Thesis Report

On

IMPROVEMENT OF FINITE ELEMENT MESHES

Submitted in the partial fulfillment of requirement for the award of the degree of

Master of Engineering

IN

MECHANICAL

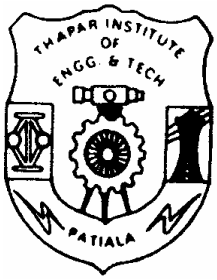
(CAD/CAM and ROBOTICS)

Submitted by

MOHINDER MOHAN
Roll No. : 8048114

Under the guidance of

Mr. J.S.SAINI
Lecturer
T.I.E.T, Patiala



Department of Mechanical Engineering
THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY
(DEEMED UNIVERSITY)
PATIALA (PUNJAB)-147004

CERTIFICATE

This is to certify that the Thesis report entitled **"IMPROVEMENT OF FINITE ELEMENT MESHES"** submitted by **MOHINDER MOHAN** in the partial fulfillment of the requirement for the award of the degree of **Master of Engineering in Mechanical (CAD/CAM and ROBOTICS) Engineering** to **Thapar Institute of Engineering and Technology (Deemed University), Patiala,** is a record of candidate's own work carried out by him under my supervision and guidance. The matter embodied in this report has not been submitted in part or full to any other university or institute for the award of any degree.

(Mr. J.S.SAINI)
Lecturer, Mechanical Engineering Department.
Thapar Institute of Engg. &Technology,Patiala

Countersigned by:

Dr. S.K. MOHAPATRA
Prof. & Head, MED.
THAPAR INSTITUTE OF ENGG. &TECH.
PATIALA-147004(PUNJAB)

Dr. T.P. SINGH
Dean, Academic Affairs
THAPAR INSTITUTE OF ENGG. &TECH.
PATIALA-147004(PUNJAB)

Dated:

Acknowledgement

I express my sincere gratitude to my guide**, Mr. J.S.SAINI** Lecturer, Mechanical Engg. Department at Thapar Institute of Engineering & Technology, for his valuable guidance, proper advice, painstaking and constant encouragement during the course or my work on this seminar.

I also feel very much obliged to **Dr. S.K MOHAPATRA,** Professor & Head, Department of Mechanical Engg. for his encouragement and inspiration for execution of the seminar work.

I am deeply indebted to my parents for their inspiration and ever encouraging moral support, which enabled me to pursue my studies.

I am also very thankful to the entire faculty and staff members of Mechanical engineering Department for their direct–indirect help and cooperation.

Dated:

MOHINDER MOHAN
(ROLL NO. 8048114)

ABSTRACT

In the mechanical systems there are numerous applications of heat transfer and temperature plays a very important role. If values of temperatures at node are not accurate it is not possible to get accurate results for heat transfer analysis.

Now, Finite Element Method which has increasingly becoming an integral tool for CAD due to the ongoing revolution in computer field is used for the analysis of heat transfer by number of researchers.

Finite Element Mesh refinement has an effect on the accuracy of the solution. The present work describes the effect of mesh refinement with regard to heat transfer.

LIST OF FIGURES

CHAPTER 1

CHAPTER 2

CHAPTER 3

CHAPTER 4

LIST OF TABLES

CHAPTER 4

NOMENCLATURE

N^e Shape function for element e
n Number of element
q Temperature gradients, C/mm
r Radius, mm
r_0 Inner radius of the cylinder, mm
r_n Outer radius of the cylinder, mm
r_{e-1} Inner radius of element e, mm
r_e Outer radius of element e, mm
T_0 Temperature at inner radius, ^oC
T_n Temperature at outer radius, ^oC
K_{ij}^e Stiffness matrix of element, e
k Thermal conductivity, W/mm^oC

INDEX

CHAPTER-4

CHAPTER-5

In the present work, the effect of mesh refined on the accuracy of the results is analysed using FEM, DQM and HDQM. This technique of mesh refinement has emerged as a powerful analysis tool with the rapid development in the computer field.

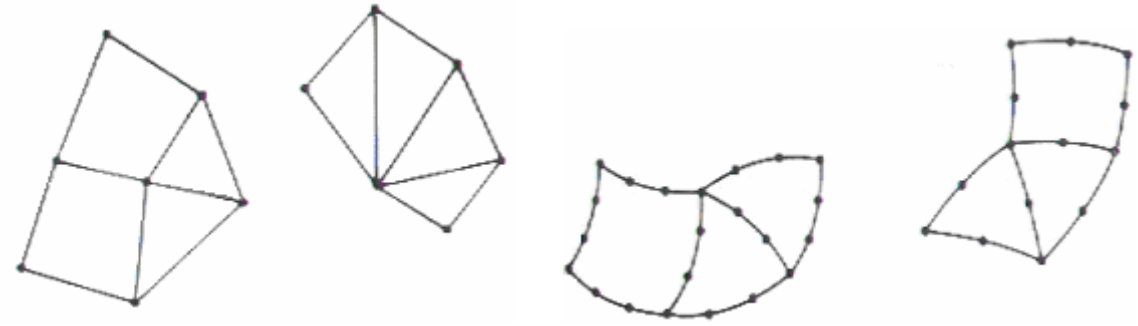
1.1 INTRODUCTION

Despite the fact that the finite element mesh generation has been developing over several decades now, a variety of real-life engineering problems imposes additional requirements on existing mesh generation technologies. For example, local mesh refinement zones around common stress concentrators (mechanical constraints and point loads) must be readily generated when the user requires that in the model's input data.

Adaptive finite element computations rely on adjustments of the spatial resolution of the domain discretization to deliver higher accuracy where it is needed. When the domain is discretized into a finite element mesh, a possible option, somewhat expensive and in some cases complex, is to create a new mesh with the desired resolution, i.e., remeshing. Another alternative is to adjust the density of the mesh by performing local refinement of the existing mesh so that in some regions finite elements are split to decrease their "size", in other regions they are merged to reduce the resolution. Both choices, remeshing and refinement have their advantages and disadvantages. We are not going to argue for one or the other option. Rather, we assume that refinement had been adopted as the method of choice. What are the desirable properties of a mesh refinement algorithm?

- It should certainly be efficient, in that, it should not become a bottleneck of the adaptive computations.
- It should generate good quality geometric meshes, i.e., elements must remain well-shaped on refinement and unrefinement.
- Finally, it must be robust, which is usually expressed as the requirement to terminate with a valid result in finite time.
- An additional plus is if it generates nested meshes in the refinement hierarchy, which simplifies the incorporation of multigrid solvers.

Generation of meshes of a single element type is easy because elements of the same degree are compatible with each other Fig 1.1. Mesh refinements involve several options. The mesh can be refined by subdividing existing elements into two or more elements of the same type Fig 1.2a. This is called h-version mesh refinement. Alternatively, existing elements can be replaced by elements of higher order Fig 1.2b. This is called p-version mesh refinement. There is also h, p-version mesh refinement, in which elements are subdivided into two or more elements in some places and replaced connecting elements of same order.



(a) (b)

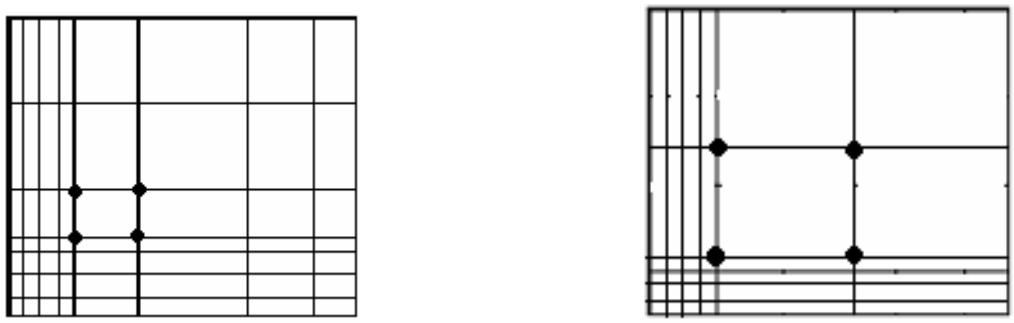
Fig. 1.1 Elements of same degree

The C^o elements of the same order ensure the C^o continuity along the element interface: (a) linear elements; (b) quadratic elements.

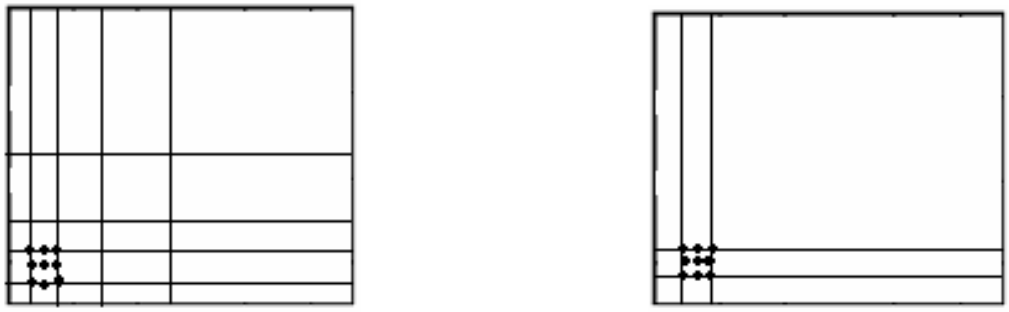
(a)

(b)

Fig. 1.2 Mesh refinement



(a)



(b)

Fig. 1.3 Local mesh refinement

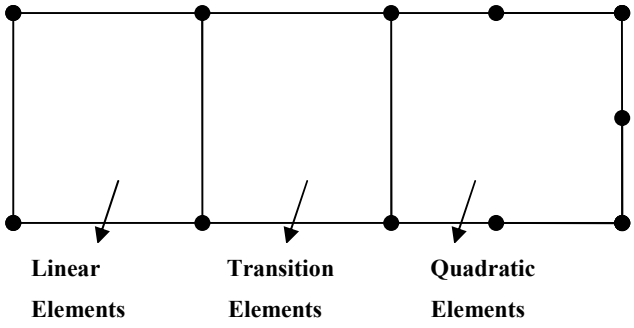


Fig 1.4a Transition element

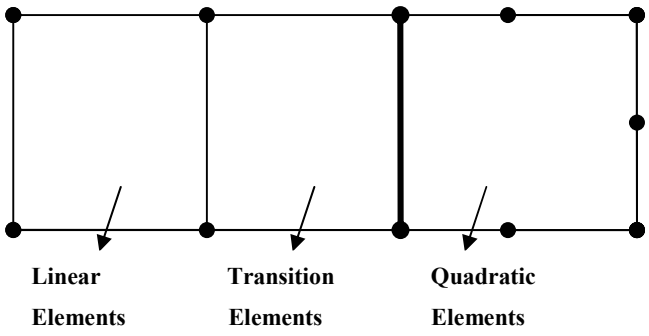


Fig 1.4b Use of transition element

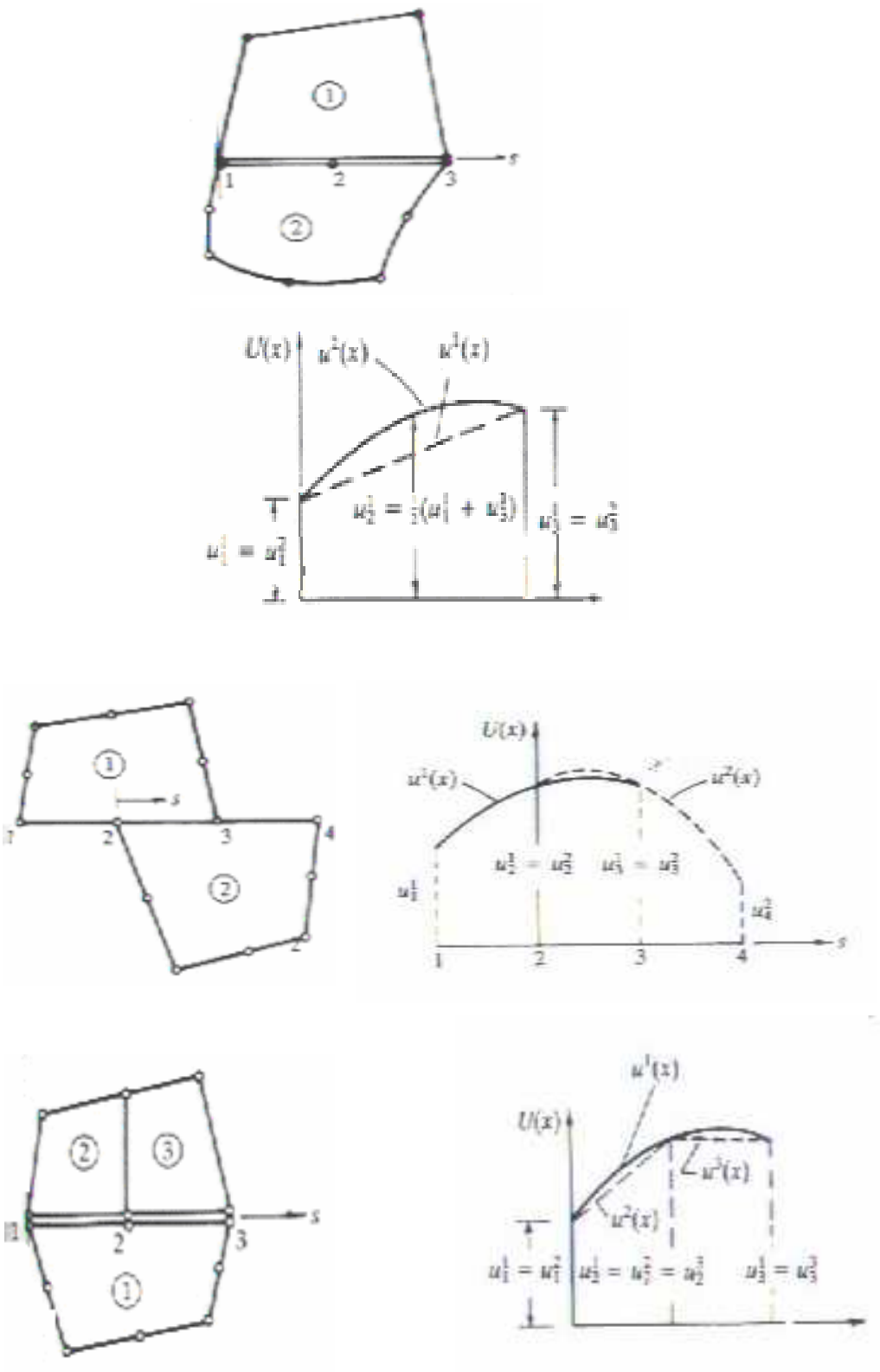


Fig. 1.5 Element connections

with higher-order elements in other places. Generally, local mesh refinements should be such that elements of very small size are not placed adjacent to those of very large size Fig 1.3.

