BOUNDARY INTEGRAL ELEMENT ANALYSIS OF STEADY-STATE HEAT CONDUCTION PROBLEMS

by

Cemal TUNALI

B.S. in M.E., Boğaziçi University, 1982

Submitted to the Institute for Graduate Studies in Science and Engineering in partial fulfillment of the requirements for the degree of

Master of Science

in

Mechanical Engineering

Boğaziçi University

1984

BOUNDARY INTEGRAL ELEMENT ANALYSIS

OF

STEADY-STATE HEAT CONDUCTION PROBLEMS

APPROVED BY:

Prof.Dr. Akın TEZEL (Thesis Supervisor)	[illegible]
Doç.Dr. Fahir BORAK	[illegible]
Dr. Vahan KALENDEROĞLU	[illegible]

DATE OF APPROVAL:

ACKNOWLEDGEMENTS

I would like to express my gratitude to my thesis supervisor, Prof.Dr. A. Tezel, for his advice and tactful criticisms that have been valuable in all parts of this work.

Special thanks are due to Doç.Dr. F. Borak and Dr. V. Kalenderoğlu for their interest in this work.

My sincere appreciation goes to A. Özmen for his invaluable help.

Special thanks must finally go to M. Kalenderoğlu for her patience and diligence in typing.

Cemal TUNALI

ABSTRACT

In this study, the boundary integral element method is used for analysis of steady-state heat conduction problems. The method is general for two-dimensional regions with arbitrary boundary shapes. The development is generalized to include the first, second, and third kind of boundary conditions as well as nonlinear conditions. A variety of problems are analyzed with this method and their solutions are compared to those obtained analytically. A comparison between the present method and the finite difference predictions is also made. Moreover, two-dimensional regions with three kind of boundary conditions, irregular shaped boundaries and regions with more than one surface are used to illustrate the versatility of the technique as a computational procedure.

ÖZET

Bu çalışmada, sabit rejimde ısı iletimi problemlerini çözmek için sınır integral elemanları yöntemi kullanılmıştır. Bu yöntem alelade sınır şekillerine sahip iki boyutlu bölgeler için geneldir. Yöntem birinci, ikinci ve üçüncü tür sınır koşulları ve aynı zamanda lineer olmayan sınır koşullarında genelleştirilmiştir. Çeşitli problemler bu metodla analiz edilmiş ve analitik çözümlerle karşılaştırılmıştır. Aynı zamanda, bu yöntemle elde edilen çözümlerle sonlu farklar çözümleri arasındaki karşılaştırma da yapılmıştır. Üç çeşit sınır koşullarına, düzensiz şekilli sınırlara ve birden fazla yüzeye sahip iki boyutlu problemler, yöntemin çok yönlülüğünü örneklemek için kullanılmıştır.

TABLE OF CONTENTS

LIST OF FIGURES

List of Figures continued...

LIST OF TABLES

LIST OF SYMBOLS

u	Temperature (potential), (^{o}C or ^{o}K)
q	Flux, ($^{o}C/m$ or $^{o}K/m$)
$\tilde{n}$	Normal vector to surface
∇^2	Laplacian operator in two dimensions
D	Domain
S	Surface
q'''	Volumetric heat generation (W/m^3)
K	Thermal conductivity ($W/m.^{o}C$ or $W/m.^{o}K$)
p	$= q'''/K$, p. 6
H_o	Convective heat transfer coefficient ($W/m^2.^{o}C$ or $W/m^2.^{o}K$)
u_∞	Ambient temperature
σ	Stefan-Boltzmann constant (5.6697×10^{-8} $W.m^2/^{o}K^4$)
R	Residual
ω	Weighting function
q^*	$= \partial u^*/\partial n$, p. 12
δ	Dirac delta function
L^*	Formal adjoint of the operator L
r	Distance from the point of application of the unit potential to the point under consideration
x,y	General Cartesian coordinates
$\overline{n}$	Number of boundary elements on S
B	$= - \int pu^*dD$, p. 20

List of Symbols continued...

G_{ij}	$= \int_{S_j} u^* dS$, p.19
H_{ij}	$= \int_{S_j} q^* dS$, p. 19
ξ	Local coordinate
ℓ	Length of the boundary element
Z_t	Gauss weighting coefficients
$\overline{m}$	Number of internal elements
ϕ	Shape function
α, β	Numerical coefficients, p. 33

I. INTRODUCTION

Integral methods for formulating governing field equations have been a subject of interest to many investigators for several years. Some exact and approximate solutions for the integral equations, arising in the above mentioned integral methods, were obtained.

