_P^n over the new grid (section 5.3).
     d. Update the triangles and neighbors of triangles, to update the data structures existing triangles as well as new triangles.

     Continue loop If step b has made changes to the grid.

  2. Solve the Discretized equation(eqn 3.10)

$$\frac{T_P^{n+1} - T_P^n}{\Delta t} = \frac{\alpha}{\Delta V}\sum\left(C_\xi \frac{T_F^n - T_P^n}{\Delta \xi} + C_\eta \frac{T_2^n - T_1^n}{\Delta \eta}\right) + S$$

  3. Find node Temperature T_{Node}^{n+1} using equation 5.1

$$T_N = \frac{\sum w_{C_i} T_{C_i}}{\sum w_{C_i}} \; where \; w_{C_i} = \frac{1}{d_{C_i}}$$

  4. Implement boundary conditions on T_P^{n+1}using eqn 3.21.
  5. Find Time step Δt_{max} (eqn 4.32) such that

$$\frac{\alpha \Delta t}{\Delta V_P} \leq 0.2 \;\; \& \frac{\left|T_P^{n+1} - T_P^n\right|}{T_{ref}} < 1\% \; for \; all \; cells$$

  6. Update temperatures of cell nodes and cell centers.

$$T_P^n = T_P^{n+1} \; \& \; T_{P_i}^n = T_{P_i}^{n+1} \; for \; i = 0 \; to \; 2$$

- If steady state is reached or required observation time finishes, Exit loop. Else go to step 1.

6.2 Test Problem 1(moving Gaussian)

The domain is square of length 1, the boundary condition is zero. The initial condition is a Gaussian in x direction and a sine curve in y direction. The source term is derived below.

The original parabolic differential equation is

$$\frac{\delta T}{\delta t} = \alpha \nabla^2 T + S \tag{6.1}$$

$$\text{Let } T = E(x,t)X(x)Y(y) \tag{6.2}$$

$$where\ X(x) = x(L-x)\ , Y(y) = C_o sin\left(\frac{\pi y}{L}\right) \tag{6.3}$$

$E(x,t) = e^{-\left(\frac{x-x_p(t)}{s}\right)^2}$ is the Gaussian distribution Function at x_p (6.4)

$$where\ x_p(t) = A_G + B_G sin\left(\frac{2\pi t}{\tau_G}\right) \tag{6.5}$$

$$A_G = 0.5L\ , B_G = 0.25L\ , \tau_G = 0.1sec$$

Substituting eqn 6.1 in solution in eqn 6.2.We get the expression for source term

$$S(x,y,t) = T_t - \alpha(T_{xx} + T_{yy}) \tag{6.6}$$

$$T_t = (ET)_t XY = XY(ET' - TE_x x_P') \tag{6.7}$$

$$T_{xx} = (EX)_{xx} YT = \{E_{xx}X + EX" + 2E_x X'\}Y \tag{6.8}$$

$$T_{yy} = EXY" \tag{6.9}$$

Use eqn below to get 6.7,6.8,6.9 . Then substitute them in 6.6 to get source term S(x.y,t).

$$E_x(x,t) = -\frac{2}{s^2}(x - x_p)E \tag{6.10}$$

$$E_{xx}(x,t) = -\frac{2}{s^2}\left(E + (x - x_p)E_x\right) \tag{6.11}$$

$$x_P'(t) = B_G cos\left(\frac{2\pi t}{\tau_G}\right)\frac{2\pi}{\tau_G} \tag{6.12}$$

The constant C_0 is such that T_{max} over the domain is 100.

Started the numerical procedure with an initial coarse triangular mesh.

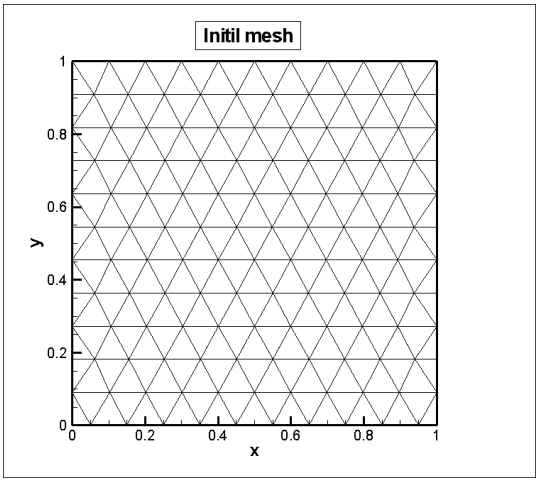


Figure 6.1 Initial mesh

Also plotted maximum error (e_{max}) as a function of t/τ_g.

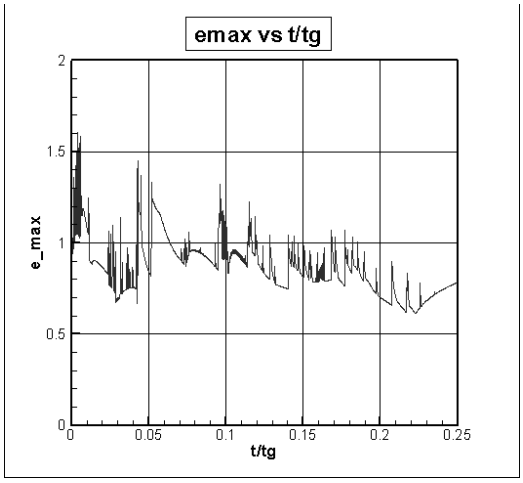
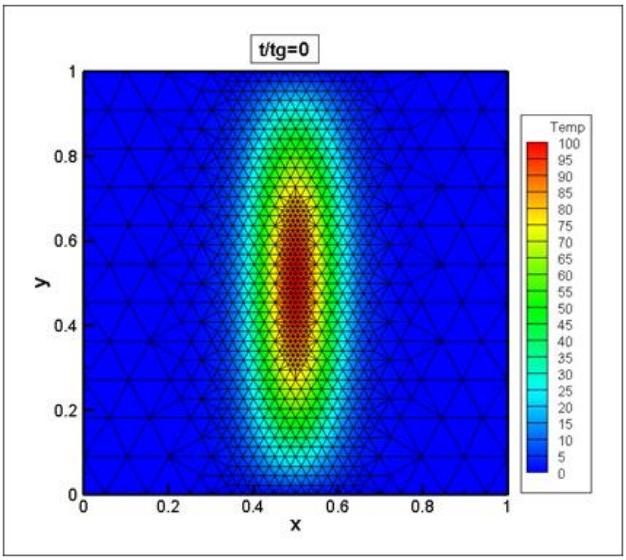
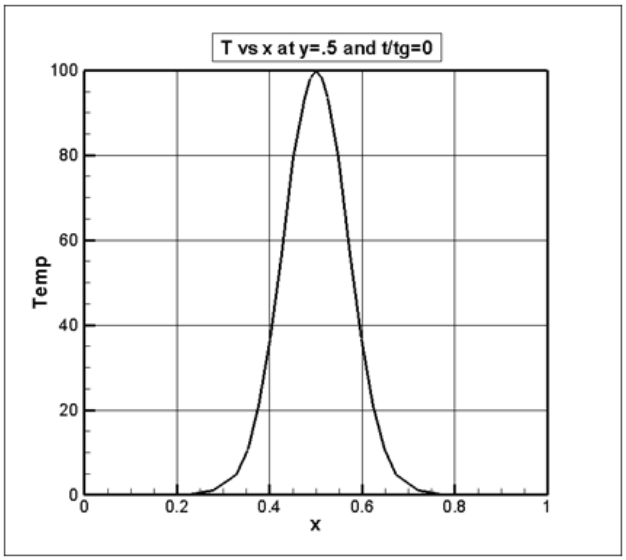