Combining elements of different kinds naturally arises in solid and structural mechanics problems. For example, plate bending elements (2-D) can be connected to a beam element (1-D). If the plate element is based on classical plate theory, the beam element should be based on the Euler-Bernoulli beam theory so that they have the same degrees of freedom at the connecting node. When a plane elasticity element is connected to a beam element, which is not compatible with the former in terms of the degrees of freedom at the nodes, one must construct a special element that makes the transition from the 2-D plane elasticity element to the 1-D beam element. Such an element is called a transition element.

Combining elements of different order, say a linear to a quadratic element, may be necessary to accomplish local mesh refinements. There are two ways to do this. One way is to use a transition element that has a different impose a condition that constrains the midside node to have the same values as that experienced at the node by the lower-order element Fig 1.4b. However, such combinations do not enforce interelement continuity of the solution. Fig 1.5 shows element connections that do not satisfy C^o continuity along the connecting sides. Use of transition elements and constraint conditions in local mesh refinements is a common practice. Fig.1.5 shows a few examples of such refinements.

Virtually every phenomenon in nature, whether biological, geological or mechanical can be described with the help of laws of physics, in term of algebraic, differential, or integral equations relating various quantities of interest. A plate subjected to various mechanical loads, thermal loading, aerodynamic loads are few examples of practical problems. There are various methods for the analysis of the objects. For studying the physical phenomenon the two major steps are as follows:

- Mathematical formulation of physical process.
- Numerical analysis of the mathematical model.

Various methods are used for implementing the above steps. These methods are given in the following sections.

1.2 FINITE ELEMENT METHOD

Basic idea of FEM originated from advances in aircraft structural analysis. Hrenikoff presented a solution of elasticity problem using the "frame work method". Courant's paper which use the piecewise polynomial interpolation over triangular sub regions to model torsion problem appeared in 1943. Turner, et al. derived stiffness matrices for truss, beam and other elements and presented their findings in 1956. The term FEM was firstly used in 1960. There are various approaches to solve the Residual equation in the FEM. These are as follows:

- Collocation method.
- Sub domain method
- Least square method
- Galerikan approach
- Ritz variational approach

From these methods the Galerikan approach has wide scope & can be applied to any differential equation. The Ritz method is applicable, where the integral equations are used.

1.3 FINITE DIFFERENCE METHOD

The finite difference method is basically a finite difference approximation of a differential equation. Firstly, the governing equation in the form of differential is derived and after that the derivatives are replaced by the difference quotient (or the function is expanded by the Taylor series) that involves the value of the solution at discrete mesh point of the domain. The resulting algebraic equations are solved, after imposing the boundary condition, for the solution at the mesh point.

Among the major techniques for numerically analyzing continuous problems, Finite Difference Method (FDM) was the earliest. The FDM uses divided difference expressions established from a local Taylor series to replace the differential or partial differential operators. It is difficult to deal with problems showing non-rectangular or complex curvilinear geometries by using the traditional FDM numerical techniques. In deriving the finite difference equations, the derivatives of the variable functions in the

functional are replaced by the traditional finite difference operators. This method can be used to carry out the finite difference discretization for a problem having a complicated domain configuration by adopting an irregular grid model.

The most commonly used numerical methods for the different analysis studies are finite difference and boundary element methods and most engineering problems can be solved by these methods to satisfactory accuracy if a proper and sufficient number of grid points are used. However, in a large number of practical applications where only reasonably accurate solutions at few specified physical coordinates are of interest, the finite element or finite difference method becomes inappropriate since they still require a large number of grid points and so large a computer capacity, especially in the cases of nonlinear problems where iteration becomes inevitable. Consequently, both CPU time and storage requirements are often considerable for the standard methods.

This section presents a quick overview about the Finite Difference Method. This method can be used to solve any Partial Differential Equation (PDE). It can handle very well free boundary problems and optimal stopping problems.

Assuming that u is function of the independent variables x and y, the x-y plan can be divided in mesh points equal to δ x = h e, δ y = k, as showed below:

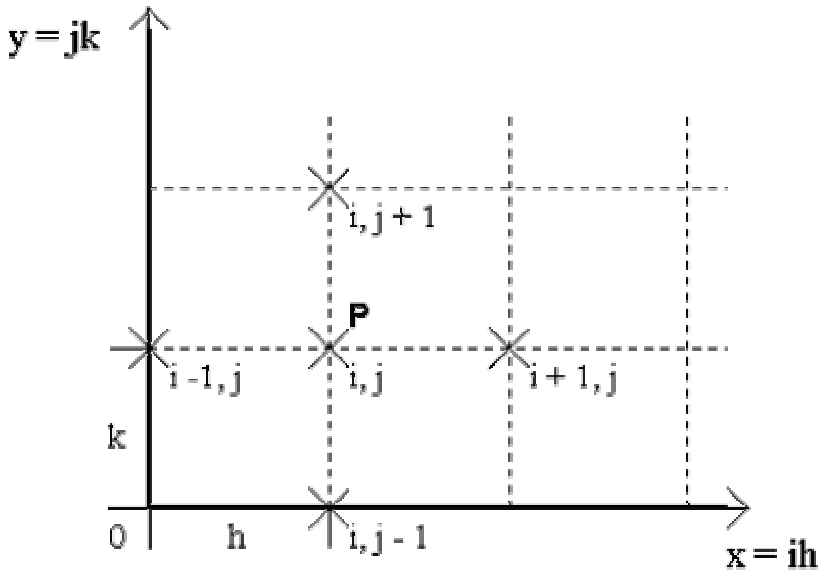


Fig. 1.6 Mesh points

u can be evaluated at point P by :

$$u_p = u(ih, jk) = u_{ij}$$

The value of the second derivative at P could also be evaluated by:

$$\left(\frac{\partial^2 u}{\partial x^2}\right)_p = \left(\frac{\partial^2 u}{\partial x^2}\right)_{ij} \cong \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2}$$

$$\left(\frac{\partial^2 u}{\partial y^2}\right)_p = \left(\frac{\partial^2 u}{\partial y^2}\right)_{ij} \cong \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{k^2}$$

The value of the first derivative at P can be evaluated by three approximations:

1) Central difference:

$$\left(\frac{\partial u}{\partial y}\right)_p = \left(\frac{\partial u}{\partial y}\right)_{ij} \cong \frac{u_{i+1,j} - u_{i-1,j}}{2h}$$

$$\left(\frac{\partial u}{\partial y}\right)_p = \left(\frac{\partial u}{\partial y}\right)_{ij} \cong \frac{u_{i,j+1} - u_{i,j-1}}{2k}$$

2) Forward difference:

$$\left(\frac{\partial u}{\partial x}\right)_p = \left(\frac{\partial u}{\partial x}\right)_{ij} \cong \frac{u_{i+1,j} - u_{i,j}}{h}$$

$$\left(\frac{\partial u}{\partial y}\right)_p = \left(\frac{\partial u}{\partial y}\right)_{ij} \cong \frac{u_{i,j+1} - u_{i,j}}{k}$$

3) Backward difference:

$$\left(\frac{\partial u}{\partial x}\right)_p = \left(\frac{\partial u}{\partial x}\right)_{ij} \cong \frac{u_{i,j} - u_{i-1,j}}{h}$$

$$\left(\frac{\partial u}{\partial y}\right)_p = \left(\frac{\partial u}{\partial y}\right)_{ij} \cong \frac{u_{i,j} - u_{i,j-1}}{k}$$

1.4 DIFFERENTIAL QUADRATURE METHOD

Differential Quadrature Method is the in seeking a more efficient numerical method which requires smaller number of grid points yet achieves acceptable accuracy. The method of Differential Quadrature (DQ), which is based on the assumptions that the partial derivatives of a function in one direction can be expressed as a linear combination of the function values at all mesh points along that direction was introduced by Bellman et al. The method of Differential Quadrature circumvents the above difficulties by computing a moderately accurate solution from only a few points. Since then, applications of differential quadrature method to various engineering problems have been investigated and their successes have demonstrated the potential of the method as an attractive numerical analysis technique.

In current scenario the following methods are the area of research and can be used for the analysis of the rectangular plate.

- Moving least square differential quadrature method
- Differential Quadrature Element Method
- Harmonic differential quadrature method

1.4.1 MOVING LEAST SQUARE DIFFERENTIAL QUADRATURE METHOD

One of the major drawbacks of DQ method is that a regular discrete node arrangement and a simple domain are usually necessary in order to express the weighting coefficients explicitly. Although the domain transformation, such as isoparametric mapping, is possible for irregular shapes, it causes a significant loss of efficiency and simplicity, especially for problems involving irregular geometries and higher-order partial derivatives.

The approximate function $u^h(x)$ of u(x) is obtained by the MLS method and the weighting coefficients c_{ij} are directly computed from the partial derivatives of shape

functions. The MLS approximate function uh(x) of u(x) at any point x in the domain Ω is written as

$$u^h(x) = \sum_{i=1}^{m} p_i(x) a_i(x) = p^T(x) a(x)$$

Where $p_i(x)$ is a finite set of basis functions and $a_i(x)$ are unknown coefficients. m denotes the total number of basis functions. For convenience, the monomial basis functions are usually used and this assures that the basis is complete. For example, a complete quadratic basis in 2D case with m = 6 is of the form,

$$p(x) = \begin{bmatrix} 1 & x & y & x^2 & xy & y^2 \end{bmatrix}^T$$

Noting that the coefficients $a_i(x)$ is functions of the spatial coordinates x, the expression for $a_i(x)$ can be obtained at any point x by minimizing the following weighted quadratic form:

$$\prod(a) = \sum_{i=1}^{n} w(x - x_i)(u^h(x)_i - u_i)^2 = \sum_{i=1}^{n} w_i(x)(p^T(x_i)a(x) - u_i)^2$$

The weight function of Gaussian type with a circular support is adopted for the MLS approximation because its partial derivatives with respect to either the x or y coordinate exist to any desired order. It takes the form of

$$w_i(x, y) = \begin{cases} \dfrac{\exp(-(d_i/c)^2) - \exp(-(r/c)^2)}{1 - \exp(-(r/c)^2)}, & d_i \le r, \\ \text{and "0" if} & d_i \ge r \end{cases}$$

Where $d_i = \sqrt{(x - x_i)^2 + (y - y_i)^2}$ is the distance from a discrete node x_i to a sampling point x in the domain of support with radius r, and c is the dilation parameter. c = *r/4* is used in the computation.

1.4.2 DIFFERENTIAL QUADRATURE ELEMENT METHOD

Han and Liew firstly introduce the DQEM for moderately thick plate and solved the different problem for thick and thin plates without any discontinuity. In these cases, the weighting coefficients of the DQEM are the same as that of the ordinary DQ method since only second-order differential equations are involved. Placing curvature quantities as the additional degree of freedom at boundary points was proposed to solve the problem of applying multi-boundary conditions for isotropic rectangular plates in bending. Since then a variety of structure problems, including deflection, free vibration and buckling problems, and initial value problems have been successfully solved with the help of DQEM.

The fourth order function can be solved using this method and the solution function $w(x)$ is assumed as:

$$w(x) = \sum h_j(x) w_j + h_{N+1}(x) w_1' + h_{N+2}(x) w_N'$$

where N, h_j(x) and w_j are the total number of grid points in the entire domain including the two end points, the interpolation function and the solution values at grid points j, w_1' and w_N' first derivatives at grid points 1 and $N.h_j(x)$ are $(N+1)$th order polynomials. Again, the kth order derivative of the solution function at grid point i can be computed by:

$$w^{(k)}(x_j) = \sum h_j^{(k)}(x) w_j + h_{N+1\,N+1}^{(k)}(x) w_1' + h_{N+2}(x) w_N' = \sum_{j=1}^{N+2} E_{ij} \delta_j$$

Where E_{ij} (i . 1, 2, . . . , N, j . 1, 2, . . . , N . 2) are also called the weighting coefficients of the kth -order derivative. The DQ element equation can be written symbolically as:

$$[k][\delta] = [F]$$

Where δ and F are generalized displacement and force vector. Using this method the DQM method can be used more practically.