A fundamental method employed in the classical potential theory is the use of Green's functions for solving the integral equation associated with the Laplace equation as given by Morse and Feshbach [1]. In spite of the generality of this method, it is limited to those problems having simple geometries. The limitations are due to the mathematical complexity in the construction of the required Green's functions for obtaining the solution to the associated integral equation.

A modified version of the method, which is studied by Jawson [2] and MacMillan [3], is based on the use of Green's functions together with the Green's second formula and has been found to be more practical and less complex. The basic idea of this modified version is to cast the field differential equation into a boundary integral equation. Although the major properties of differential equations were well established by the nineteenth century, the first rigorous investigation of the classical kinds of integral equations was published by Fredholm at 1905. Since then they have been studied intensively, particularly in connection with field theory.

A major contribution to the formal understanding of integral equations has been made more recently by Mikhlin [4-6]. He discusses such equations with both scalar and vector (multidimensional) integrands and, in particular, those with singularities and discontinuties within the range of integration. Despite of the great advances that have been made in the classification and analysis of the properties of integral equations, none of the major authors who deals with applied mathematics appear to have considered the possibility that a general numerical algorithm for solving a wide range of practical problems might be based on the integral equations. The impetus for such a development has been provided by the high-speed digital computers and as a result the boundary integral element method has been developed.

Applications of the boundary integral element method to heat conduction problems have received less attention as compared to those problems in solid mechanics. This is due to the limitation of the boundary integral element method to the problems subject to linear boundary conditions. Certain papers [7-10] have appeared in the literature which show the application of the method to the solution of conduction problems where both the temperature and flux at the boundaries are constant.

In many aspects, the boundary integral element method for solving boundary value problems proves to be advantageous over the conventional numerical methods of finite difference and finite elements. Since the technique uses only the boundary data in the solution, this in turn reduces the size of numerical calculations. In addition, the solution at any interior point is easily obtained with a resolution and without further involvement of the other points. Furthermore, the method does not require any modifications or special handling of points near the domain boundaries, unlike the case of finite differences. This particular feature makes the boundary integral element method well-suited, as it is the case in finite element method, to those problems with irregular shaped boundaries.

In this investigation, the boundary integral element numerical method is modified to be applied as iterative technique. This modification enables the method to solve numerically steady-state heat conduction problems with no restrictions imposed on its boundary conditions. This technique is applicable for two-dimensional problems with nonlinear boundary conditions resulting from radiation at the boundary. Also, the problems with more than one surface, such as the case of hollow cylinder, are investigated.

II. THEORY AND PRINCIPLES

When an engineer constructs a mathematical model of almost any kind of a system, he usually starts by establishing the behaviour of an infinitesimal differential element of it. This establishment is based on assumed relationships between the major variables involved. This leads to a description of the system in the form of a set of differential equations. Once the basic model has been constructed and the properties of the particular differential equation is understood, subsequent efforts are then directed towards obtaining a solution of the equations within the region of interest. The regions are often of very complicated shapes in where various conditions are specified on the boundaries.

The numerical methods most widely used at present deal with the differential mathematical manipulation in one of two ways: EITHER by approximating the differential operators in the equations by simpler, localized algebraic ones valid at a series of nodes within the region OR by representing the region itself by noninfinitesimal (i.e., finite) elements which are assembled to provide an approximation to the real region.

An obvious alternative approach to solve the set of differential equations would be to attempt to integrate them analytically in some way before either proceeding with any discretization scheme or introducing any approximations. We are, of course, attempting to integrate the differential equations to find a solution whatever method we use, but the essence of boundary integral equation techniques is the transformation of the differential equations into equivalent sets of integral ones as the first step in their solution.

However, the numerical methods are predominant over the analytic methods in respect of problem solving ability. It is also a fact that further improvements in computer technology will enhance the improvement and applicability of numerical methods.

2.1 DEFINITION OF THE PROBLEM

In this study, the temperature distribution in simply and multiply connected regions under the influence of steady-state conduction heat transfer with heat generation and constant thermal conductivity is explored.

The governing field equation is shown to be

$$\nabla^2 u + \frac{q'''}{K} = 0 \tag{2.1.1}$$

in the domain of interest where u is the temperature, q'" is the volumetric heat generation and K is the thermal conductivity.