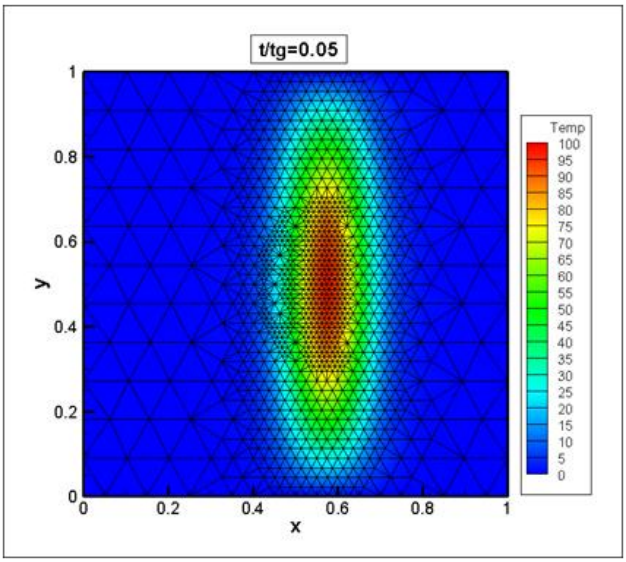
Figure 6.2 Maximum error vs t/t_g

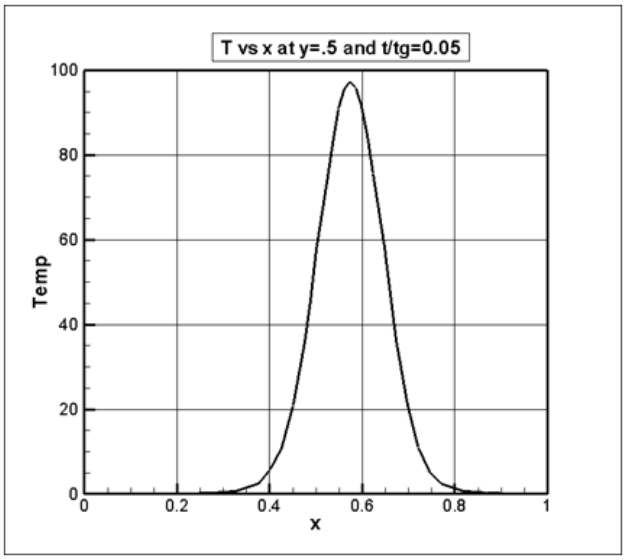
The solution is a Gaussian moving in the x direction. Contour Plots T(x,y) at different t/τ_g for a quarter of the time cycle is given below. To the right is a cross sectional view of solution at y=0.5

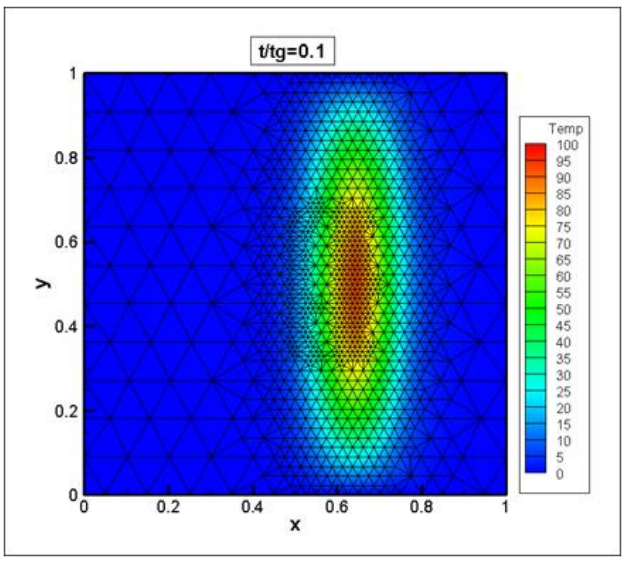
Figure 6.3 Contour plots of T(x,y) at different t/tg for square mesh

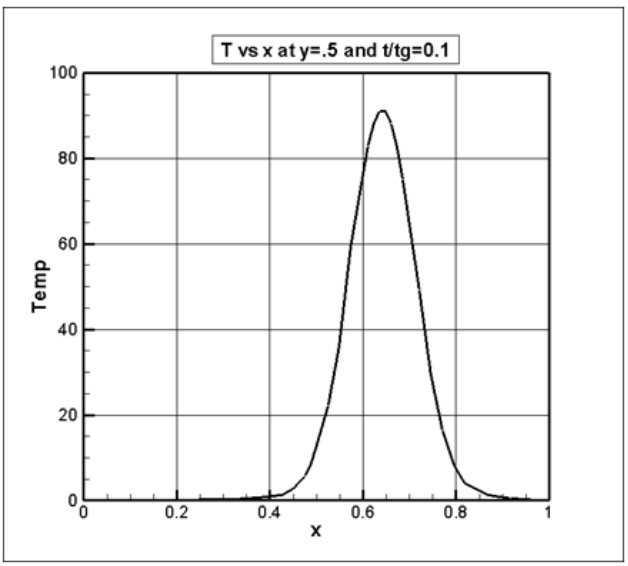

t/tg=0
Temp
100
95
90
85
80
75
70
65
60
55
50
45
40
35
30
25
20
15
10
5
0
y
x
0
0.2
0.4
0.6
0.8
1

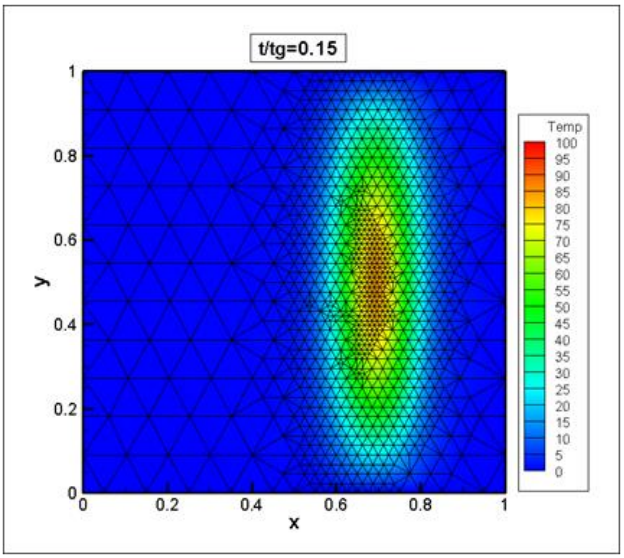

T vs x at y=.5 and t/tg=0
Temp
x
0
20
40
60
80
100
0.2
0.4
0.6
0.8
1

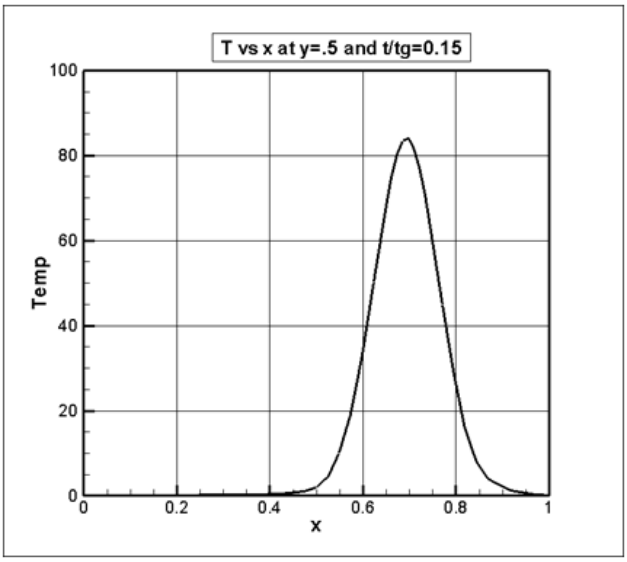

t/tg=0.05
Temp
100
95
90
85
80
75
70
65
60
55
50
45
40
35
30
25
20
15
10
5
0
y
x
0
0.2
0.4
0.6
0.8
1