1.4.3 HARMONIC DIFFERENTIAL QUADRATURE METHOD

In a recent approach the original differential quadrature approximation called the Harmonic Differential Quadrature (HDQ) has been proposed by Striz et al.. Unlike the differential quadrature that uses the polynomial functions, such as Lagrange interpolated, and Legendre polynomials as the test functions, harmonic differential quadrature uses harmonic or trigonometric functions as the test functions. As the name of the test function suggested, this method is called the HDQ method. In this method the harmonic test function is used and can be defined as:

$$g_k(x) = \frac{\sin\frac{(x-x_1)\pi}{2}.....\sin\frac{(x-x_{k-1})\pi}{2}\sin\frac{(x-x_{k+1})\pi}{2}.....\sin\frac{(x-x_N)\pi}{2}}{\sin\frac{(x_k-x_1)\pi}{2}.....\sin\frac{(x_k-x_{k-1})\pi}{2}\sin\frac{(x_k-x_{k+1})\pi}{2}.....\sin\frac{(x_k-x_N)\pi}{2}}$$

.

using this method the the weighting coefficient are calculated as follows:

$$A_{ij}^{(1)} = \frac{(\pi/2)P(x_i)}{P(x_j)\sin((x_i-x_j)/2)\pi}\text{ , I,j = 1, 2, 3,……,N}$$

where,
$$P(x_j) = \prod_{j=1, j\neq i}^{N} \sin\left(\frac{x_i - x_j}{2}\pi\right) \text{ for j = 1, 2,……, N}$$

the weighting coefficient of the second order derivatives $A_{ij}^{(2)}$ for $i \neq j$ can be obtained using following formula:

$$B_{ij} = A_{ij}^{(1)}\left[2A_{ij}^{1} - \pi ctg\left(\frac{x_i - x_j}{2}\right)\pi\right], \text{ I, j = 1, 2,……., N}$$

The weighting of the first order and second-order derivative $A_{ij}^{(p)}$ for i = j are given as:

$$A_{ii}^{(p)} = -\sum_{j=1, j\neq i}^{N} A_{ij}^{(p)} \text{ , p=1 or 2; and for i 1, 2, 3,……., N}$$

The weighting coefficient of the third and fourth order derivative can be calculated easily from A_{ij} and B_{ij}

$$C_{ij} = \sum_{k=1}^{N} A_{ik}B_{kj}\ ,\ D_{ij} = \sum_{k=1}^{N} B_{ik}B_{kj}$$

With the advance in computer technology, the numerical technique has made significant progress in the past 50 years. Numerical approach has from then on become a major branch in the field of engineering or scientific research. Among the major techniques for numerically analyzing continuous problems, finite difference method (FDM) was the earliest. The FDM uses divided difference expressions established from a local Taylor series to replace the differential or partial differential operators. It is difficult to deal with problems showing non-rectangular or complex curvilinear geometries by using the traditional FDM numerical techniques. In deriving the finite difference equations, the derivatives of the variable functions in the functional are replaced by the traditional finite difference operators. This method can be used to carry out the finite difference discretization for a problem having a complicate domain configuration by adopting an irregular grid model.

The DQ approximates a derivative or partial derivative of a variable function with respect to a co-ordinate at a discrete point as a weighted linear sum of the function values at all discrete points along that co-ordinate direction. By using the DQ weighting coefficients and adopting the operations of tensor products, certain finite difference operators can be obtained. So basically the DQM is the numerical solution technique which is used to solve the linear and non linear partial differential equations. As the solution of the algebraic equation is easy and by this method the linear algebraic equation to the differential equation can be obtained.

Following are the review of the application of the differential quadrature method subjected to various analysis of the structural component.

2.1 LITERATURE REVIEW

W. E. Carroll *et. al.* [1] presented a development which serves to characterize the nature of an optimum finite-element idealization. It is shown, utilizing the displacement formulation, that a true minimum of the system potential energy must consider the idealization geometry as a primary parameter. As a consequence, two optimization equations result, one the usual equilibrium equation and the other a residual equation

involving gradients of the stiffness matrix and load vector resulting from changes in the idealization. A technique for determining the optimum solution was described and then applied to an elementary two-dimensional example. Practical recommendations are given based on an examination of the residuals associated with the optimization process.

William Prager [2] presented Rayleigh-Ritz procedure, which use "local" coordinate functions that differ from zero only over a few neighboring "elements" and determine the coefficients of that linear combination of these functions which minimizes the appropriate functional, for instance the potential energy in a problem concerning displacements of an elastic structure. Since the class of admitted functions has been artificially restricted in this procedure, the minimum value of the functional found in this manner is an upper bound for the true minimum. With reference to the structural problem mentioned above, this fact is usually expressed by the statement that the finite element solution overestimates the stiffness of the structure. The suggestion has recently been made that the geometry of the element grid should be included in the minimization procedure. In other words, of all finite element solutions with a given number of elements and a chosen type of local coordinate functions the one should be used that minimizes the appropriate functional.

Peter R. Eiseman [3] presented a review of adaptive grid generation with an emphasis on the basic concepts and the interrelationship between the various methods. The concepts are developed in a multifaceted progressive sense with enough detail so as to instill an operative spirit for the methods. The operational capabilities come from an explicit display of the necessary formulas for algorithmic construction. While virtually all adaptive procedures are aimed at problems with rapid solution variations, our main concern is the construction of methods that are not fundamentally restricted by the choice of problem or solution algorithm. Moreover, to maintain a simple treatment for the computational data and to have access to many of the best solution algorithms, we consider coordinate transformations. As a consequence, particular attention is given to grid point motion that occurs in response to the influence from the solution data, regardless of how that data was obtained. After some introductory discussion on the

utilization of solution data, the topic of grid point motion is addressed first in one dimension and then in higher dimensions. The basic equidistribution process is first seen from a dozen different viewpoints in one dimension. This is further amplified with the practical notions of precise coefficient specification, the attraction to a given grid, and the action of evolutionary forces. With the definition of the metric, the direct extension into higher dimensions is developed with curve-by-curve methods. This is followed by finite volume methods and variational methods. With the various movement strategies established in a multidimensional context, the next consideration is the temporal coupling of the movement with the solution algorithm. This is undertaken in the discussion of temporal aspects.

K. M. Liew *et. al.* [4] Obtained **a**xisymmetric free vibrations of moderately thick circular plates described by the linear shear-deformation Mindlin theory are analyzed by the differential quadrature (DQ) method. The first fifteen natural frequencies of vibration are calculated for uniform circular plates with free\ simply-supported and clamped edges. Through these computations the capability and simplicity of the differential quadrature method for moderately thick plate eigenvalue analysis is demonstrated\ and convergence and accuracy are thoughtfully examined. The case of a rigid point support at the plate centre is also considered in the present paper, for which special attention is paid to the capability and convergence of the current method.

In this method the grid points are designated by:

$$R_i = \frac{1}{2}\left[1 - \cos\left(\frac{(i-1)\pi}{N-1}\right)\right], i = 1,2,....., N.$$

A non-dimensional frequency parameter λ^2 is adopted for the results presentation\ and is defined as:

$\lambda^2 = \omega a^2 \sqrt{\rho h / D}$ and $\omega = \Omega \sqrt{E / \rho a^2 (1 - v^2)}$.

Xinwei Wang *et. al* [5] extended the Differential Quadrature Element Method to solve the problem to analyze the two dimensional plate problem. The DQM can be regarded as a special class of mixed collocation method. The DQM solution function for the one dimension problem can be considered as: -

$$w(x) = \sum_{j=1}^{Nn} l_j(x) w_j$$

Where N, $l_j(x)$ and w_j are the total number of grid points in the entire domain including the end points, the Lagrange interpolation function, and the solution values at grid points j, respectively. Then the kth-order derivative of the solution function at grid point i can be computed by

$$w_i^{(k)} = \sum_{j=1}^{N} l_j^k(x_i) w_j = \sum_{j=1}^{N} E_{ij} w_j \text{ where i=1 to N}$$

where E_{ij} are called the weighting coefficient of the kth order derivative. Thus the differential equation of the governing equation can be solved. The kth-order derivative of the solution function at grid point i can be computed by

$$w^{(k)}(x_j) = \sum h_j^{(k)}(x) w_j + h_{N+1\,N+1}^{(k)}(x) w_1' + h_{N+2}(x) w_N' = \sum_{j=1}^{N+2} E_{ij} \delta_j$$

where E_{ij} (i . 1, 2, . . . , N, j . 1, 2, . . . , N . 2) are also called the weighting coefficients of the kth -order derivative. The DQ element equation can be written symbolically as:

$$[k][\delta] = [F]$$

where δ and F are generalized displacement and force vector. And in this method to expedite the convergence rate the following non-uniform grid spacing was used.

$$x_k(y_k) = -\cos\frac{(k-1)\pi}{N-1} \text{ where k = 1, 2, ……, N.}$$

DQ element method has provided a new approach to applying the boundary conditions for the ordinary differential quadrature method and makes the DQM more practical. It should be noted, however, that the weighting coefficient matrix is asymmetric and DQEM solutions for multi element problems are sensitive to grid spacing.

K. M. Liew *et. al* [6] compared the accuracy of the Differential Quadrature method with the Harmonic Differential Quadrature method for the three dimensional analysis of the rectangular plate. A novel approach to the DQ approximation named the Harmonic Differential Quadrature (HDQ) method was proposed recently. Unlike the DQ method that uses the Lagrange interpolated polynomials as the test functions, the HDQ method used the trigonometric/harmonic functions as the test functions. It was reported that by using the harmonic functions, as the test functions, the HDQ method yields higher accuracy for solutions of higher mode vibration problems of beams and thin plates as well as problems with periodic behavior. Despite the promising results reported, the calculation of the weighting coefficients was cumbersome. Like the original DQ method, the HDQ method needed to solve a system of linear algebraic equations in order to determine the weighting coefficients.

In this, two methods of determining the weighting coefficients were outlined. In the first method, the Lagrange interpolated polynomials were used as the test function. The test function $f_k(x)$ is defined as

$$f_k(x) = \frac{M(x)}{(x - x_k)M^{(1)}(x_k)} \quad \text{for k= 1,2,……., N.}$$

where $M^{(1)}(x)$ is the first derivative of M(x) at point x_k and it can be expressed as:-

$$M(x) = \prod_{j=1}^{N}(x - x_j) \quad \text{and} \quad M^{(1)}(x_k) = \sum_{j=1, j\neq i}^{N}(x_k - x_j) \quad \text{for k=1, 2, ……..., N}$$

The second method of computing the weighting coefficients used the harmonic functions as the test function. As the name of the test function suggested, this method was called the HDQ method. The harmonic test function *gk* (*x*) used in the HDQ method was defined as

$$g_k(x) = \frac{\sin\frac{(x - x_1)\pi}{2} \ldots\ldots \sin\frac{(x - x_{k-1})\pi}{2} \sin\frac{(x - x_{k+1})\pi}{2} \ldots\ldots\ldots\ldots \sin\frac{(x - x_N)\pi}{2}}{\sin\frac{(x_k - x_1)\pi}{2} \ldots\ldots\ldots\ldots \sin\frac{(x_k - x_{k-1})\pi}{2} \sin\frac{(x_k - x_{k+1})\pi}{2} \ldots\ldots\ldots\ldots \sin\frac{(x_k - x_N)\pi}{2}}$$

.

$$\text{for k=1, 2, ….., N.}$$

According to the HDQ, the weighting coefficients of the first-order derivatives $A_{ij}^{(1)}$ for $i \neq j$ can be obtained using the following formula

$$A_{ij}^{(1)} = \frac{(\pi/2)P(x_i)}{P(x_j)\sin((x_i - x_j)/2)\pi}, \text{ I,j = 1, 2, 3,.....,N}$$

where

$$P(x_j) = \prod_{j=1, j\neq i}^{N} \sin\left(\frac{x_i - x_j}{2}\pi\right) \text{ for j = 1, 2,....., N}$$

The weighting coefficient of the second order derivatives $A_{ij}^{(2)}$ for $i \neq j$ can be obtained using following formula:

$$A_{ij}^{(2)} = A_{ij}^{(1)}\left[2A_{ij}^{1} - \pi ctg\left(\frac{x_i - x_j}{2}\right)\pi\right], \text{ I, j = 1, 2,....., N}$$

The weighting of the first order and second-order derivative $A_{ij}^{(p)}$ for i = j are given as

$$A_{ii}^{(p)} = -\sum_{j=1, j\neq i}^{N} A_{ij}^{(p)}, \text{ p=1 or 2; and for i 1, 2, 3,....., N}$$

from this method the comparison between the DQ and HDQ is done and the result shows the usage of both the methods for the different modes.

T.Y. Wu *et. al.* [7] gave differential quadrature method to solve boundary-value and initial-value differential equations for the linear or nonlinear nature.

According to this method derivative of a function Ψ (x) at point is given by:

$$\frac{d^r \psi(y_i)}{dy^r} = \sum_{k=1}^{N} W_{ik}^{r} U_k \qquad (i = 1, 2, 3, \ldots, N)$$

These are called the weighting coefficients of the r_{th} order derivative of the function at point x_i and $\{U_k\}=\{ U_1, U_2, \ldots\ldots, U_N \}$

Where $N=\sum_{j=1}^{N} n_j$, n_j = no of equations corresponding to the point x_i.

As in the traditional error only the function value is the independent variable and this is the only difference between the traditional and this DQM. So in the traditional DQM only one differential quadrature analog can be implemented. But in this proposed

differential quadrature one has the function values and their derivatives wherever necessary as the independent variables.