Due to the limited availability of analytic solutions of the above equation for a given set of boundary conditions effort has been spent on utilizing boundary integral element method, which is one of the numerical approaches, to broaden the range of problems which can be solved.

The boundary conditions of concern which can be grouped as follows.

- Boundary conditions of the first kind in which the value of temperature is prescribed at the boundary.
- Boundary condition of the second kind in which the value of flux is prescribed at the boundary.
- Boundary condition of the third kind in which convective heat transfer into a medium at a prescribed temperature occurs at the boundary.
- The nonlinear boundary condition in which the formulation of this kind of boundary condition involves a power of temperature. In our study, the fourth power model of radiation boundary condition will be considered.

2.2 BOUNDARY INTEGRAL FORMULATION

The aim of an approximate solution scheme is to reduce a governing equation (or set of equations) and boundary conditions to a system of algebraic equations. This is usually done by subdividing the continuum into a number of cells or elements and assuming over each of these a known variation of the approximating and weighting functions [21, p.8]. Consider the Poisson's boundary value problem

$$\nabla^2 u_0 + p = 0 \qquad \text{in} \quad D \tag{2.2.1}$$

where u_0 indicates the exact solution. The corresponding boundary conditions are of the following two types:

- Temperature is prescribed on the boundary, i.e.,

$$u_0 = \overline{u} \tag{2.2.2}$$

  where $\overline{u}$ is its value on the boundary S_1.

- Flux is prescribed on the boundary, i.e.,

$$q_0 = \overline{q} \tag{2.2.3}$$

  where,

$$q_0 = \frac{\partial u_0}{\partial n} \tag{2.2.4}$$

which is the normal derivative of the exact solution u_0 and $\overline{q}$ is its value on the boundary S_2.

The total boundary of the domain D is

$$S = S_1 + S_2$$

as shown in Figure 2.2.1.

The exact solution u_0 can be found only for a few and simple cases and, generally, the solution will have to be approximated. This can be done by using a set of known linearly independent functions ψ_i and unknown coefficients γ_i so as to construct the approximating function u of the exact solution u_0. Hence the approximating function u is the linear combination of the linearly independent function ψ_i:

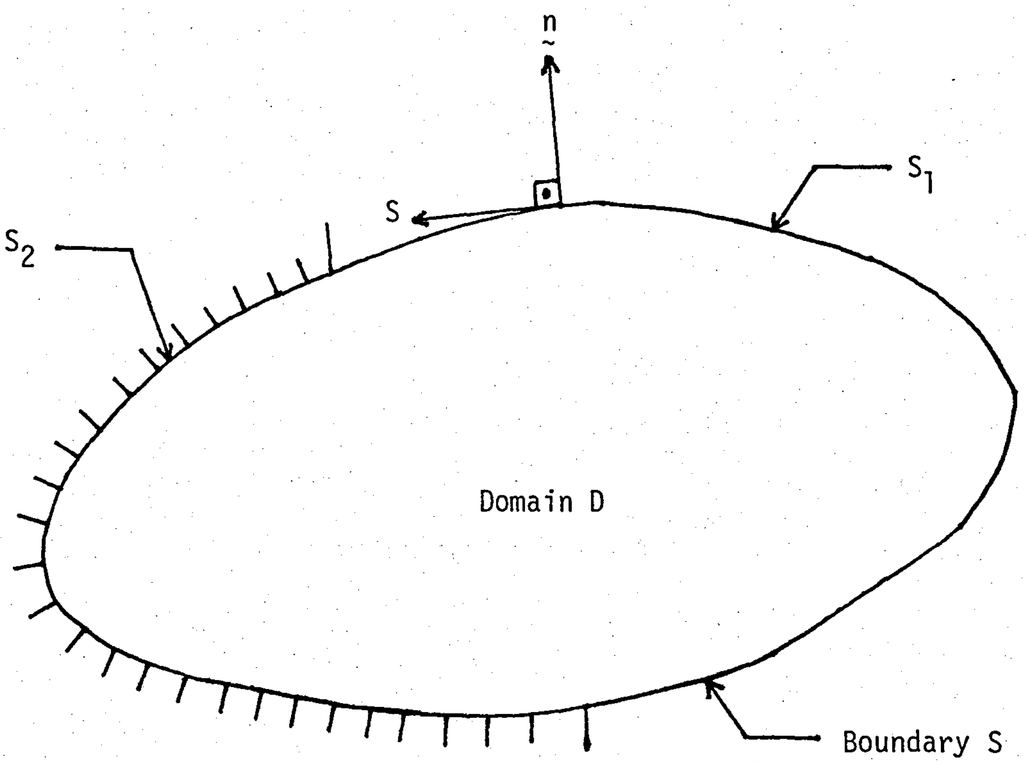


Figure 2.2.1 - Schematic diagram of the domain D and its boundary S.