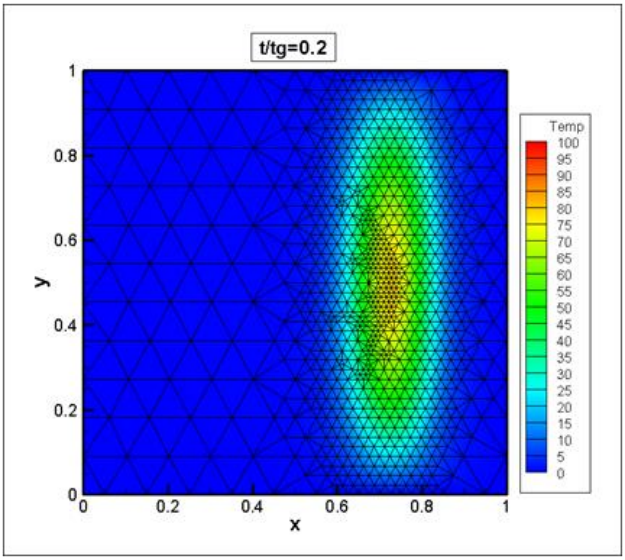

T vs x at y=.5 and t/tg=0.05
Temp
x
0
20
40
60
80
100
0.2
0.4
0.6
0.8
1

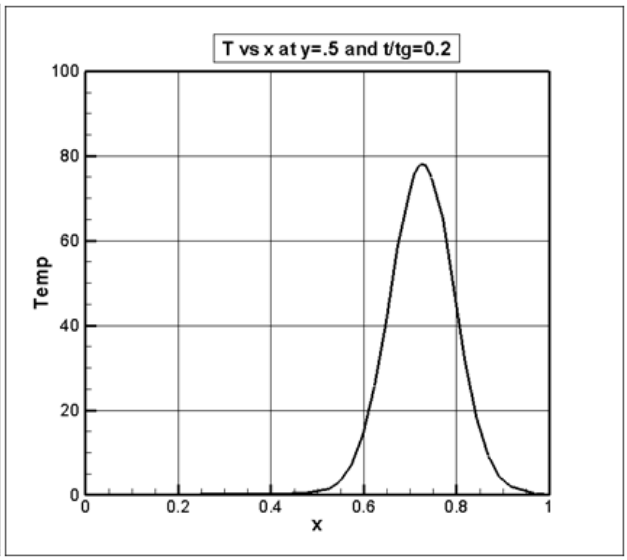

t/tg=0.1
Temp
100
95
90
85
80
75
70
65
60
55
50
45
40
35
30
25
20
15
10
5
0
y
x
0
0.2
0.4
0.6
0.8
1

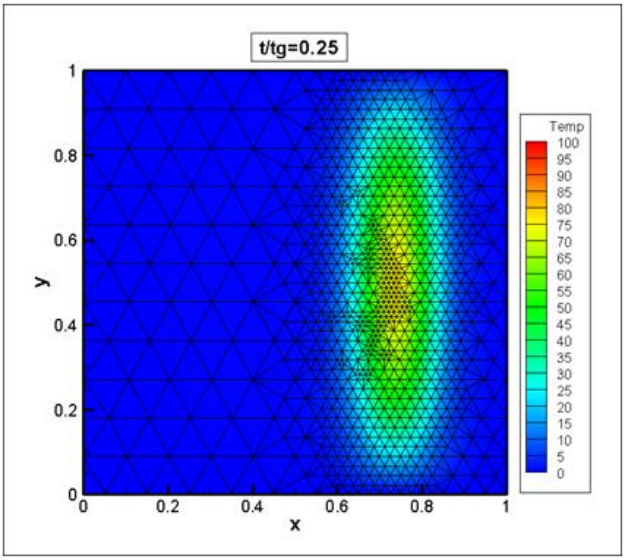

T vs x at y=.5 and t/tg=0.1
Temp
x
0
20
40
60
80
100
0.2
0.4
0.6
0.8
1

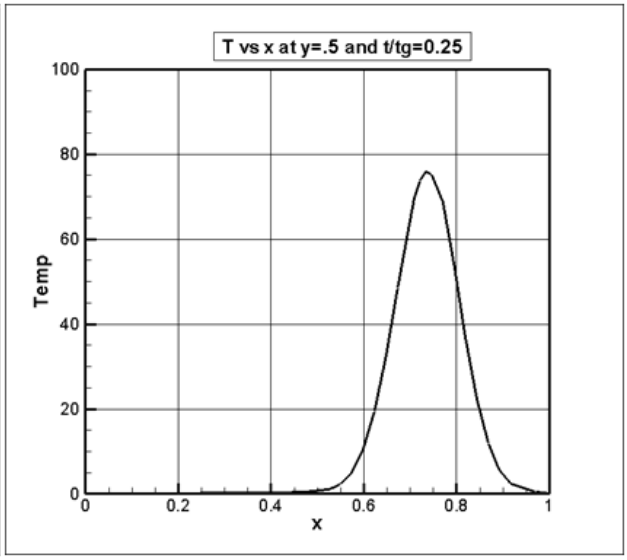

t/tg=0.15
Temp
100
95
90
85
80
75
70
65
60
55
50
45
40
35
30
25
20
15
10
5
0
y
x
0
0.2
0.4
0.6
0.8
1


T vs x at y=.5 and t/tg=0.15
Temp
x
0
20
40
60
80
100
0.2
0.4
0.6
0.8
1


t/tg=0.2
Temp
100
95
90
85
80
75
70
65
60
55
50
45
40
35
30
25
20
15
10
5
0
y
x
0
0.2
0.4
0.6
0.8
1


T vs x at y=.5 and t/tg=0.2
Temp
x
0
20
40
60
80
100
0.2
0.4
0.6
0.8
1


t/tg=0.25
Temp
100
95
90
85
80
75
70
65
60
55
50
45
40
35
30
25
20
15
10
5
0
y
x
0
0.2
0.4
0.6
0.8
1


T vs x at y=.5 and t/tg=0.25
Temp
x
0
20
40
60
80
100
0.2
0.4
0.6
0.8
1

6.3 Test problem 2(outward moving circular Gaussian)

The domain is circular concentric cylinder of radius R_i and R_o, The boundary condition is zero. The initial condition is a circular Gaussian. The source term is derived below.

The original parabolic differential equation is

$$\frac{\delta T}{\delta t} - \alpha \nabla^2 T = S \tag{6.13}$$

Let the solution be $T = E(r,t)R(r)$ 6.14

$$where\ E(r,t) = e^{-\left(\frac{r - r_p(t)}{s}\right)^2} \tag{6.15}$$

$$r_p(t) = A_G + B_G sin\left(\frac{2\pi t}{\tau_G}\right), \tag{6.16}$$

$$A_G = \frac{(R_o + R_1)}{2}, B_G = 0.25(R_o - R_i)$$

$$\tau_G = 0.1sec, R_o = 1, R_i = 0.1, s = .2 \tag{6.17}$$

$$R(r) = C_0 \left(1 - \frac{r}{R_0}\right)\left(\frac{r}{R_i} - 1\right) \tag{6.18}$$

The constant C_0 is such that T_{max} over the domain is 100.