T. C. Fung [8] gave the different approach to modify the weighting coefficient matrices. For that let there be m boundary conditions for the differential equation. Equation can be given in the following non-homogenous mixed form as:

$$\gamma_{i1}\left.\frac{d^{m-1}y}{dx^{m-1}}\right|_{x=x_{ii}} + \gamma_{i2}\left.\frac{d^{m-1}y}{dx^{m-1}}\right|_{x=x_i} + \ldots\ldots + \gamma_m\gamma(\bar{x}_i) = \beta_i \; for \text{ i=1,2,....,m}$$

when n is replaced by n+m then the matrix is given as

$$\left\{\begin{array}{c} y_1^{(r)} \\ \vdots \\ y_n^{(r)} \\ \hline y_{n+1}^{(r)} \\ \vdots \\ y_{n+m}^{(r)} \end{array}\right\} = \left[\begin{array}{ccc|ccc} A_{11}^{(r)} & \cdots & A_{1n}^{(r)} & A_{1,n+1}^{(r)} & \cdots & A_{1,n+m}^{(r)} \\ \vdots & & \vdots & \vdots & & \vdots \\ A_{n1}^{(r)} & \cdots & A_{nn}^{(r)} & A_{n,n+1}^{(r)} & \cdots & A_{n,n+m}^{(r)} \\ \hline A_{n+1,1}^{(r)} & \cdots & A_{n+1,n}^{(r)} & A_{n+1,n+1}^{(r)} & \cdots & A_{n+1,n+m}^{(r)} \\ \vdots & & \vdots & \vdots & & \vdots \\ A_{n+m,1}^{(r)} & \cdots & A_{n+m,n}^{(r)} & A_{n+m,n+1}^{(r)} & \cdots & A_{n+1,n+m}^{(r)} \end{array}\right] \left\{\begin{array}{c} y_1 \\ \vdots \\ y_n \\ \hline y_{n+1} \\ \vdots \\ y_{n+m} \end{array}\right\}$$

or

$$\{Y_1^{(r)}\} = \left[A_1^r A_2^r\right]\begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix}$$

$$\{Y_2^{(r)}\} = \left[A_3^r A_4^r\right]\begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix}$$

and finally the interpolated solutions would satisfy the boundary conditions exactly. $\left[\overline{A}^r\right]$ and $\left[\overline{B}^r\right]$ are the modified weighting coefficient matrix and the coefficient matrix for the non-homogenous terms, respectively.

$$\left(\sum_{r=0}^{m}\alpha_r[\tilde{\mathbf{A}}^{(m-r)}]\right)\{\mathbf{Y}_1\} = \{\mathbf{f}\} - \left(\sum_{r=0}^{m}\alpha_r[\tilde{\mathbf{B}}^{(m-r)}]\right)\{\boldsymbol{\beta}\}$$

so, Y_1 can be solved by the above equation.

C. Shu ***et. al.*** **[9]** used the Lagrange interpolating functions and derived the following recurrence formulae.

$$c_{ij}^{1} = \frac{\prod(x_i)}{(x_i - x_j).\prod(x_j)} \text{ Where i,j=1,…….,n and } j \neq 1$$

$$c_{ij}^{(k-1)} = k\left[c_{ii}^{(k-1)}.c_{ij}^{1} - \frac{a_{ij}^{k-1}}{x_i - x_j} \right] \qquad \text{where } 2 \le k \le N-1$$

$$c_{ii}^{m} = -\sum_{\substack{i=1 \\ j\neq 1}}^{N} c_{ij}^{(m)} \text{ where m=1,……..,N-1}$$

Where

$$\prod(x_i) = \prod_{\substack{i=1 \\ j\neq 1}}^{N} (x_i - x_j)$$

The above relations are not restricted to the choice of sampling points. Also calculation of weighting coefficients by these formulae contains a substantial reduction in numerical computations.

For choosing the sampling point there are two approaches, in first when equally spaced points are considered then we use

$x_i = \frac{i-1}{N-1}(b-a)$ For a domain specified by [a, b] and discretized by N points.

And for the unequally spaced sampling point we use

$$x_i = \frac{1}{2}\left(1 - \cos\frac{2i-1}{2N}\pi\right)(b-a)$$

G. Karami ***et. al.*** **[10]** gave the new methodology for the free vibration analysis of the plate. As one troublesome point in the original application of DQM to structural problems has been its difficulty in imposing the boundary conditions for partial differential equations with multiple boundary conditions. This was due to the reason that the linear algebraic equations resulting from the application of DQM became redundant after imposition of the boundary conditions. In order to overcome this difficulty, three different methodologies were introduced by researchers. One of them was the δ technology. But the results by this technique are not equally successful for the entire

structural problem if the δ is not assumed very small. But if the δ is assumed very small then the result for the problem may oscillate. To overcome this problem some researchers implemented the boundary conditions in a different manner from the δ method by building them into the weighting coefficients for those cases that have not mixed derivatives. This method is limited to certain types of boundary conditions and for general cases such as the free edge or curved edge boundary conditions would fail. Thereafter a new methodology is introduced followed by analogous DQ formulation of plate problems generally and for different classes of boundary conditions. In that, the irregular domain is converted to the rectangular domain. This process is identical to the finite element shape function formulation. The coordinate transformation law would be employed is:

$$X = \sum_{i=1}^{N_s} X_i \psi_i(\xi, \eta)$$

$$Y = \sum_{i=1}^{N_s} Y_i \psi_i(\xi, \eta)$$

Where X and Y are co-ordinate domain in the actual domain, $\psi_i(\xi, \eta)$ is the shape function and N_s are number of grid point on each boundary of the domain.

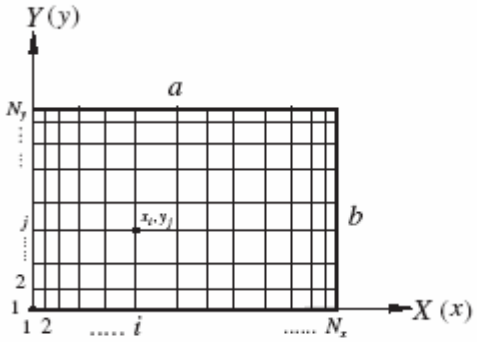


Fig 2.1 Grid points and the rectangular plate coordinate system

Hongzhi Zhong *et. al*. [11] applied Differential Quadrature method to analyze free vibration of simply supported Timoshenko beams with immovable ends. The Differential Quadrature method is employed to solve the non-linear differential equations.

The differential equations for the problem are given by:

In the w direction:

$$-\rho A w_{11} + kGA(w_{xx} - \theta_x) + EA\frac{\partial}{\partial x}[(u_x + 1/2w^2{}_x)w_x] + EI\frac{\partial}{\partial x}[\theta^2{}_x(-w_x + 2w^3{}_x)]$$

$$-1/3kGA\frac{\partial}{\partial x}[w^3{}_x + 3(w_x - \theta)w^2{}_x] = 0$$

In the θ-direction as:

$$-\rho I\theta_{11} + EI\theta_{xx} + EI\frac{\partial}{\partial x}[(-w^2{}_x + w^2{}_x)\theta_x] - 1/3kGAw^3{}_x + kGA(w_x - \theta) = 0$$

In the u-direction as:

$$EA\frac{\partial}{\partial x}[(u_x + 1/2w^2{}_x) = 0$$

The boundary conditions, in the w-direction:

$-kGA(w_x - \theta) - EA(u_x + 1/2w^2{}_x)w_x - EI\theta^2{}_x(-w_x + 2w^3{}_x) + 1/3kGA[w^3{}_x + 3(w_x - \theta)w^2{}_x] = 0$

Or w = 0 at x = 0, L.

In the θ-direction $-EI\theta_x - EI[(-w^2{}_x + w^4{}_x)\theta_x] = 0$ or $\theta = 0$ at x= 0, L

In the u-direction: $-EA(u_x + 1/2w^2{}_x) = 0$ or u=0 at x=0, L

The results obtained for the above non-linear partial differential equations shows excellent convergence characteristics of DQM.

Y.L. Kuo *et. al* [12] gave three common refinement methods of achieving more accurate finite element solutions are to increase the number of elements, to employ higher-degree interpolation functions and to implement adaptive mesh by moving the nodes but maintaining the same number of elements as well as the degree of interpolation functions. In this paper, these refinement methods are applied to a planar high-speed four-bar mechanism. Since the selection of refinement methods depends on the demands and requirement of application problems, and the refinement mesh are highly dependent on error indicators, some guidelines are presented, which are based on several error indicators, such as natural frequencies, total energy, strains and global variables.

Furthermore, their efficiencies are demonstrated through implementing cubic and quintic shape functions on the mechanism.

M. Ulker *et. al.* [13] obtained numerical solution to static analysis of one and two-dimensional structures such as beams, circular and rectangular plates having various support and load conditions by the method of Harmonic Differential Quadrature (HDQ). The conventional small deflection theory was used in the study with the governing differential equations transformed into a set of linear algebraic equations by the HDQ formulation. The method of harmonic differential quadrature that proposed a very simple algebraic formula to determine the connections weighting coefficients required by differential quadrature approximation without restricting the choice of mesh grids. The known boundary conditions are easily incorporated in the harmonic differential quadrature as well as the other type differential quadrature. The method presented gave accurate results and was computationally efficient.

The study was done for various structures using the HDQ method for solving the partial differential equation. The governing equation for the thin rectangular plate was given as:

$$\frac{\partial^4 U}{\partial X^4} + 2k^2 \frac{\partial^4 U}{\partial X^2 \partial Y^2} + k^4 \frac{\partial^4 U}{\partial Y^4} = \frac{qa^4}{D}$$

where U is the dimensionless deflection, $X = x/a, Y = y/a$ are the dimensionless coordinates, a and b are the dimensions of the plate as parallel to x-axis and y-axis, $k = a/b$ is ratio of the plate edge length or aspect ratio, u is the transverse displacement of the midsurface of the plate, D denotes the flexural rigidity of plates and its given $D = Eh^3/12(1-v^2)$, ν the Poisson' ratio, E the modulus of elasticity of the plate material, and h is the thickness of plate. The above equation can be given by applying the HDQ as

$$\sum_{k=1}^{N_x} D_{ik} U_{kj} + 2k^2 \sum_{k=1}^{N_x} \sum_{m=1}^{N_y} B_{ik} B_{jm} U_{km} + \sum_{k=1}^{N_y} D_{jk} U_{ik} = \frac{qa^4}{D}$$

for i = 1, 2, ……., N_x and j = 1, 2, ……., N_y.

where N_x and N_y are the number of grid points along the *x*- and *y*- directions respectively, as shown in the Fig.1, and $D_{ik}, D_{jk}, B_{ik}, B_{jm}$ are represent the weighting coefficients of the fourth and second order derivatives along *x*-and *y*-directions for the differential quadrature approximation.

S. Moradi *et. al*. [14] gave the application of the DQM for solving the buckling problem of the composite plate. The DQM uses the basis of the Gauss method in deriving the derivative of a function. It follows that the partial derivative of a function with respect to a space variable can be approximated by a weighted linear combination of function values at some intermediate points in that variable. In order to show the mathematical representation of DQM, consider a function

$$\frac{d^n f(x_i)}{dx^n} = \sum_{k=1}^{N} c_{ik}^n f(x_k) \qquad \text{i= 1,……,N and n= 1,……N-1} \qquad \text{….(2.1)}$$

where $c_{ik}^{(n)}$ are the weighting coefficients of the nth derivative in the domain that is divided by N discrete points. Choose a test function (in the form of polynomials of order n-1), such that the condition of Eq. (2.1) holds for all polynomials up to nÿ1 order. Using a test function of the form.

$$p_k(x) = x^{k-1} \text{ for k= 1,……..,N}$$

And then substituting it in Eq. (2.1), for all discrete points yields.

$$\frac{d^n (x_j^{k-1})}{dx^n} = \sum_{k=1}^{N} c_{ik}^n x_j^{k-1} \qquad i = 1,\ldots\ldots\ldots\ldots,N \text{ and } n = 1,\ldots\ldots N\text{-}1 \qquad \text{…….(2.2)}$$

so from equation 3 it is clear that.

$$\left[c^n\right] = \left[c^1\right] - \left[c^{n-1}\right]$$

The procedure outlined above bears a major problem; the system of Eq. (2.2) becomes ill-conditioned, and consequently, the weighting coefficients obtained by this method become inaccurate as the number of sampling points is increased. The weighting coefficients may be determined explicitly for all discrete points, irrespective of the number of sampling points. Bellman et al. chose the roots of the shifted Legendre

polynomials of degree N, as the sampling points, and derived the following relation for the first order derivative weighting coefficients.

$$c_{ij}^{1} = \frac{P_N^*(x_i)}{(x_i - x_j).P_N^*(x_j)}$$

$$c_{ij}^{1} = \frac{(1 - 2x_i)}{2x_i.(x_i - 1)}$$

Wen Chen A *et. al.* [15] studied that harmonic Differential Quadrature is a new development of the Differential Quadrature (DQ) which has been used successfully to solve a variety of problems. Harmonic Differential Quadrature chooses harmonic functions as its test functions instead of polynomials as in the Differential Quadrature and has been found especially efficient for problems with periodic behaviors]. Chen and Yu pointed out earlier that the weighting coefficient matrices of DQ method possess centrosymmetric and skew centrosymmetric structure if the grid spacing is symmetric. And a new type matrix, called the skew centrosymmetric matrix, was introduced in that paper. In the present study, it is found that the weighting coefficient matrices of HDQ have similar structure to the DQ method if the grid spacing is symmetric. It is known that the properties of centrosymmetric matrices can be used to factorize its determinant and characteristic equations into two smaller size sub-matrices. Therefore, the computational effort of the HDQ method can be further reduced up to 75 per cent by using the properties of the centrosymmetric and skew centrosymmetric matrices. The free vibration of rectangular plate is investigated to show the efficiency of the HDQ method by using the centrosymmetric behavior. For completeness, they discussed briefly the behaviors of both the centrosymmetric and skew centrosymmetric matrices and found that they possess similar properties. Finally, some conclusions were drawn based on the results reported herein.

Madan G. Kittur *et. al*. [16] presented an extension of a structural finite element mesh improvement technique to heat conduction analysis. The mesh improvement concept was original]y presented by Prager in studying tapered, axially loaded bars. It was further

developed by Kittur, et al., who showed that an improved mesh can be obtained by minimizing the trace of the stiffness matrix.

2.2 SUMMARY OF LITERATURE REVIEW

From the extensive literature review discussed above, it is seen that finite element method (FEM) is used as a tool for the analysis of node parameters. Mesh generation play a very important role in the accuracy of the results obtained from finite element method.

The present work is step towards the improvement of accuracy to be done by analyzing the refined mesh.

3.1 INTRODUCTION

The finite element analyst is continually confronted with the decision of selecting a "good" mesh for a given problem. Unfortunately there seems to be no simple mesh optimization method which is generally applicable. Kittur, et. al. [16] proposed a method which provided an improved mesh over a uniform mesh for a broad class of structural problems. The method is based upon the simple procedure of node relocation to minimize the trace of the stiffness matrix. An advantage of the method is that the analyst may select the nodal positions prior to solving the equilibrium equations.