$$u = \sum_{i=1}^{n} \gamma_i \psi_i \ .$$

Due to its nature, substitution of the approximating function u instead of the exact solution u_0 will not satisfy the Eq. (2.2.1) so that, a residual will be produced subsequently. The procedure is as follows:

$$\nabla^2 u + p \neq 0 \ .$$

It is seen that this yields an inequality. However, the approximating function u is taken to be satisfying the boundary conditions, i.e.,

$$u - \bar{u} = 0 \qquad \text{on } S_1 ,$$

and

$$q - \bar{q} = 0 \qquad \text{on } S_2 .$$

Letting

$$\nabla^2 u + p = R \tag{2.2.5}$$

where R is the residual, the inequality is transformed into an equality. We try to minimize this residual. The residual is normalized with respect to properly chosen weighting function ω.

$$(R,\omega)_D = 0 \quad ,$$

or

$$\int_D R\omega dD = 0 \tag{2.2.6}$$

where,

$$R = \nabla^2 u + p$$

so that

$$\int_D (\nabla^2 u + p)\omega dD = 0 . \tag{2.2.7}$$

We are trying to minimize the residual by distributing it all over the domain so as to force it to be zero in an average sense.

The Green's second identity [11, p. 451] is given as follows;

$$\int_D (a\nabla^2 b - b\nabla^2 a)dD = \int_S (a \frac{\partial b}{\partial n} - b \frac{\partial a}{\partial n})dS \quad . \tag{2.2.8}$$

From Eq. (2.2.7), we obtain

$$\int_D (\nabla^2 u)\omega dD = - \int_D p\omega dD \quad . \tag{2.2.9}$$

Applying the Green's second identity to the left hand side of the Eq. (2.2.9), we get

$$\int_D (\nabla^2 u)\omega dD = \int_D u\nabla^2\omega dD + \int_S \omega \frac{\partial u}{\partial n} dS - \int_S u \frac{\partial \omega}{\partial n} dS \ . \qquad (2.2.10)$$

Hence,

$$-\int_D p\omega dD = \int_D u\nabla^2\omega dD + \int_S \omega\frac{\partial u}{\partial n} dS - \int_S u \frac{\partial \omega}{\partial n} dS \qquad (2.2.11)$$

and, rearranging the above equation, we have

$$-\int_D u\nabla^2\omega dD = \int_S \omega \frac{\partial u}{\partial n} dS - \int_S u \frac{\partial \omega}{\partial n} dS + \int_D p\omega dD \ . \qquad (2.2.12)$$

Now, it remains to find and insert the weighting function ω into Eq. (2.2.12). Utilizing the reproducing property of Dirac delta function [12, p. 315]

$$\int_D u\delta dD = u \qquad (2.2.13)$$

we see that there is a possibility of simplifying the left hand side of the Eq. (2.2.12).

Letting

$$L = \nabla^2 \qquad (2.2.14)$$

so that we can write the Poisson's equation as

$$Lu = -p \ . \qquad (2.2.15)$$

Eq. (2.2.9) then takes the form

$$(\omega, Lu)_D = (\omega, -p)_D \ . \qquad (2.2.16)$$

Further manipulation on the left hand side of Eq. (2.2.16) by introducing the adjoint L^* of the operator L results in

$$(\omega,Lu)_D = (u,L^*\omega)_D + \pi \tag{2.2.17}$$

where π denotes the boundary integral terms appearing as surface integrals in Eq. (2.2.12). Since the operator L is formally self adjoint [13, p. 247], we have,

$$L^* = L \quad . \tag{2.2.18}$$

We use the reproducing property of Dirac delta function to simplify the $(u, L^*\omega)_D$ term in the Eq. (2.2.17). Thus, letting

$$\nabla^2\omega = -\delta \quad , \tag{2.2.19}$$

or

$$L^*\omega = -\delta \tag{2.2.20}$$

we obtain from Eq. (2.2.17)

$$(u,\delta)_D = -(\omega, Lu)_D + \pi \quad . \tag{2.2.21}$$

Using Eq. (2.2.13),

$$u = -(\omega,Lu)_D + \pi \tag{2.2.22}$$

we obtain the GREEN's Formula.