Substituting eqn 6.14 in in eqn 6.13.

$$S(r,t) = T_t - \alpha \nabla^2 T \tag{6.19}$$

The source term and each of its differential is explained below

The laplacian in circular geometry (r,ɵ) is

$$\nabla^2 T = T_{xx} + \mathrm{T}_{yy} = T_{rr} + \frac{T_r}{r} + \frac{T_{\theta\theta}}{r^2} \tag{6.20}$$

Using eqn 6.20 and eqn 6.14 in eqn 6.19. we get the expression for the source term.

$$S(r,t) = T_t - \alpha\left(T_{rr} + \frac{T_r}{r} + \frac{T_{\theta\theta}}{r^2}\right) \tag{6.21}$$

$$Where\ T_t = (E)_t R = R(E_r r_P') \tag{6.22}$$

$$T_r = (ER)_r = (E_r X + ER') \tag{6.23}$$

$$T_{rr} = (ER)_{rr} = (E_{rr}R + ER" + 2E_r R') \tag{6.24}$$

The partial derivatives required are given below

$$E_r(r,t) = -\frac{2}{s^2}(r - r_p)E \tag{6.25}$$

$$E_{rr}(r,t) = -\frac{2}{s^2}\left(E + (r - r_p)E_r\right) \quad 6.26$$

$$r_P'(t) = B_G cos\left(\frac{2\pi t}{\tau_G}\right)\frac{2\pi}{\tau_G} \quad 6.27$$

Started the numerical procedure with an initial coarse triangular concentric mesh.(refer Figure 6.4)

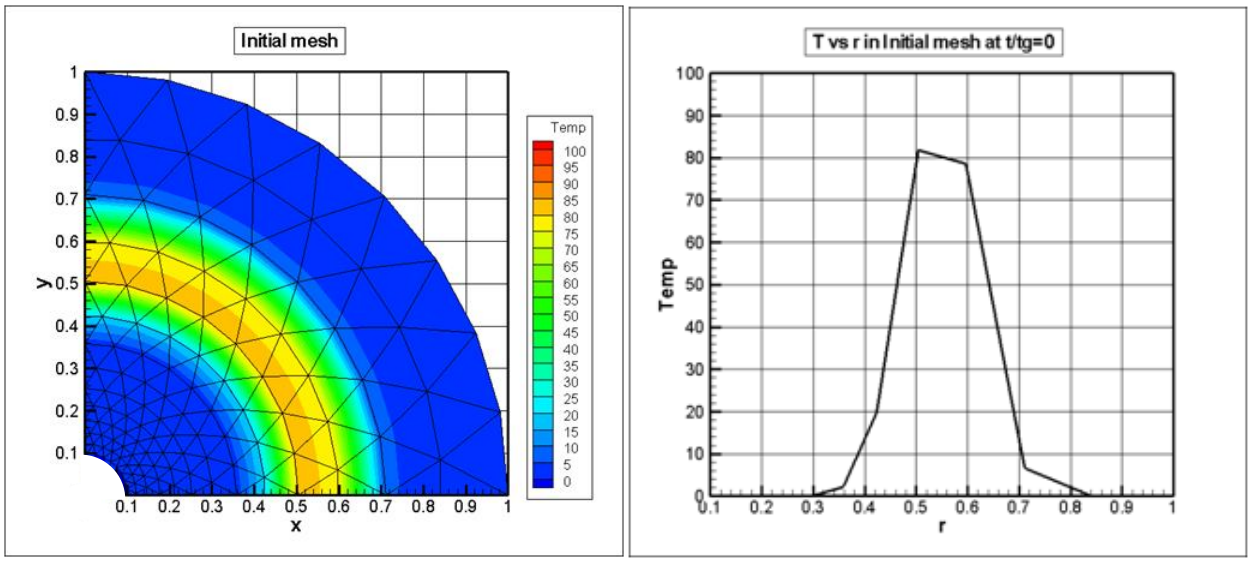


Figure 6.4 Initial mesh and initial solution for circular unrefined mesh

Also plotted maximum error (e_{max}) as a function of t/τ_g.

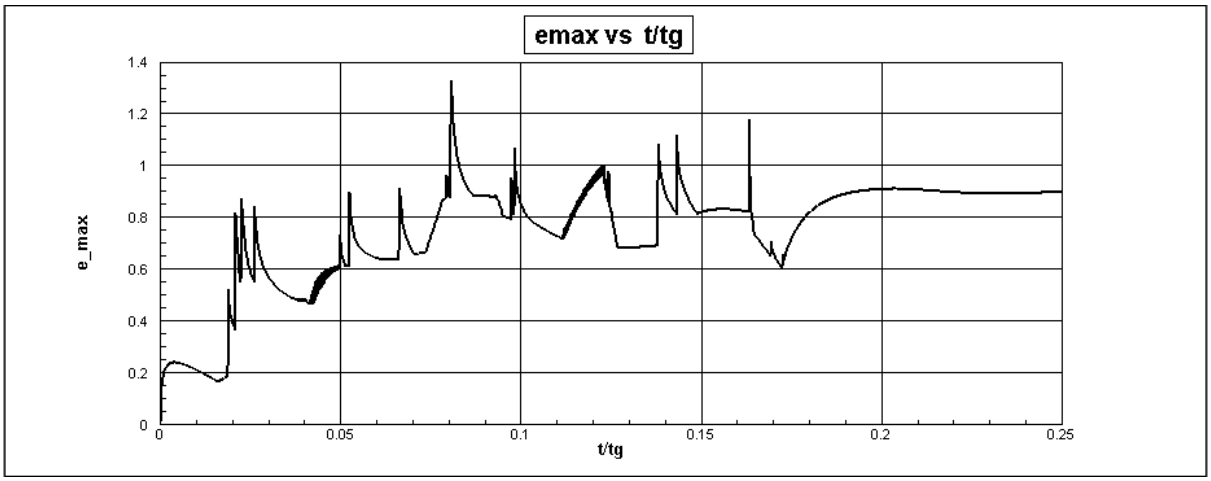
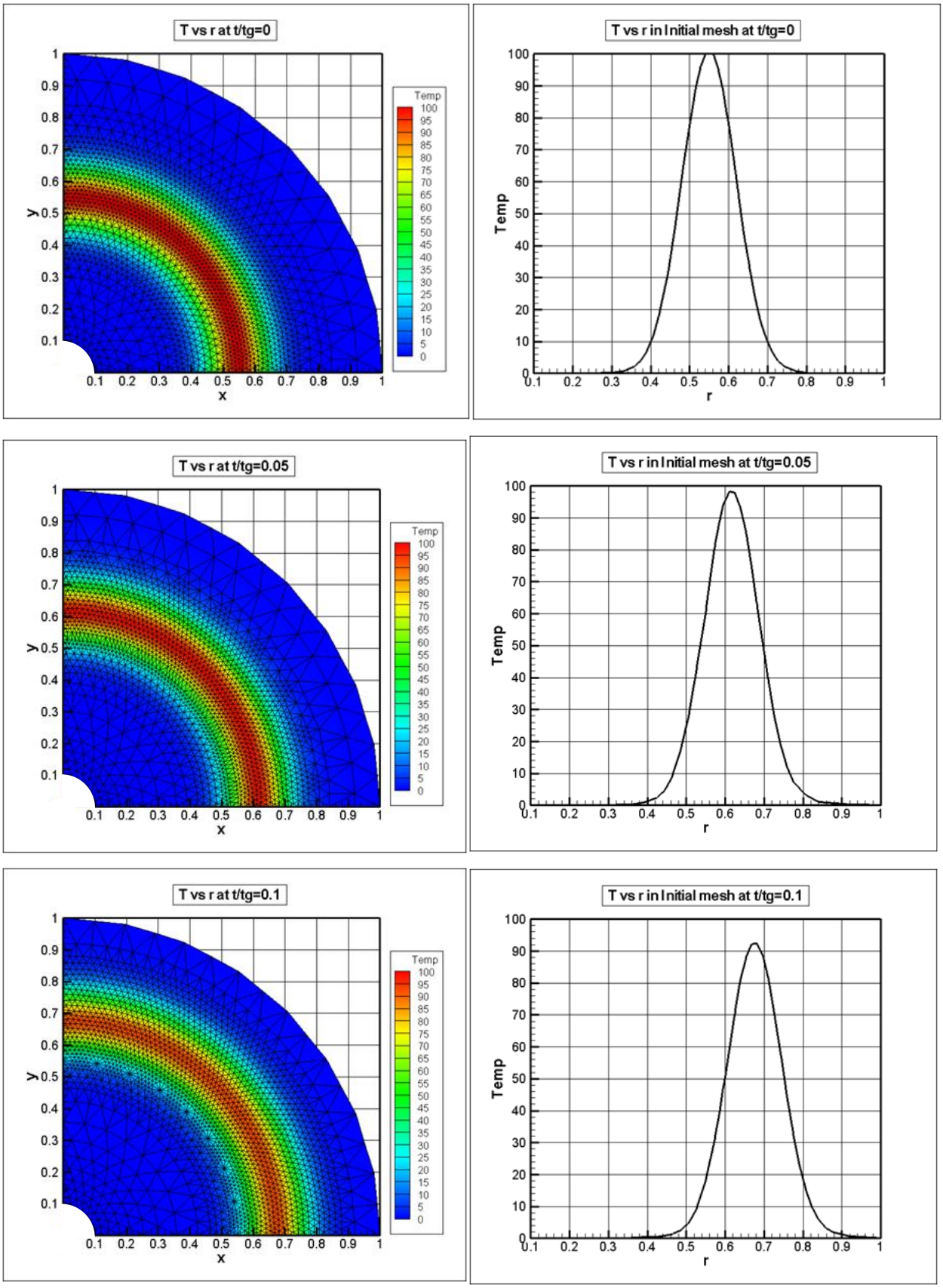
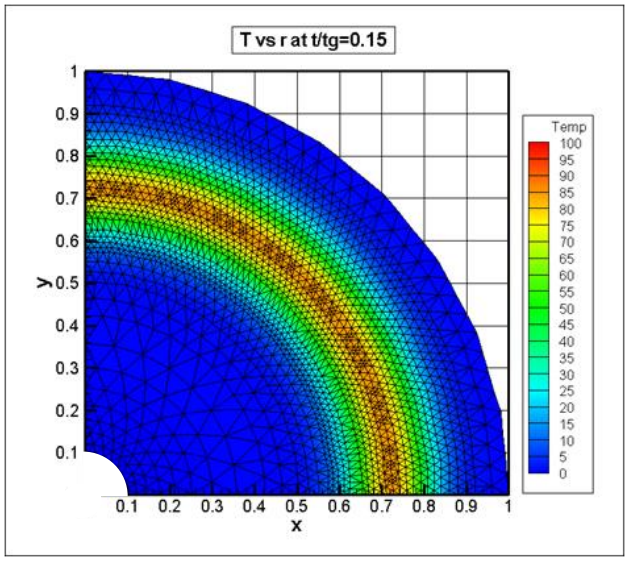