In the present work the method is used for analyzing the t problem.

3.2 FEM ANANLYSIS FOR HEAT TRANSFER PROBLEM

Considering that an infinitely long hollow cylinder with inner radius r_0 and outer radius r_n having the temperatures T_0 and T_n respectively.

The governing equation for the temperature along the radial line is given by:

$$\frac{d}{dr}\left[rk\frac{dT}{dr}\right] = 0 \qquad \text{----------3.1}$$

And the boundary conditions are given by:

$$T = T_0 \text{ at } r = r_0$$

$$T = T_n \text{ at } r = r_n \qquad \text{----------3.2}$$

The solution of equation (3.1) subject to equation (3.2) is:

$$T = T_n + (T_0 - T_n)\frac{\ln\left(\frac{r_n}{r}\right)}{\ln\left(\frac{r_n}{r_0}\right)} \qquad \text{----------3.3}$$

3.2.1 FINITE ELEMENT FORMULATION AND MESH OPTIMIZATION

The figure below shows the finite element model. It consists of a series of annular elements. For element (e) let the inner and outer radii be r_e and r_{e+1}. The entries of the element stiffness matrix are:

$$[k_{ij}^e] = 2\pi k_e \int_{R_e}^{R_{e+1}} R \frac{dN_i^e}{dR} \frac{dN_j^e}{dR} dR \qquad \text{----------3.4}$$

where the element shape functions N_1^e *and* N_2^e are:

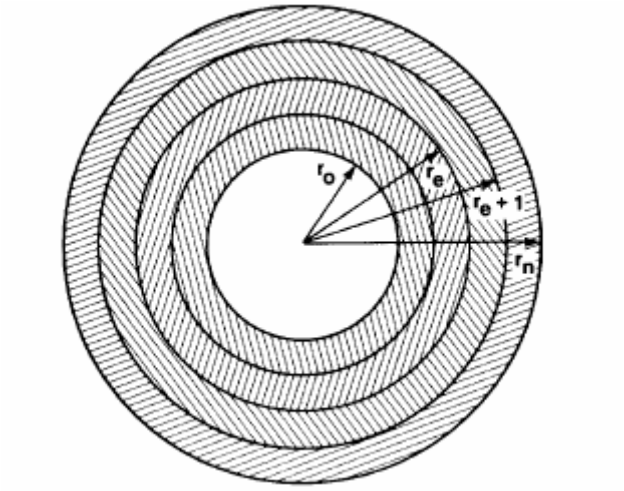


Fig. 3.1 The finite element model

$$N_1^e = \frac{(R_{e+1} - R)}{(R_{e+1} - R_e)}$$

$$N_2^e = \frac{(R - R_e)}{(R_{e+1} - R_e)}$$

So, the element stiffness matrix becomes:

$$\left[k_{ij}^e\right] = S_e \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

Where S_e *is* defined as:

$$S_e = \pi k_e \frac{(R_{e+1} + R_e)}{(R_{e+1} - R_e)}$$

Hence the trace τ of the global stiffness matrix is:

$$\tau = 2\sum_{e=1}^{n} S_e$$

Where n is the number of elements.

The trace may be minimized with respect to the nodal coordinates by setting the partial derivative of τ with respect to r_e equal to zero and solving for the ratio r_{e+1} / r_e.

$$\frac{\partial \tau_e}{\partial r_e} = 0 \qquad \text{----------3.5}$$

Since the derivative of a sum is the sum of the derivatives, and assuming that the conductivity k is uniform thorough the body (i.e. $k_{e-1} = k_e = k_{e+1}$) equation (l0) becomes:

$$\frac{\partial}{\partial r_e}\left[\ldots\ldots + \frac{r_e + r_{e-1}}{r_e - r_{e-1}} + \frac{r_{e+1} + r_e}{r_{e+1} - r_e} + \ldots\ldots \right] = 0 \qquad \text{----------3.6}$$

This simplifies to

$$\frac{-2r_{e-1}}{(r_e - r_{e-1})^2} + \frac{2r_{e+1}}{(r_{e+1} - r_e)^2} = 0 \qquad \text{----------3.7}$$

Inverting (3.7) and factoring

$$\frac{r_e^{\,2}\left(\frac{r_{e+1}}{r_e} - 1\right)^2}{r_{e+1}} = \frac{r_{e-1}^{\,2}\left(\frac{r_e}{r_{e-1}} - 1\right)^2}{r_{e-1}} \qquad \text{----------3.8}$$

Rearranging (3.8)

$$\frac{r_e}{r_{e-1}}\left(\frac{r_{e+1}}{r_e} - 1\right)^2 = \frac{r_{e+1}}{r_e}\left(\frac{r_e}{r_{e-1}} - 1\right)^2 \qquad \text{----------3.9}$$

Let $A=\frac{r_{e+1}}{r_e}$ *and* $B=\frac{r_e}{r_{e-1}}$ then equation (3.9) becomes:

$$B(A-1)^2 = A(B-1)^2$$

Or

$$AB(A-B) - (A-B) = 0 \qquad \text{----------3.10}$$

Solving the equation (3.10) gives:

$$(A-B)(AB-1) = 0 \qquad \text{----------3.11}$$

Equation (3.11) implies that:

Either $(AB-1)=0$ or $(A-B)=0$ if $(AB-1)=0$ then $AB=1$ which means:

$$\frac{r_{e+1}}{r_e}\times\frac{r_e}{r_{e-1}} = 1 \text{ which is not possible.}$$

Now $(A-B)=0$ then $\frac{r_{e+1}}{r_e}=\frac{r_e}{r_{e-1}}$. This is a simplified result which leads to

$\frac{r_{e+1}}{r_e}=\frac{r_e}{r_{e-1}}=\gamma$. Where γ is a constant.

Then:

$$\frac{r_n}{r_0}=\left(\frac{r_n}{r_{n-1}}\right)\left(\frac{r_{n-1}}{r_{n-2}}\right).....\left(\frac{r_e}{r_{e-1}}\right).....\left(\frac{r_1}{r_0}\right) \qquad \text{----------3.12}$$

Using the equation (3.12) we concluded that

$$\frac{r_n}{r_0} = \gamma^n \qquad \text{----------3.13}$$

Where n is the number of elements.

Thus: $r_n = r_0\gamma^n$ and $\gamma = (r_n / r_0)^{1/n}$. Changing n to e, we have: $r_e = r_0\gamma^e$ or

$$r_e = r_0\left(\frac{r_n}{r_0}\right)^{e/n}$$

Which leads to $S_e = \pi k_e \frac{(\gamma+1)}{(\gamma-1)}$. which can be used for the further solution.

RESULT AND DISCUSSIONS CHAPTER-4

In the following section the results presented as per the work of Kittur *et. al* [16] using the methodology given in the previous chapter. The results presented are for temperatures at various nodes when an annular cylinder is subjected to heat transfer using three methods namely FEM, DQM, and HDQM for uniform and refined mesh.

4.1 ILLUSTRATION

Consider an annular cylinder with the following temperatures specified on the boundaries and made from a material with constant thermal conductivity $k = k_e = 1$.

$$T = T_0 = 100^o C \text{ at } r_0 = 20 \text{ mm}$$

$$T = T_n = 0^o C \text{ at } r_n = 50 \text{ mm}$$

The problem is solved by taking both the type of meshing (uniform meshing and refined mesh..

The objective is to determine the internal temperature distribution and compare the results with the analytical solutions.

4.2 RESULTS

The above illustration is solved using the methodology given in the previous chapter. Thereafter the problem is solved using FEM, DQM, and HDQM. A source code is also developed which has capability to solve different problem using FEM, DQM, and HDQM. The flowchart of the source code is shown in fig. 4.1 and the full source code is given in appendix.

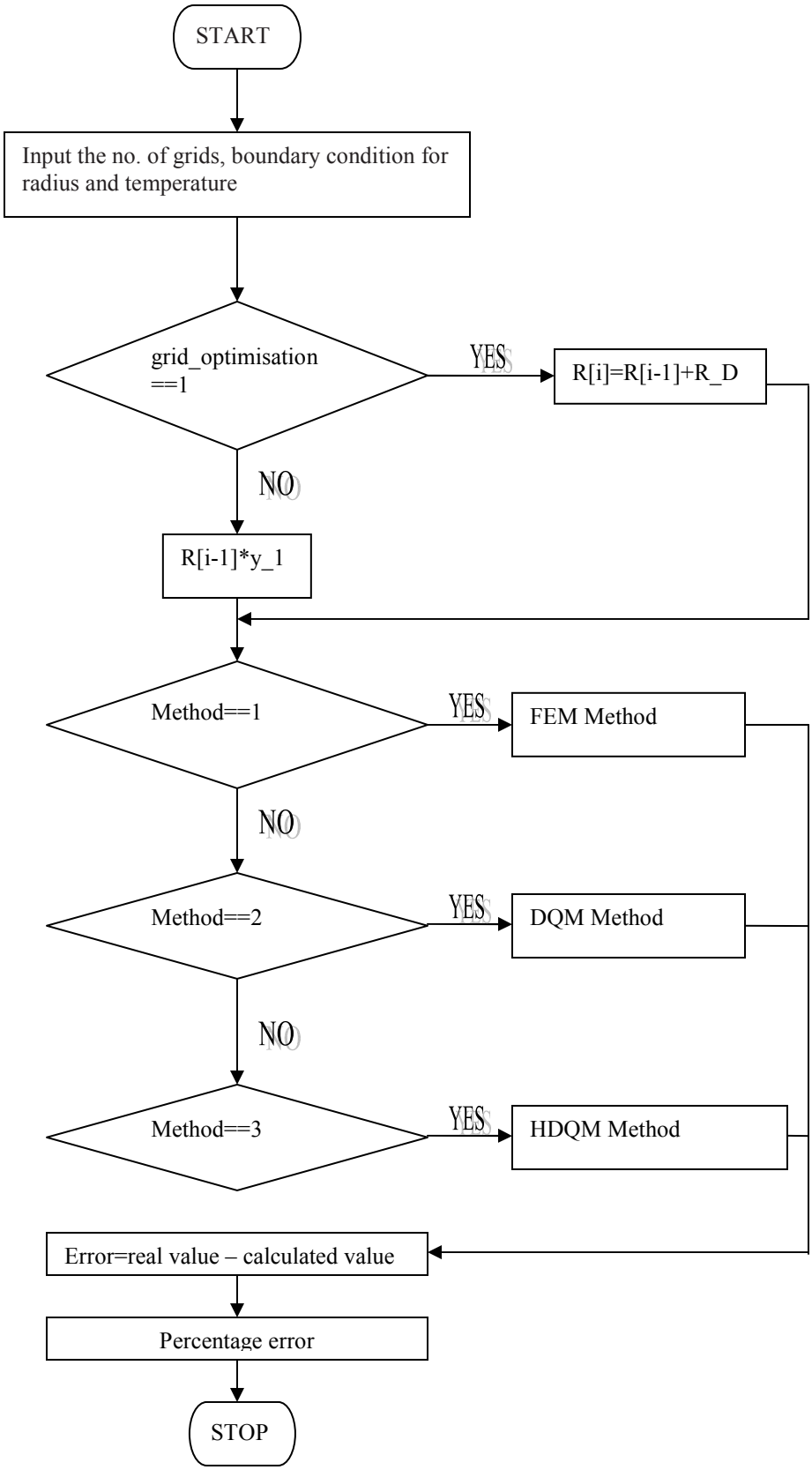


Fig 4.1 Flowchart of the source code

After applying the FEA, DQM and HDQM procedure the values of temperatures on the respective nodes were calculated. The various values of temperatures for the uniform and refined mesh using these three methods have been given in table 4.1 onwards.

Table 4.1 Finite element method using uniform grid.

node	Radius (mm)	Analytical value of temp. (C)	Temperature using FEM (C)	Error	%age error
1.	20	100	100	0	0
2.	27.5	65.2453	65.355	-0.109628	0.168025
3.	35	38.926	39.0247	-0.098786	0.253779
4.	42.5	17.7366	17.7907	-0.0540803	0.304908
5.	50	0	0	0	0

Table 4.2 Finite element method using optimized grid.

node	Radius (mm)	Analytical value of temp. (C)	Temperature using FEM (C)	Error	%age error
1.	20	100	100	0	0
2.	25.1487	74.9999	75	-0.00013629	0.00018172
3.	31.6229	49.9997	50	-0.00027257	0.000545162
4.	39.7637	24.9996	25	-0.00040886	0.0016355
5.	50	0	0	0	0

Table 4.3 Differential Quadrature Method using uniform mesh

node	Radius (mm)	Analytical value of temp.(C)	Temperature using DQM (C)	Error	%age error
1.	20	100	100	0	0
2.	27.5	65.2453	65.2335	0.0118232	0.0181212
3.	35	38.926	38.9261	-0.00011609	0.000298235
4.	42.5	17.7366	17.751	-0.0143873	0.0811162
5.	50	0	0	0	0

Table 4.4 Differential Quadrature Method at optimized grid

node	Radius (mm)	Analytical value of temp.(C)	Temperature using DQM (C)	Error	%age error
1.	20	100	100	0	0
2.	25.1487	74.9999	75.0192	-0.019294	0.0257254
3.	31.6229	49.9997	50.0414	-0.0417117	0.0834239
4.	39.7637	24.9996	25.0745	-0.0748974	0.299594
5.	50	0	0	0	0

Table 4.5 Harmonic differential Quadrature Method using uniform mesh

node	Radius (mm)	Analytical value of temp.(C)	Temperature using HDQM (C)	Error	%age error
1.	20	100	100	0	0
2.	27.5	65.2453	65.3831	-0.137775	0.211164
3.	35	38.926	38.9687	-0.0427414	0.109802
4.	42.5	17.7366	17.7294	0.00716474	0.0403952
5.	50	0	0	0	0

Table 4.6 Harmonic differential Quadrature Method at optimized grid

node	Radius (mm)	Analytical value of temp.(C)	Temperature using HDQM (C)	Error	%age error
1.	20	100	100	0	0
2.	25.1487	74.9999	75.0063	-0.006483	0.0086447
3.	31.6229	49.9997	49.9576	0.04209	0.0841804
4.	39.7637	24.9996	24.9309	0.0686899	0.274764
5.	50	0	0	0	0

After analyzing the above data it is concluded that the result in optimized grid is better for all three methods. Out of these three method the FEM method with optimized grid gives the best result.