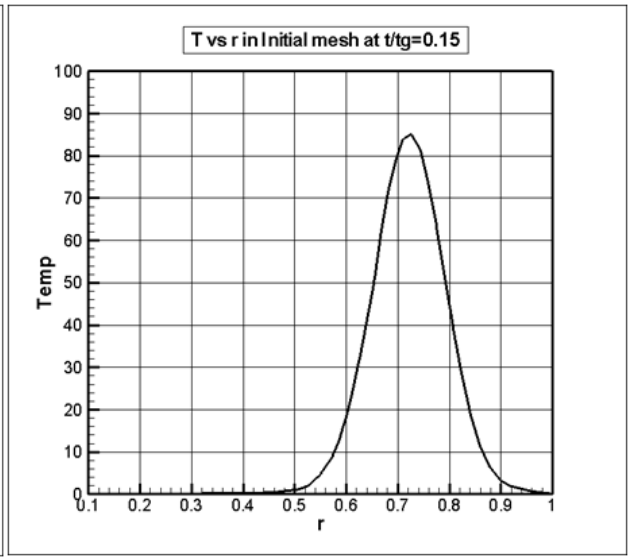
Figure 6.5 Maximum error vs t/tg

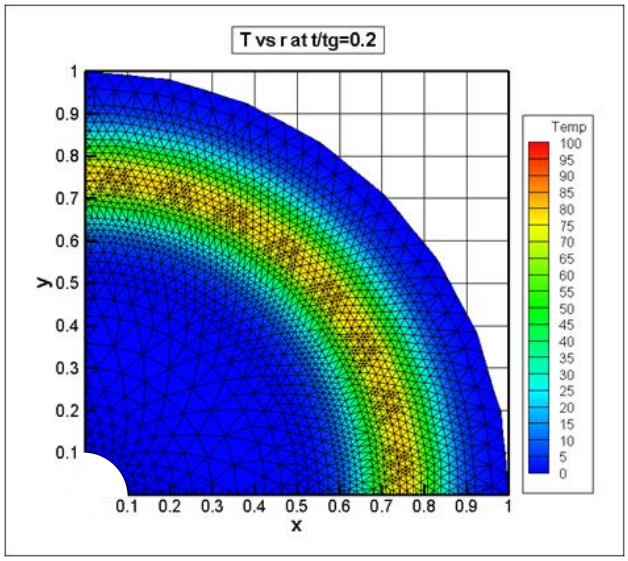
The solution is a Gaussian moving radially outward. Contour Plots T(x,y) at different t/τ_g for a quarter of the time cycle. A quarter of mesh is shown below.

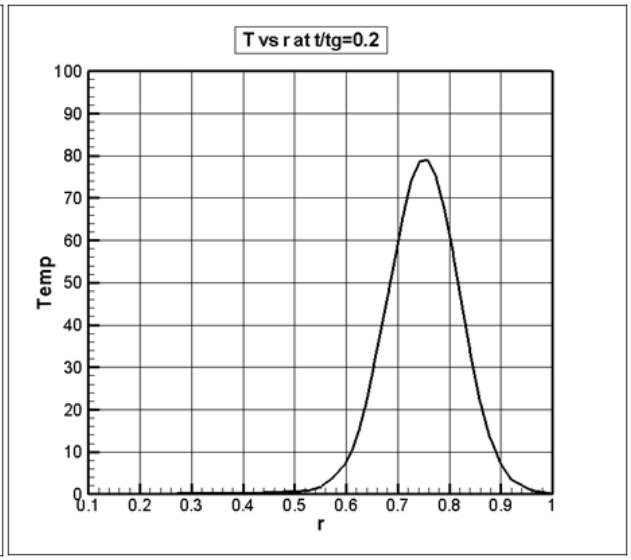
Figure 6.6 Contour plots of T(x,y) at different t/tg for circular mesh

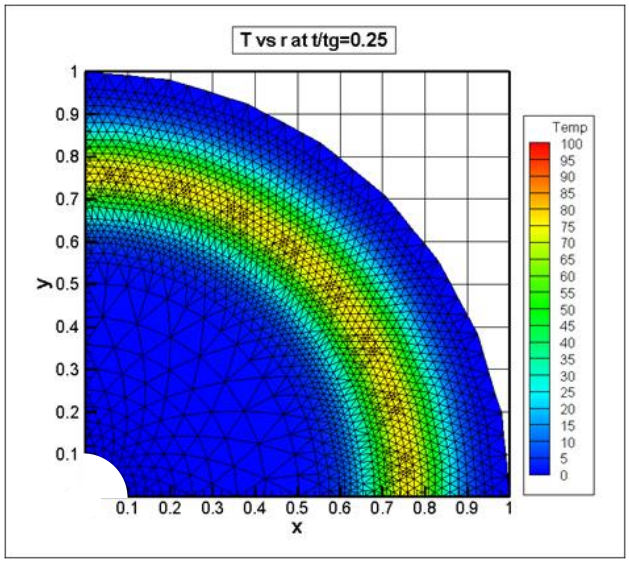


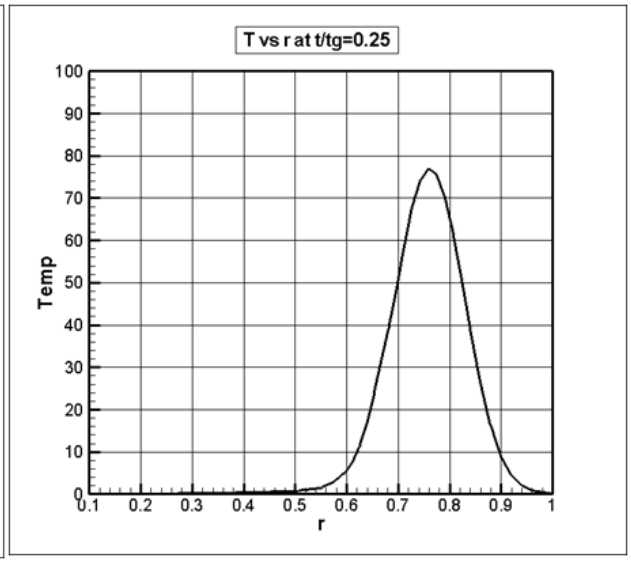

T vs r at t/tg=0.15
Temp
100
95
90
85
80
75
70
65
60
55
50
45
40
35
30
25
20
15
10
5
0
y
x
0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1


T vs r in Initial mesh at t/tg=0.15
Temp
r
0 10 20 30 40 50 60 70 80 90 100
0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1


T vs r at t/tg=0.2
Temp
100
95
90
85
80
75
70
65
60
55
50
45
40
35
30
25
20
15
10
5
0
y
x
0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1


T vs r at t/tg=0.2
Temp
r
0 10 20 30 40 50 60 70 80 90 100
0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1


T vs r at t/tg=0.25
Temp
100
95
90
85
80
75
70
65
60
55
50
45
40
35
30
25
20
15
10
5
0
y
x
0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1


T vs r at t/tg=0.25
Temp
r
0 10 20 30 40 50 60 70 80 90 100
0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1

6.4 Test Problem 3(moving Gaussian of varying width)

The domain is square of length 1, the boundary condition is zero. The initial condition is a Gaussian in x direction and a sine curve in y direction. The source term is derived below.

The original parabolic differential equation is

$$\frac{\delta T}{\delta t} = \alpha \nabla^2 T + S \tag{6.28}$$

$$\text{Let } T = E(x,t)X(x)Y(y) \tag{6.29}$$

$$where\ X(x) = x(L-x)\ , Y(y) = C_o sin\left(\frac{\pi y}{L}\right) \tag{6.30}$$

$$E(x,t) = e^{-\left(\frac{x-x_p(t)}{s(t)}\right)^2} \tag{6.31}$$

is the Gaussian distribution Function at x_p with width s

$$where\ x_p(t) = A_G + B_G sin\left(\frac{2\pi t}{\tau_G}\right) \tag{6.32}$$

$$\text{And } A_G = 0.5L\ , B_G = 0.25L\ , \tau_G = 0.1sec$$

$$s(t) = \frac{s_{max} + s_{min}}{2} + \frac{s_{max} - s_{min}}{2} cos\left(\frac{4\pi t}{\tau_G}\right) \tag{6.33}$$

$$s_{max} = 0.2L, s_{min} = 0.05L$$

Substituting eqn 6.29 in solution in eqn 6.28. We get the expression for source term

$$S(x,y,t) = T_t - \alpha(T_{xx} + T_{yy}) \tag{6.34}$$

$$T_t = (E)_t XY = XY\left(E_s s' + E_{x_p} x_P'\right) \tag{6.35}$$

$$T_{xx} = (EX)_{xx} Y = \{E_{xx}X + EX" + 2E_x X'\}Y \tag{6.36}$$

$$T_{yy} = EXY" \tag{6.37}$$

The derivatives required in the above equations are given below

$$E_x(x,t) = -\frac{2}{s^2}(x - x_p)E \tag{6.38}$$

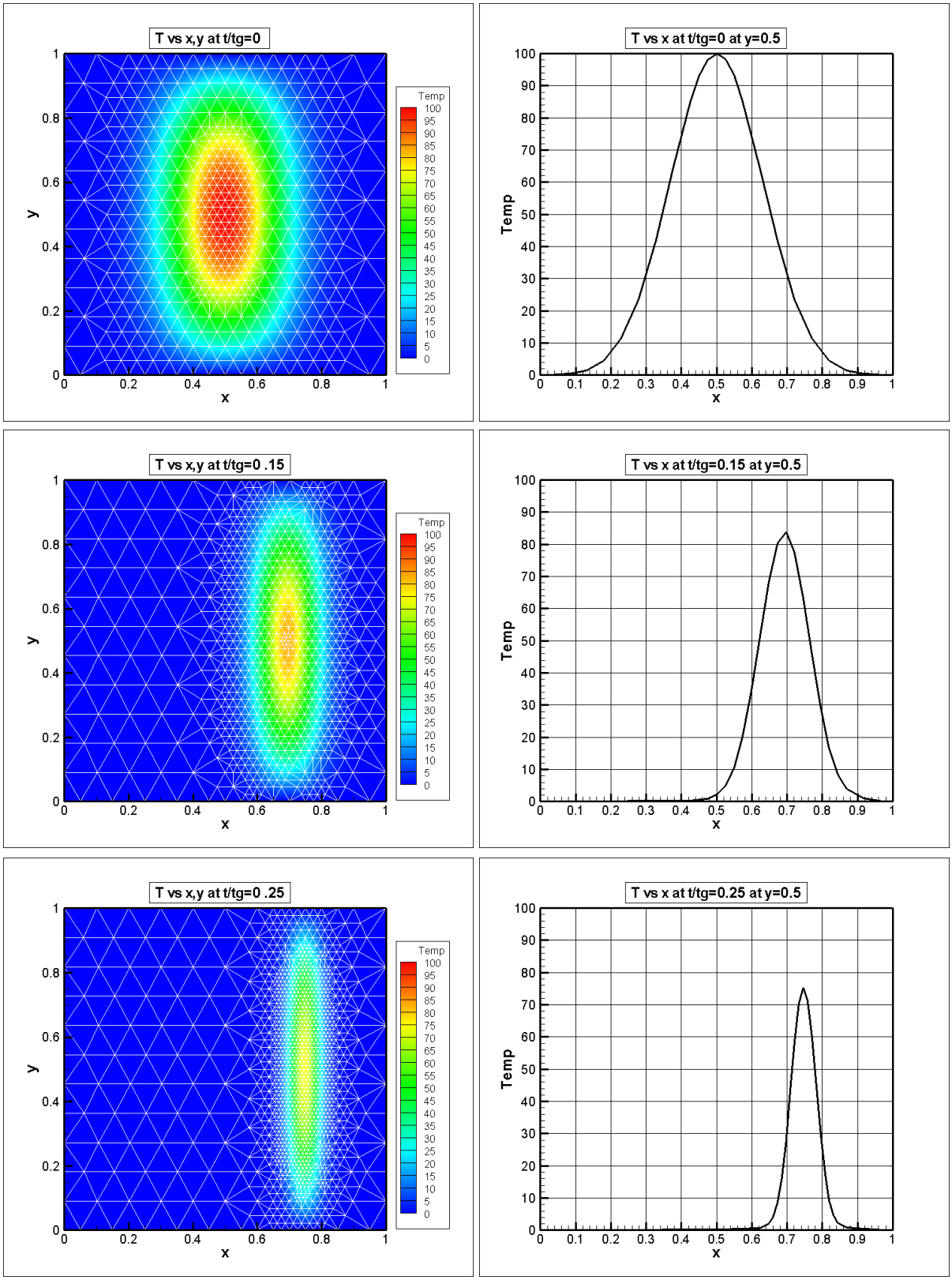
$$E_{x_P}(x,t) = -E_x \tag{6.39}$$

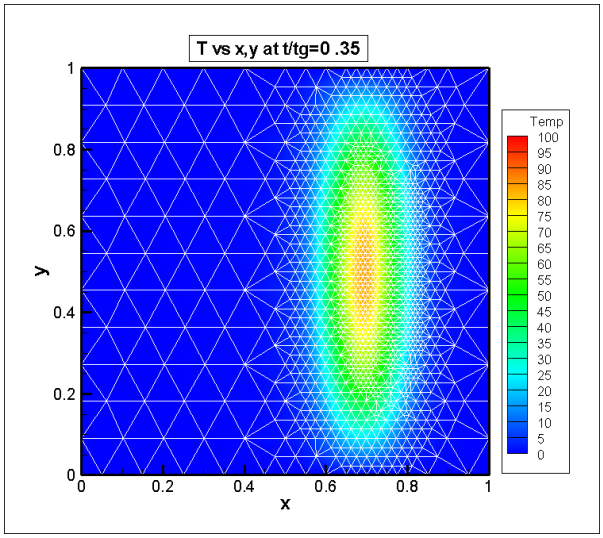
$$E_s(x,t) = \frac{2}{s^3}(x - x_p)^2 E \tag{6.40}$$