To have the better view, the result are shown graphically in fig. 4.2 onwards.

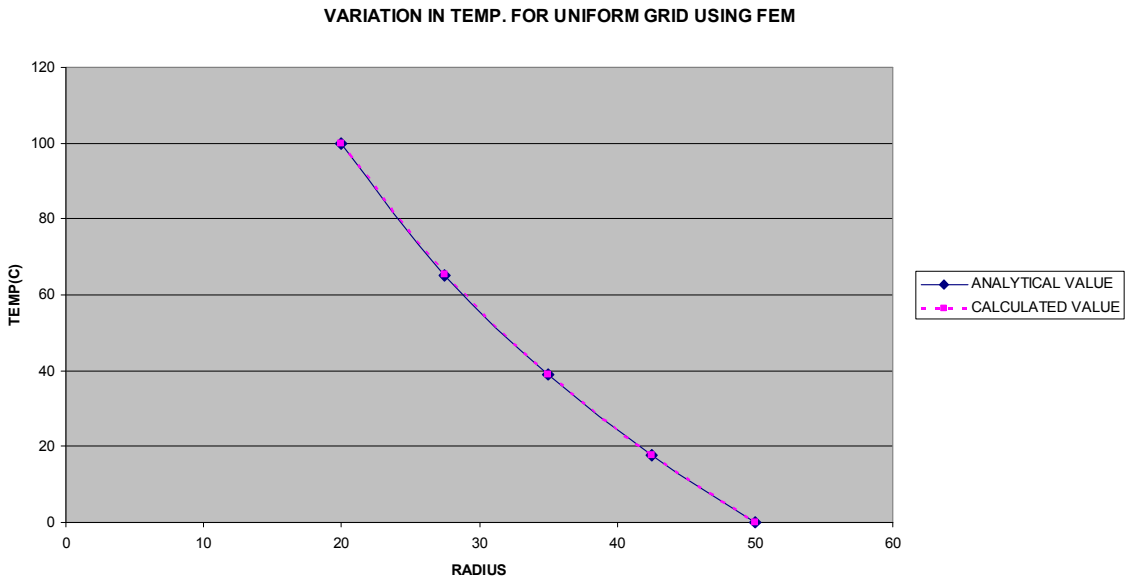


Fig 4.2 Comparison using FEM for uniform grid

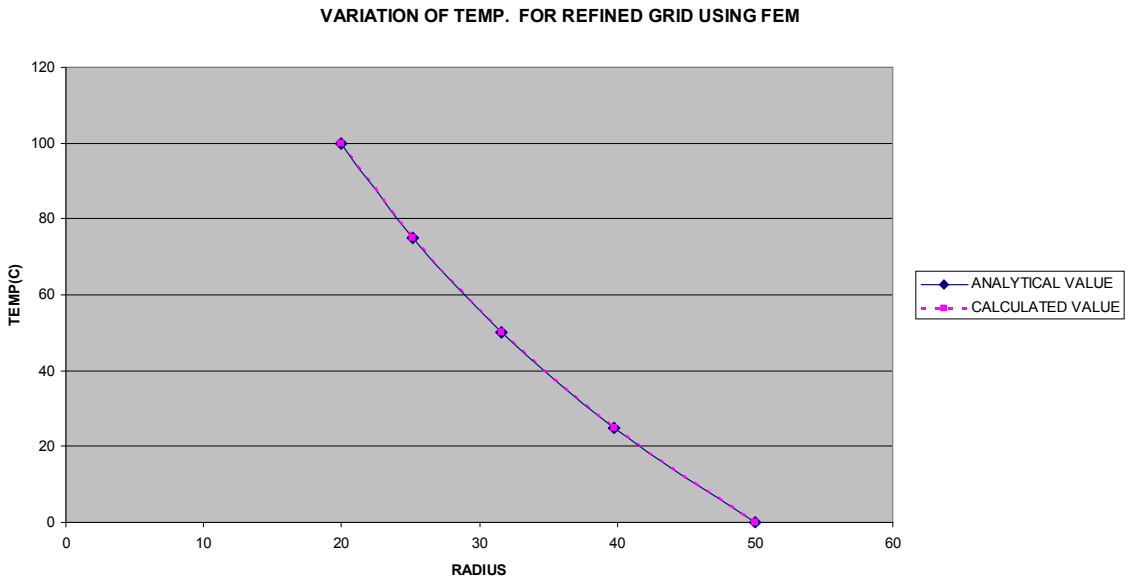


Fig 4.3 Comparison using FEM for refined grid

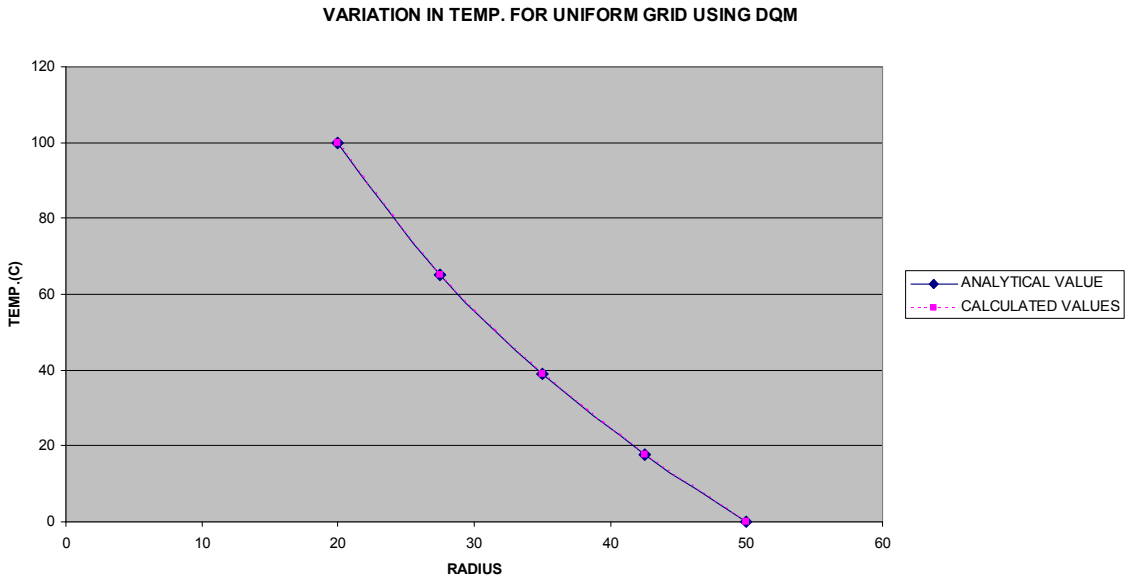


Fig 4.4 Comparison using DQM for refined grid

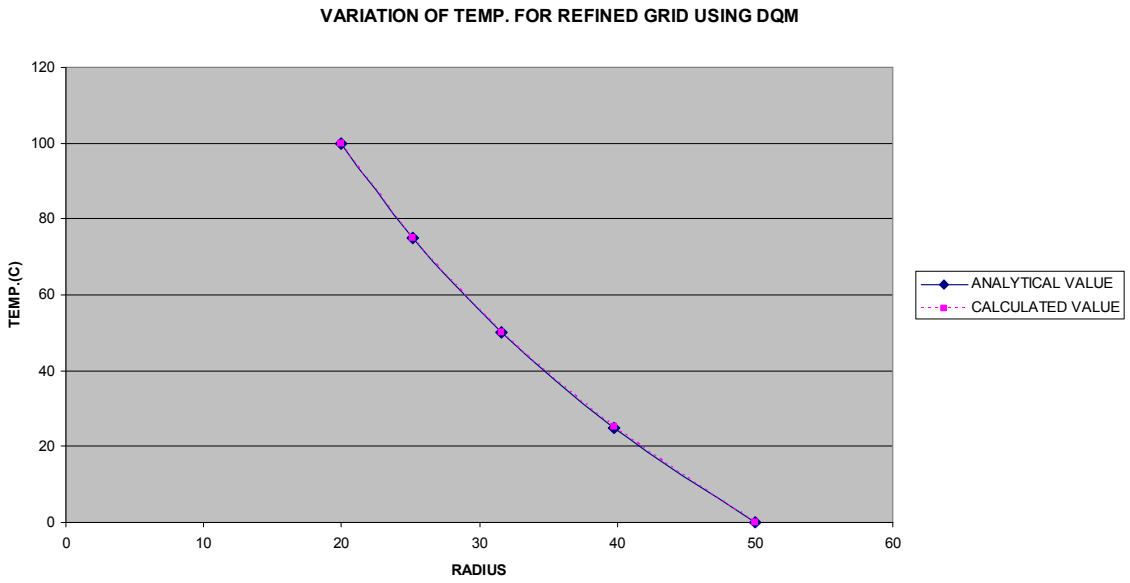


Fig 4.5 Comparison using DQM for refined grid

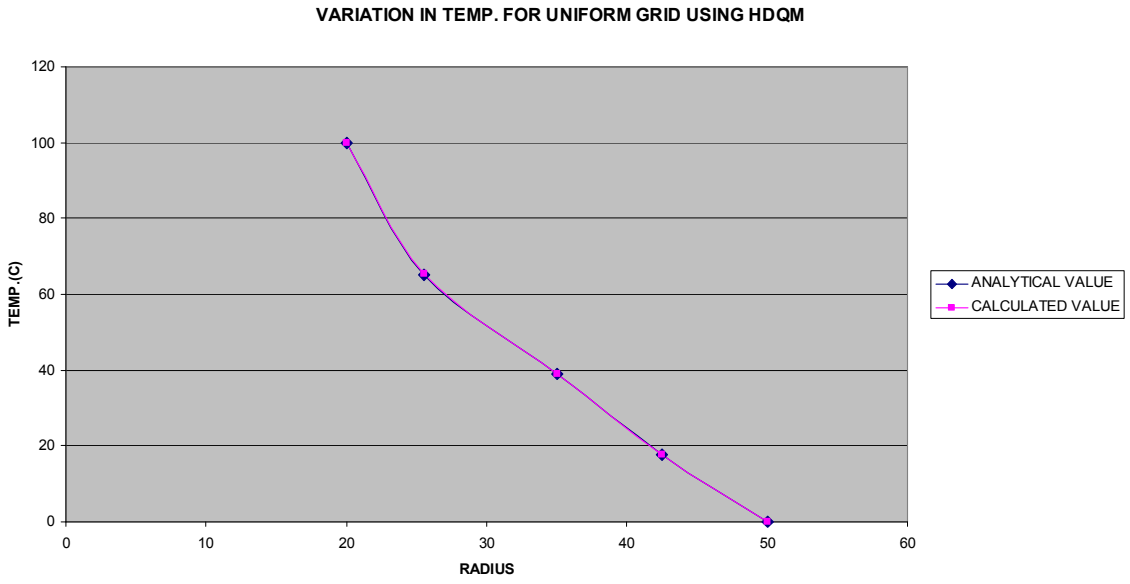


Fig 4.6 Comparison using HDQM for uniform grid

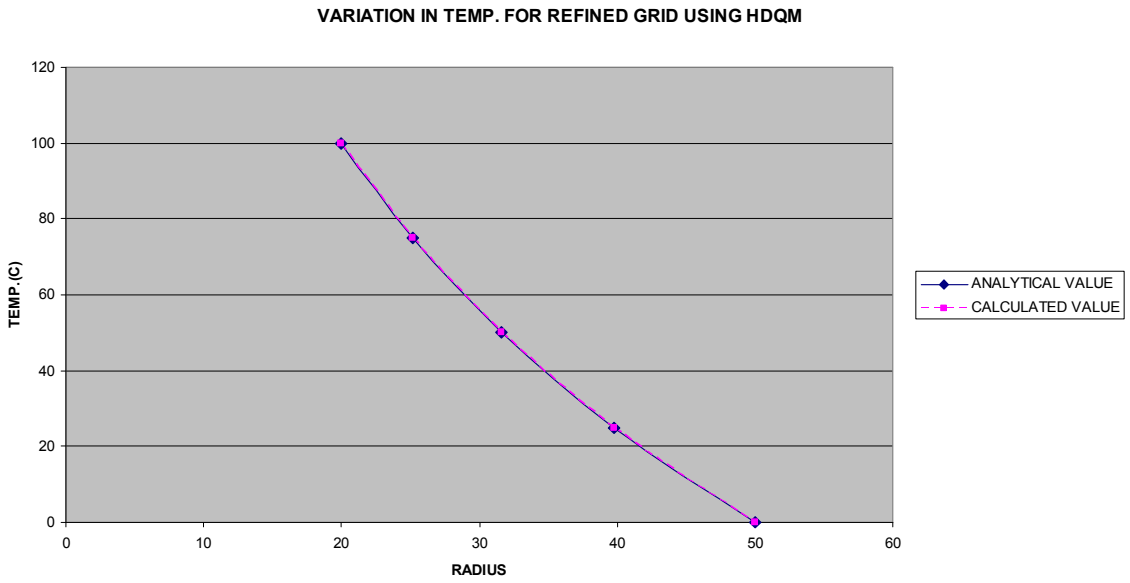


Fig 4.7 Comparison using HDQM for refined grid

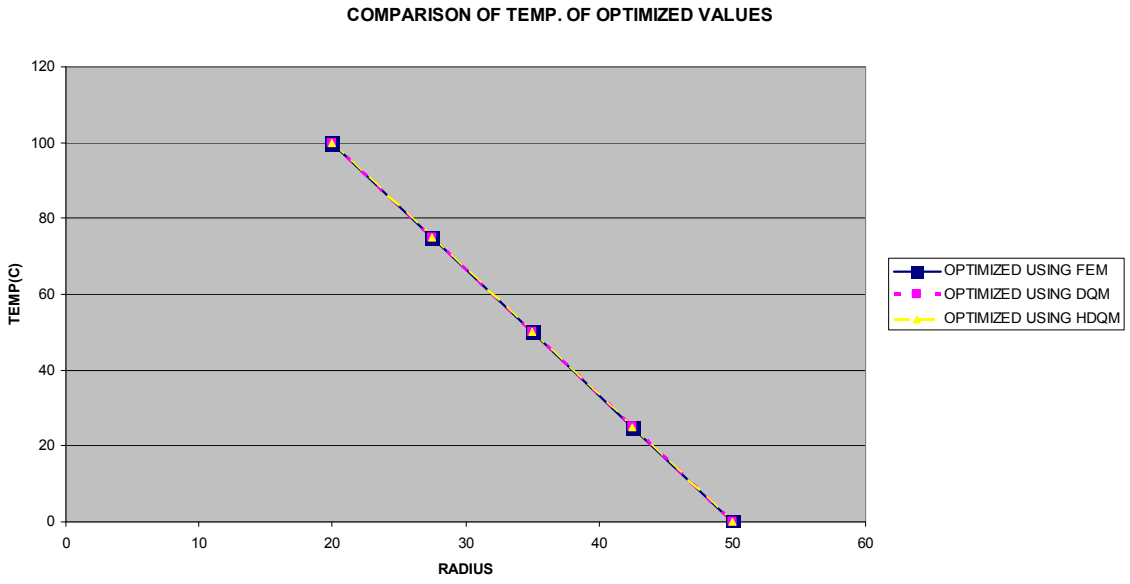


Fig 4.8 Comparison of FEM, DQM, HDQM values for refined grid

CONCLUSIONS CHAPTER-5

As the various methods are used to solve the problem for finding the temperature distribution in an infinitely long cylinder, and in the solutions using these methods it is seen that the error goes on decreasing as the number of nodes goes on increasing along the radial line of cylinder. So by all this it can be concluded that:-

1) Nodal positioning obtained by minimizing the trace of the stiffness matrix leads to an improved mesh over that obtained by uniform positioning of the nodes.

2) Since trace minimization is an a priori method, the mesh may be refined without solving the finite element problem. This makes the minimization procedure computationally inexpensive to perform. The mesh resulting from trace minimization may be used as a starting mesh for other mesh refinement procedures.

3) In the uniform meshing the DQ method is better than the other methods.

4) In case of improved mesh FE method is better than other methods.

REFERENCES

[1] W. E. Carroll And R. M. Barker, " *A Theorem For Optimum Finite-Element Idealizations*, Int. J. Solids Structures, Vol. *9, pp 883-895,*1973.

[2] William Prager, " *A Note on the Optimal Choice of Finite Element Grids*", Computer Methods in Applied Mechanics and Engineering, Vol.6, pp 363-366, 1975.

[3] Peter R. Eiseman "*Adaptive Grid Generation*", Computer Methods in Applied Mechanics and Engineering, Vol. 64, pp 321-376, 1987.

[4] K. M Liew, J.B Han and Z. M Xiao "*Vibration Analysis of Circular Mindlin Plates Using the Differential Quadrature Method*" Journal of Sound and Vibration, Vol. 5, issue 205, pp 617-630, 1997.

[5] Xinwei Wang, Yong-Liang Wang and Rong-Bing Chen, " *Static and Free Vibrational Analysis of Rectangular Plates by the Differential Quadrature Element Method*", Communications in Numerical Methods in Engineering, Vol. 14, pp 1133-1141,1998.

[6] K. M. Liew, T. M. Teo and J.B. Han "*Comparative Accuracy of DQ and HQM Methods For Three-Dimensional Vibration Analysis of Rectangular Plates*" International Journal For Numerical Methods in Engineering, Vol. 45, pp 1831-1848, 1999.

[7] T. Y. Wu and G. R. Liu , " *Application Of Generalized Differential Quadrature Rule to* S*ix-Order Differential Equations*", Communications in Numerical Methods In Engineering, Vol. 16, pp 777-784, 2000.