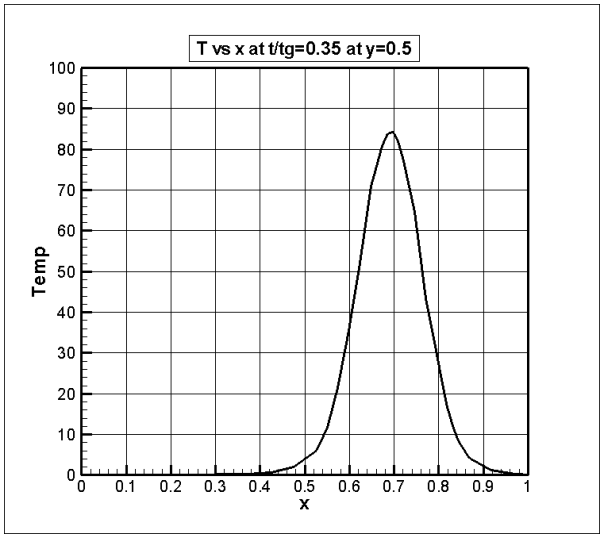
$$E_{xx}(x,t) = -\frac{2}{s^2}(E + (x - x_p)E_x) \tag{6.41}$$

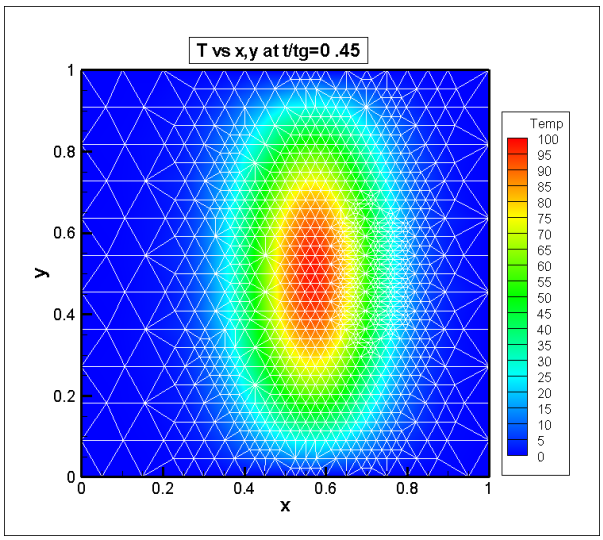
The solution is a Gaussian moving in the x direction with varying width is presented below. Contour Plots T(x,y) at different t/τ_g for half of the time cycle is given below. To the right is a cross sectional view of solution at y=0.5.

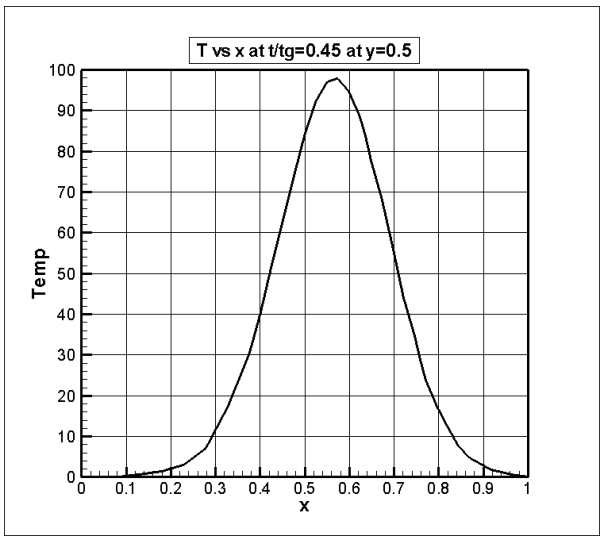
Figure 6.7 Contour plots of T vs x,y for different t/tg

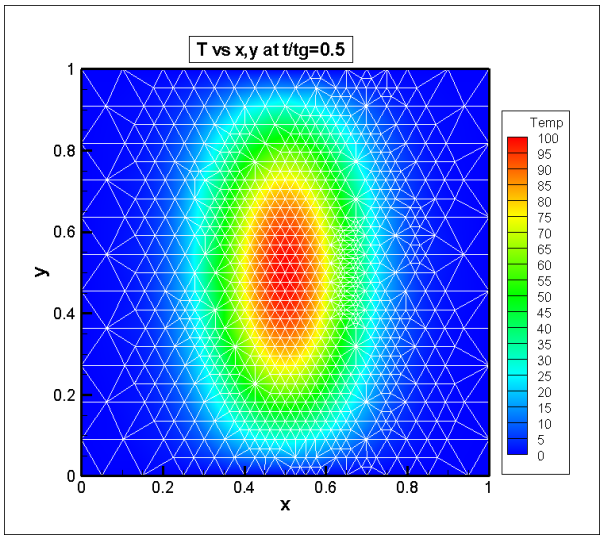


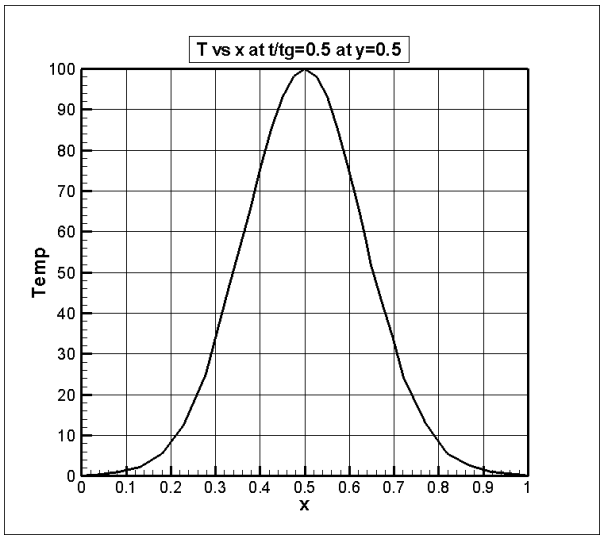

T vs x,y at t/tg=0 .35
Temp
100
95
90
85
80
75
70
65
60
55
50
45
40
35
30
25
20
15
10
5
0
y
x
0
0.2
0.4
0.6
0.8
1


T vs x at t/tg=0.35 at y=0.5
Temp
x
0
10
20
30
40
50
60
70
80
90
100
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


T vs x,y at t/tg=0 .45
Temp
100
95
90
85
80
75
70
65
60
55
50
45
40
35
30
25
20
15
10
5
0
y
x
0
0.2
0.4
0.6
0.8
1


T vs x at t/tg=0.45 at y=0.5
Temp
x
0
10
20
30
40
50
60
70
80
90
100
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


T vs x,y at t/tg=0.5
Temp
100
95
90
85
80
75
70
65
60
55
50
45
40
35
30
25
20
15
10
5
0
y
x
0
0.2
0.4
0.6
0.8
1


T vs x at t/tg=0.5 at y=0.5
Temp
x
0
10
20
30
40
50
60
70
80
90
100
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

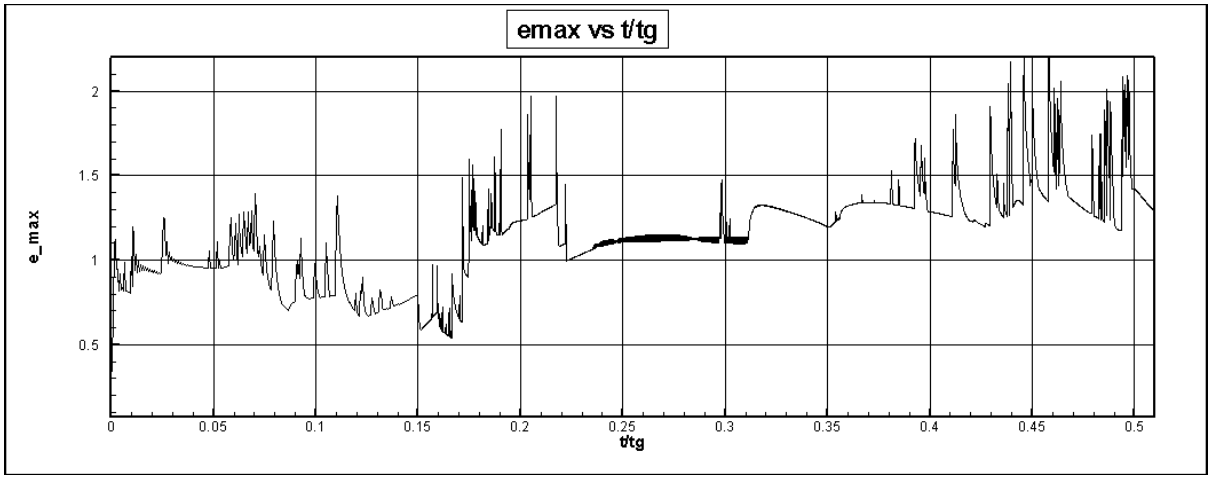


Figure 6.8 Maximum error vs t/tg

6.5 Observations (For all test problems)

- In the initial mesh,(Figure 6.1 and Figure 6.4), we could see that the grid was not able to map the curve due to its high curvatures. This would have been a source of errors.
- In the initial mesh, the methodology refined the grid wherever there were large curvatures the left the remaining grid intact.
- As the algorithm progressed (Figure 6.3, Figure 6.6 & Figure 6.7), the numerically calculated Gaussian solution's peak moved.
- Along with that the peak refinement also moved. It refined the regions of the peak more intensely then other regions and coarsened the regions that were losing curvature.
- As the curvature increased so did the degree of refinement fig 6.7.
- In the entire observation period (refer Figure 6.2, Figure 6.5 & Figure 6.8)the maximum error was plotted as a function of t/t_g. It was always hovering around 1% but below 2%.

6.6 Inference

- We can see that initial mesh (Figure 6.4), is not able to capture the solution. Thus grid modification has to be done in such cases.
- The mesh was being refined in places of high curvatures (Figure 6.3 and Figure 6.6) thus the refinement indicator was working as expected. It was ensuring a smooth curve.
- In Figure 6.7, the indicator was working as expected since it was increasing refinement with high curvature and reducing it with decrease in curvature.
- The grid refinement methodology was ensuring a normal triangular unstructured domain throughout the observation and it created no mesh defects like a one cell in contact with two cells at a face.
- The errors were always hovering near 1% and below 2% throughout the observation period, thus the methodology works. Thus the aim the work is complete.

Chapter 7 Conclusions and Future work

A problem of optimizing the grid refinement was introduced. An extensive literature survey of various methodologies was done. A grid generation methodology that can create unstructured triangular mesh on any domain was looked at and implemented. A numerical method was developed that can solve the diffusion equation which is a parabolic equation on an unstructured triangular mesh. A refinement indicator was developed that could tell where on the grid refinement is required and where coarsening is required. A grid modification methodology was developed that could refine and coarsen unstructured triangular grids. Three test problems where generated and solved using the grid refinement approach such that errors were below prescribed limit and the method was computationally optimum to a good degree. The curvature and errors are related.

Future work includes extending the methodology for 3-D diffusion equation. The refinement indicators can still be further improved to make it error based. A dependence relation between curvature and errors is found out. An analysis of interpolation errors can be done to find out the best suited interpolation technique. A computational time analysis can to be done to see how much of computational time the method saves as compared to the refinement of the entire domain. The method can be extended to 3-D Navier stokes solver.

Bibliography

[1] Baines, M. (1998). Grid adaptation via node movement. *Applied Numerical Mathematics*, 77-96.

[2] Bank, R. E., & Xu, J. (1996). An algorithm for coarsening unstructured meshes. *Numer. Math vol 73*, 1–36.

[3] Bowyer, A. (1981). Computing Dirichlet tessellations . *The Computer Journal, Vol. 24*, 162-166.

[4] Coeiho, P., & Argain, J. (1997). A local grid refinement technique based upon Richardson extrapolation. *Applied Mathmatical Modelling Vol 21*, 427-436.

[5] Jasak, H. (1996). *Error Analysis and Estimation for the Finite Volume Method with Applications to Fluid Flows.* PhD thesis, Imperial College of Science, Technology and Medicine, Department of Mechanical Engineering, London.

[6] Jones, M. T., & Plassmann, P. E. (1997). Adaptive refinement of unstructured finite-element meshes. *Finite Elements in Analysis and Design 25* , 41-60.

[7] Marchant, M., Weatherill, N., & Hassan, O. (1997). The adaptation of unstructured grids for transonic viscous flow simulation. *Finite Elements in Analysis and Design 25*, 199-218.

[8] Maria, & Rivara, C. (1984). Mesh Refinement Processes Based on the Generalized Bisection of Simplices. *Journal on Numerical Analysis Vol. 21, No. 3*, 604-613.

[9] Muzaferija, S., & Gosman, D. (1997). Finite-Volume CFD Procedure and Adaptive Error Control Strategy for Grids of Arbitrary Topology. *Journal of Computational Physics 138*, 766–787.

[10] Oden, J., Strouboulis, T., & Devloo, P. (1986). Adaptive Finite Element Methods For The Analysis Of Inviscid Compressible Flow:Part I. Fast Refinement And Moving Mesh Methods For Unstructured Meshes. *Computer Methods in Applied Mechanics and Engineering 59*, 327-362.

[11] Prabhudharwadkar, D. (2007). *Analysis of hydrogen diffusion in nuclear reactor containment.* PhD thesis, IIT Bombay, Mechanical Engineering, Mumbai.

[12] Ripley, R., Lien, F., & Yovanovich, M. (2003). Adaptive Mesh Refinement of Supersonic Channel Flows on Unstructured Meshes. *International Journal of Computational Fluid Dynamics*, 1–10.

[13] Silva, L. F., Azevedo, J. L., & Korzenowski, H. (2000). Unstructured Adaptive Grid Flow Simulations of Inert and Reactive Gas Mixtures. *Journal of Computational Physics 160*, 522–540.

[14] Tomlin, A., Ghorai, S., Hart, G., & Berzin, M. (2000). 3-D Multi-scale air pollution modelling using adaptive unstructured meshes. *Environmental Modelling & Software 15*, 681–692.

[15] Waanders, B. G., & Carnes, B. R. (2010). Optimization under adaptive error control for finite element based. *Computational Mechanics 47 (1)*, 49-63.

[16] Walter, M. A., Abdul, A. A., Silva, L. F., & Azevedo, J. L. (2005). Evaluation of adaptive mesh refinenement and coarsening for the computation of compressible flows on unstructured meshes. *International Journal for Numerical Methods in Fluids*, 999–1014.

[17] Wang, Y., & Ragusa, J. C. (2011). Standard and goal-oriented adaptive mesh refinement applied to radiation transport on 2D unstructured triangular meshes. *Journal of Computational Physics 230*, 763–788.

[18] Watson, D. (1981). Computing n-dimensional Delaunay tessellation with application to Voronoi polytopes. *The Computer Journal, Vol. 24.*, 167-172.

[19] Weatherill, N. P. (1988.). A method for generating irregular computational grids in multiply connected planar domains. *Int. J. Numerical Methods in Fluids, Vol. 8*, 181-197.

[20] Weatherill, N. P. (1992). Delaunay triangulation in computational fluid dynamics. *Computers Mathematics with Applications Vol. 24*, 129-150.

[21] Weatherill, N., & Marcurn, D. L. (1995). A procedure for efficient generation of solution adapted unstructured grids. *Computational Methods in Applied Mechanical engeneering*, 259-268.

[22] Wikipedia. (n.d.). *An online enclyclopedia*. Retrieved May 10, 2011, from http://en.wikipedia.org/wiki/Ruppert%27s_algorithm