[8] T.C. Fung, "*Solving Initial Value Problems by Differential Quadrature Method-Part 2: Second and Higher Order Eequations*",International Journal of Numerical Method in Engineering, Vol. 50, pp 1429-1454, 2001.

[9] C. Shu1,Y.W. Chen, H Xue and H. Du , " *Numerical Study of Grid Distribution E Ect On Accuracy Of DQ Analysis of Beams and Plates by Error Estimation of Derivative Approximation*", International Journal For Numerical Methods in Engineering , Vol. 51, pp 159-179, 2001.

[10] G. Karami and P. Malekzadeh, "*Application of a New Differential Quadrature Methodologyfor Free Vibration Analysis Of Plates*", International Journal of Numerical Method in Engineering, pp 847-868, 2003.

[11] Hong Zhi Zhong and Yuhong He, "*A Note on Incorporation of Domain Decomposition into the Differential Quadrature Method*", Communications in Numerical Methods in Engineering, Vol. 19, pp 297-306, 2003.

[12] Y.L. Kuo A, W.L. Cleghorn A, K. Behdinan B, R.G. Fenton A , "*The H–P–R-Refinement Finite Element Analysis of a Planar High-Speed Four-Bar Mechanism,* Mechanism and Machine Theory", Vol. 41, pp 505–524, 2006.

[13] Mehmet Ulker and Omer Civalek, "*Application of Harmonic Differential Quadrature(HDQ) to Deflection and Bending Analysis of Beams and Plates*", Boca Raton: CRC Press..

[14] S. Moradi And F. Taheki, "*Application of Differential Quadrature Method to the Delamination Bucklings of Composite Plates*". Boca Raton: CRC Press.

[15] Wen Chen A, Xinwei Wang B And Tingxiu Zhong, " *The Structure of Weighting Coefficient Matrices of Harmonic Differential Quadrature and its Applications***",** Boca Raton: CRC Press.

[16] Madan G. Kittur And Rona]D L. Huston, "*Improvement Of Finite Element Meshes: Heat Transfer In An Infinite Cylinder*", AVSCOM Technical Report 98-C-021.

.

BIBLIOGRAPHY

[1] S. Chandrakant Desai and F. John Abel, Introduction to Finite Element Method, East-West Edition, 1977.

[2] O.C. Zienkiewicz, The Finite Element Method, Tata McGraw-Hill Edition, London, 1979.

[3] J.N. Reddy, Finite Element Method, McGraw-Hill International Editions, 1993.

[4] S.Rajesekaran, Finite Element Analysis in Engineering Design,Wheeler Publishing,1994.

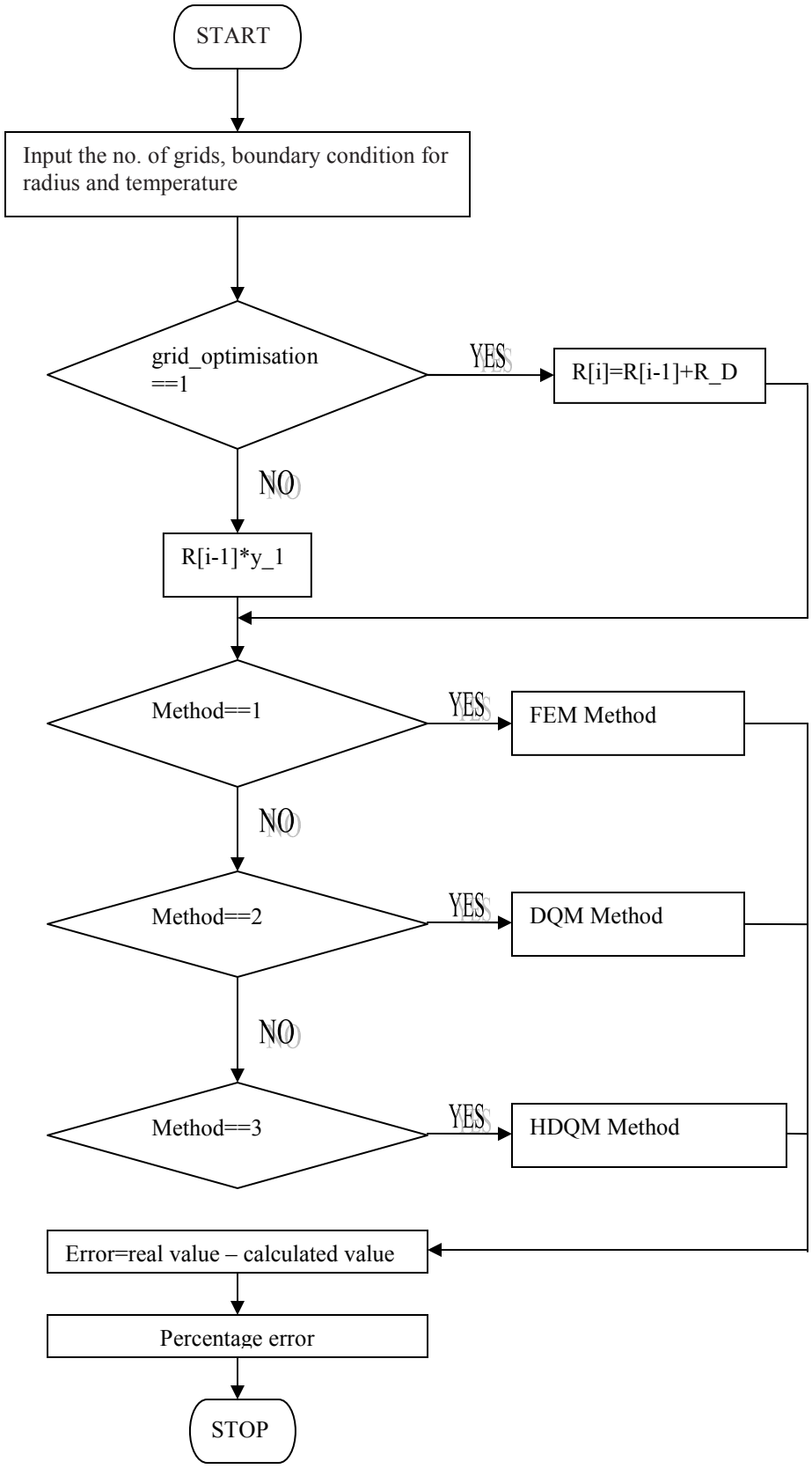
[5] U.C. Jindal, Introduction to Strength of Material, Galgotia Publications Pvt. Ltd., 1998.

[6] Sadhu Singh, Strength of Materials, Khanna Publishers, 1999.

[7] M. James Gere and P. Stephen Timoshenko ,Mechanics of Materials, CBS Publishers and Distributors, 2002.

[8] I.M. Smith and D.V.Griffiths, Programming the Finite Element Method, John Wiley and Sons Ltd., 2004.

[9] R. Tirupathi Chandrupatla and D. Ashok Belegundu, Introduction to Finite Elements in Engineering, Pearson Education , Singapore, 2004.


START
Input the no. of grids, boundary condition for radius and temperature
grid_optimisation ==1
YES
R[i]=R[i-1]+R_D
NO
R[i-1]*y_1
Method==1
YES
FEM Method
NO
Method==2
YES
DQM Method
NO
Method==3
YES
HDQM Method
Error=real value – calculated value
Percentage error
STOP

APPENDIX

```
#include "stdafx.h"
double *R;
int method,grid_optimization,N;
double *T,**A_W,**B_W,**C_W,**D_W,*P;
char heading[40];
double Pi=3.14159265358979323846264338327795;
ifstream fin("input.txt");
ofstream fout("output.txt");
double cotan(double angle)
{
	double A,B,C;
	A = sin(angle);
	B = cos(angle);
	C = B/A;
	return C;
}

//MULTIPLIER DQM

void multiplier_DQM()
{
	int i,j;
	for(i=1;i<=N;i++)
	{
		P[i]=1;
		for(j=1;j<=N;j++)
		{
			if(j!=i)
				P[i]*=(R[i]-R[j]);
		}
	}
	fout<<"\nmultiplier in R direction\n";
	for(i=1;i<=N;i++)
		fout<<P[i]<<endl;
}

//MULTIPLIER HDQM

void multiplier_HDQM()
{
	int i,j;
	double angle,value;
```

```
	for(i=1;i<=N;i++)
	{
		P[i]=1;

		for(j=1;j<=N;j++)
		{
			if(j!=i)
			{
				angle = ((R[i]-R[j])/2)*Pi;
				value = sin(angle);
				P[i]*=value;
			}
		}
	}
	fout<<"\nmultiplier in R direction\n";
	for(i=1;i<=N;i++)
	fout<<P[i]<<"\n";
}

//WEIGHTING CO-EFFICIENT HDQM

void HDQM()
{

	int i,j,k;
	double value,angle;
	fout<<endl<<"WEIGHTING CO-EFFICIENT IN R DIRECTION"<<endl;
		for(i=1;i<=N;i++)
		{
			for(j=1;j<=N;j++)
			{
				if(j!=i)
				{
					angle = ((R[i]-R[j])/2)*Pi;
					value = sin(angle);
					A_W[i][j]=(((Pi/2)*(P[i]))/(value*P[j]));
				}
			}
		}
		for(i=1;i<=N;i++)
		{
			for(j=1;j<=N;j++)
			{
				if(j!=i)
				{
					A_W[i][i]+=A_W[i][j];
```

```
                }
            }
            A_W[i][i]=-A_W[i][i];
        }

        fout<<endl<<"1st order co-efficients along R Direction"<<endl;
        for(i=1;i<=N;i++)
        {
            for(j=1;j<=N;j++)
            {
                fout<<A_W[i][j]<<"  ";

            }
            fout<<"\n";
        }
        for(i=1;i<=N;i++)
        {
            for(j=1;j<=N;j++)
            {
                if(j!=i)
                {
                    angle = ((R[i]-R[j])/2)*Pi;
                    value = cotan(angle);
                    B_W[i][j]=2*(A_W[i][i]*A_W[i][j])-
(Pi*(A_W[i][j])*value);
                }
            }
        }

        for(i=1;i<=N;i++)
        {
            for(j=1;j<=N;j++)
            {
                if(j!=i)
                {
                    B_W[i][i]+=B_W[i][j];
                }
            }
            B_W[i][i]=-B_W[i][i];
        }

        fout<<endl<<"2nd order co-efficients in R Direction"<<endl;
        for(i=1;i<=N;i++)
        {
            for(j=1;j<=N;j++)
            {
```

```
                fout<<B_W[i][j]<<"  ";
        }
        fout<<"\n";
}

for(i=1;i<=N;i++)
{
        for(j=1;j<=N;j++)
        {
                for(k=1;k<=N;k++)
                {
                        if(j!=i)
                        {
                                C_W[i][j]+=A_W[i][k]*B_W[k][j];
                        }
                }
        }
}

for(i=1;i<=N;i++)
{
        for(j=1;j<=N;j++)
        {
                if(j!=i)
                {
                        C_W[i][i]+=C_W[i][j];
                }
        }
        C_W[i][i]=-C_W[i][i];
}
for(i=1;i<=N;i++)
{
        for(j=1;j<=N;j++)
        {
                for(k=1;k<=N;k++)
                {
                        if(j!=i)
                        {
                                D_W[i][j]+=B_W[i][k]*B_W[k][j];
                        }
                }
        }
}
for(i=1;i<=N;i++)
{
        for(j=1;j<=N;j++)
```

```
                {
                        if(j!=i)
                        {
                                D_W[i][i]+=D_W[i][j];
                        }
                }
                D_W[i][i]=-D_W[i][i];
        }
        fout<<endl<<"3rd order co-efficients in R Direction"<<endl;
        for(i=1;i<=N;i++)
        {
                for(j=1;j<=N;j++)
                {
                        fout<<C_W[i][j]<<"  ";

                }
                fout<<"\n";
        }
        fout<<endl<<"4th order co-efficients in R Direction"<<endl;
        for(i=1;i<=N;i++)
        {
                for(j=1;j<=N;j++)
                {
                        fout<<D_W[i][j]<<"  ";

                }
                fout<<"\n";
        }
}

double power(double y,int N)
{
        int i;
        double y_2,y_1;

        y_1=1.000;
        y_2=1;
        while(1)
        {
                for(i=1;i<N;i++)
                {
                        y_2=y_2*y_1;
                }
                y_1=y_1+0.000001;
                if(y_2>=y)
                return y_1;
```

```
                y_2=1.000;
        }
}


//GUASS ELIMINATION APPLIED

void guass_elimination(double **combined,double *RHS,int n)
{
        int k,i,j,heat;
        heat = 2;
        for(k=heat;k<N;k++)
        {
                for(i=k+1;i<=N;i++)
                {
                        double c=combined[i][k]/combined[k][k];
                        for(j=k;j<=N;j++)
                        {
                                combined[i][j]=combined[i][j]-c*combined[k][j];

                        }
                        RHS[i]=RHS[i]-c*RHS[k];
                }
        }
        RHS[1] = T[1];
        RHS[N]=  T[N];
        for(k=N-1;k>=heat;k--)
        {
                for(j=k+1;j<=N;j++)
                {
                        RHS[k]=RHS[k]-combined[k][j]*RHS[j];
                }

                RHS[k]=RHS[k]/combined[k][k];
        }
}

//WEIGHTING CO-EFFICIENT DQM

void DQM()
{
                int i,j;
                fout<<endl<<"WEIGHTING CO-EFFICIENT IN R
DIRECTION"<<endl;
                for(i=1;i<=N;i++)
                {
```

```
		for(j=1;j<=N;j++)
		{
			if(j!=i)
			{
				A_W[i][j]=((P[i])/((R[i]-R[j])*P[j]));
			}
		}
	}

	for(i=1;i<=N;i++)
	{
		for(j=1;j<=N;j++)
		{
			if(j!=i)
			{
				A_W[i][i]+=A_W[i][j];
			}
		}
		A_W[i][i]=-A_W[i][i];
	}
	fout<<endl<<"1st order co-efficients along R Direction"<<endl;
	for(i=1;i<=N;i++)
	{
		for(j=1;j<=N;j++)
		{
			fout<<A_W[i][j]<<"  ";

		}
		fout<<"\n";
	}

	for(i=1;i<=N;i++)
	{
		for(j=1;j<=N;j++)
		{
			if(j!=i)
			{
			B_W[i][j]=2*(A_W[i][i]*A_W[i][j])-2*((A_W[i][j])/(R[i]-
R[j]));
			}
		}
	}

	for(i=1;i<=N;i++)
	{
		for(j=1;j<=N;j++)
```

```
                {
                        if(j!=i)
                        {
                                B_W[i][i]+=B_W[i][j];
                        }
                }
                B_W[i][i]=-B_W[i][i];
        }

        fout<<endl<<"2nd order co-efficients in R Direction"<<endl;

        for(i=1;i<=N;i++)
        {
                for(j=1;j<=N;j++)
                {
                        fout<<B_W[i][j]<<"  ";

                }
                fout<<"\n";
        }
    for(i=1;i<=N;i++)
                {
                        for(j=1;j<=N;j++)
                        {
                                if(j!=i)
                                {
                                   C_W[i][j]=3*(B_W[i][i]*A_W[i][j])-
3*((B_W[i][j])/(R[i]-R[j]));
                                }
                        }
                }

                for(i=1;i<=N;i++)
                {
                        for(j=1;j<=N;j++)
                        {
                                if(j!=i)
                                {
                                        C_W[i][i]+=C_W[i][j];
                                }
                        }
                        C_W[i][i]=-C_W[i][i];
                }
        for(i=1;i<=N;i++)
                {
                        for(j=1;j<=N;j++)
```

```
                        {
                                if(j!=i)
                                {
                                  D_W[i][j]=4*(C_W[i][i]*A_W[i][j])-
4*((C_W[i][j])/(R[i]-R[j]));
                                }
                        }
                }


                for(i=1;i<=N;i++)
                {
                        for(j=1;j<=N;j++)
                        {
                                if(j!=i)
                                {
                                        D_W[i][i]+=D_W[i][j];
                                }
                        }
                        D_W[i][i]=-D_W[i][i];
                }
        fout<<endl<<"3rd order co-efficients in R Direction"<<endl;

                for(i=1;i<=N;i++)
                {
                        for(j=1;j<=N;j++)
                        {
                                fout<<C_W[i][j]<<"  ";

                        }
                        fout<<"\n";
                }
                fout<<endl<<"4th order co-efficients in R Direction"<<endl;
                for(i=1;i<=N;i++)
                {
                        for(j=1;j<=N;j++)
                        {
                                fout<<D_W[i][j]<<"  ";

                        }
                        fout<<"\n";
                }
}
void percentage_error(double *error,double *real_T)
{
        double *percent_error;
```

```
        percent_error = new double [N];
        for(int i=1;i<=N;i++)
        {
                percent_error[i]=0;
        }

for(i=1;i<=N;i++)
        {
                if(real_T[i]==0)
                {
                        real_T[i]=1;
                }
                percent_error[i]=(error[i]/real_T[i])*100;
                if(percent_error[i]<0)
                        percent_error[i]=-percent_error[i];
                fout<<"percentage error at "<<i<<"  "<<percent_error[i]<<endl;
        }
        double sum =0;
        for(i=1;i<=N;i++)
                sum = sum + percent_error[i];
        sum=sum/N;
        fout<<"% average_error  =  "<<sum<<endl;
}
void main()
{
        int i,j;
        cout<<"enter the no of grid points"<<endl;
        cin>>N;
        double R_D,C=1;
        R=new double[N];
        for(i=1;i<=N;i++)
        R[i]=0;

        fin>>heading;
        fin>>R[1];
        fin>>heading;
        fin>>R[N];
        fout<<"inside radius  = "<<R[1]<<endl;
        fout<<"outside radius = "<<R[N]<<endl;
        fout<<"\n";
        cout<<"press 1 for using uniform grid"<<endl;
        cout<<"press 2 for using optimized grid"<<endl;
        cin>>grid_optimization;
        double y = R[N]/R[1];
        double y_1 = power(y,N);
        if(grid_optimization ==2)
```

```
	{
		for(i=2;i<N;i++)
		{
			R[i]=R[i-1]*y_1;
		//	fout<<R[i]<<endl;
		}
	}
	else
	{
		R_D = (R[N]-R[1])/(N-1);
		for(i=2;i<N;i++)
		{
			R[i]=R[i-1]+R_D;
		//	printf("\n%lf\n",R[i]);
		}
	}
//fout<<"radius at different nodes"<<endl;
for(i=1;i<=N;i++)
	fout<<"Radius at "<<i<<" node ="<<R[i]<<endl;
	fout<<"\n";
	T = new double[N];
	for(i=1;i<=N;i++)
	T[i]=0;
						//boundary conditions
	fin>>heading;
	fin>>T[1];
	fin>>heading;
	fin>>T[N];
	fout<<"inside temperature  = "<<T[1]<<endl;
	fout<<"outside temperature = "<<T[N]<<endl;
	cout<<"press 1 for FEM"<<endl;
	cout<<"press 2 for DQM"<<endl;
	cout<<"press 3 for HDQM"<<endl;
	cin>>method;
			//combined matrix
	double **combined,*S,*RHS;
	combined = new double*[N];
	for(i=1;i<=N;i++)
	combined[i]= new double[N];
	for(i=1;i<=N;i++)
		for(j=1;j<=N;j++)
			combined[i][j]=0;

	RHS=new double[N];
	for(i=1;i<N;i++)
	{
```

```
        RHS[i]=0;
}
S=new double [N-1];
y=y_1;
//FEM applied
if(method==1)
{
        for(i=1;i<N;i++)
        {
                if(grid_optimization==2)
                {
                        S[i]= Pi*C*((y+1)/(y-1));
                }

                else
                {
                        S[i]=Pi*C*((R[i+1]+R[i])/(R_D));
                }
                combined[i][i]=combined[i][i]+S[i];
                combined[i+1][i]=combined[i][i+1]=-S[i];
                combined[i+1][i+1]=S[i];
        }
        for(j=1;j<=N;j++)
        {
                for(int l=1;l<=N;l++)

                        printf(" %lf  ",combined[j][l]);

                printf("\n");
        }

}
//weighting coeffient defined
A_W = new double*[N];
B_W = new double*[N];
C_W = new double*[N];
D_W = new double*[N];
for(i=1;i<=N;i++)
{
        A_W[i] = new double[N];
        B_W[i] = new double[N];
        C_W[i] = new double[N];
        D_W[i] = new double[N];
}
for(i=1;i<=N;i++)
for(j=1;j<=N;j++)
```

```
        {
                A_W[i][j] = 0;
                B_W[i][j] = 0;
                C_W[i][j] = 0;
                D_W[i][j] = 0;
        }

//multiplier defined
P= new double [N];
for(i=1;i<=N;i++)
        P[i]=0;
//DQM applied
if(method==2)
{
        int l;
for(j=1;j<=N;j++)
        {
                R[j]=R[j]/R[N];
                fout<<R[j]<<endl;
        }
        multiplier_DQM();
        DQM();
        for(j=1;j<=N;j++)
        fout<<R[j]<<endl;
        for(j=1;j<=N;j++)
        {
                for(l=1;l<=N;l++)

                        combined[j][l]=R[l]*B_W[j][l]-A_W[j][l];
        }
}
fout<<"\n";
//HDQM Applied
if(method==3)
{
        int l;
        for(j=1;j<=N;j++)
        {
                R[j]=R[j]/R[N];
                out<<R[j]<<endl;
        }
        multiplier_HDQM();
        HDQM();
        for(j=1;j<=N;j++)
        {
                R[j]=R[j]/R[N];
```

```
                //fout<<R[j]<<endl;
        }
        for(j=1;j<=N;j++)
        {
                for(l=1;l<=N;l++)
                combined[j][l]=R[l]*B_W[j][l]-A_W[j][l];
        }
}
for(i=2;i<=N;i++)
        RHS[i]=-combined[i][1]*T[1];
//guass_elimination applied
guass_elimination(combined,RHS,N);
for(i=1;i<=N;i++)
{
        T[i] = RHS[i];
}

//analytical results
double *real_T,a,b,c;
real_T = new double [N];
real_T[1] = T[1];
real_T[N] = T[N];
for(i=2;i<N;i++)
{
        a = real_T[1]-real_T[N];
        b = log(R[N]/R[i]);
        c = log(R[N]/R[1]);
        real_T[i] = real_T[N]+a*(b/c);
}
for(i=1;i<=N;i++)
{
        cout<<"analytical value of temperature["<<i<<"] = "<<real_T[i]<<endl;
        cout<<"\n";
        fout<<"analytical value of temperature["<<i<<"] = "<<real_T[i]<<endl;
        fout<<"\n";
}
fout<<"\n";
cout<<"\n";
for(i=1;i<=N;i++)
{
        cout<<"calculated value of temperature["<<i<<"] = "<<T[i]<<endl;
        cout<<"\n";
        fout<<"calculated value of temperature["<<i<<"] = "<<T[i]<<endl;
        fout<<"\n";
}
fout<<"\n";
```

```
        //error in calculated values
        double *error;
        error = new double [N];
        for(i=1;i<=N;i++)
        {
                error[i] = (real_T[i]-T[i]);
//              fout<<"error["<<i<<"] = "<<error[i]<<endl;
        //      fout<<"\n";
                //"
                cout<<"error["<<i<<"] = "<<error[i]<<endl;
                cout<<"\n";
        }
        percentage_error(error,real_T);
}
```