We also note that in arriving the boundary integral equation, the weighting function ω is defined as the solution of Eq. (2.2.19). This kind of weighting function is known as the unit singular solution [14, p. 58] or fundamental solution denoted by u* for an infinite domain and the associated flux is

$$q^* = \frac{\partial u^*}{\partial n} \quad .$$

Thus, choosing u* as the weighting function ω enables us to simplify Eq. (2.2.12) as

$$u = \int_S u^* q dS - \int_S u q^* dS + \int_D p u^* dD \ . \qquad (2.2.23)$$

Comparing with Eq. (2.2.12), it may be seen that Eq. (2.2.23) is the proper form of Green's formula. In this formula, it should be noted that the function u* is a function of two points: the 'source' point $\underset{\sim}{x}_i$ at which we have the singularity of delta function; and the 'observation' point $\underset{\sim}{x}$ which is the variable involved in our differential equation. The fundamental solution is a function only of the distance between the 'source' point A and the 'observation' point B as shown in Figure 2.2.2. We denote this distance by

$$r = |\underset{\sim}{x} - \underset{\sim}{x}_i| \ . \qquad (2.2.24)$$

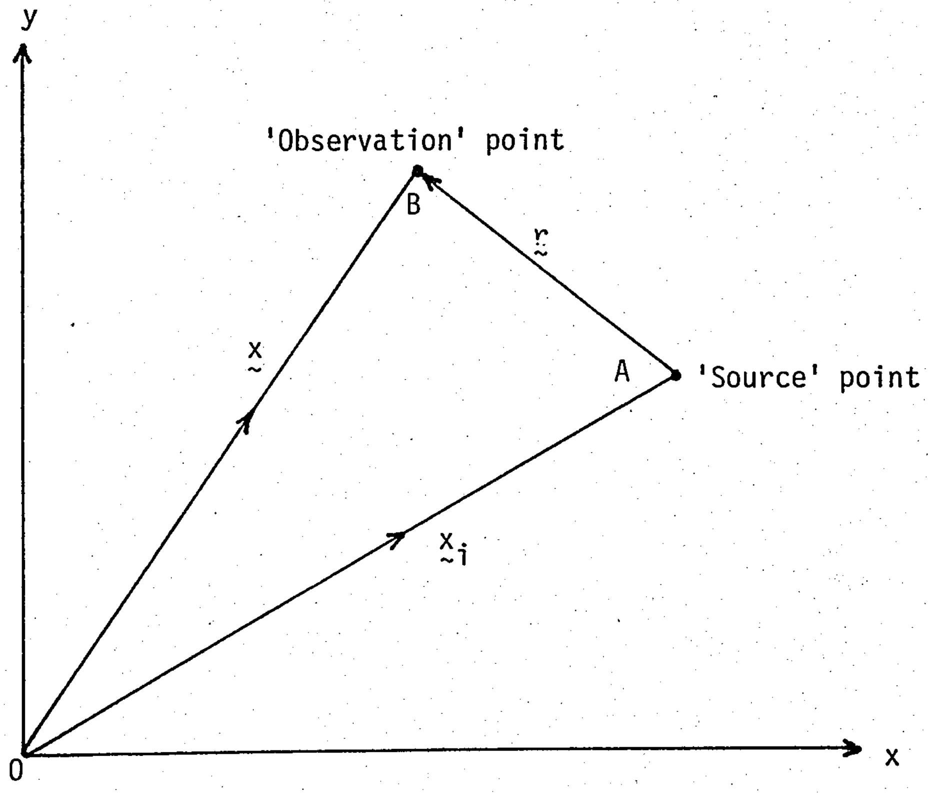


Figure 2.2.2 - Schematic diagram for definition of the fundamental solution.

The fundamental solution u* is found to be

$$u^* = \frac{1}{2\pi} \ln(1/r) \tag{2.2.25}$$

for two-dimensional case, where the solution is given in Appendix A.

Green's Formula on the Boundary

Equation (2.2.23) is valid for any point in the open domain. We need to find the formulation of Green's formula on the boundary [15, p. 48] so as to find the u values at the boundary points. This is done in a simple way. Consider a semi-circle on the boundary of a two-dimensional domain as shown in Figure 2.2.3.

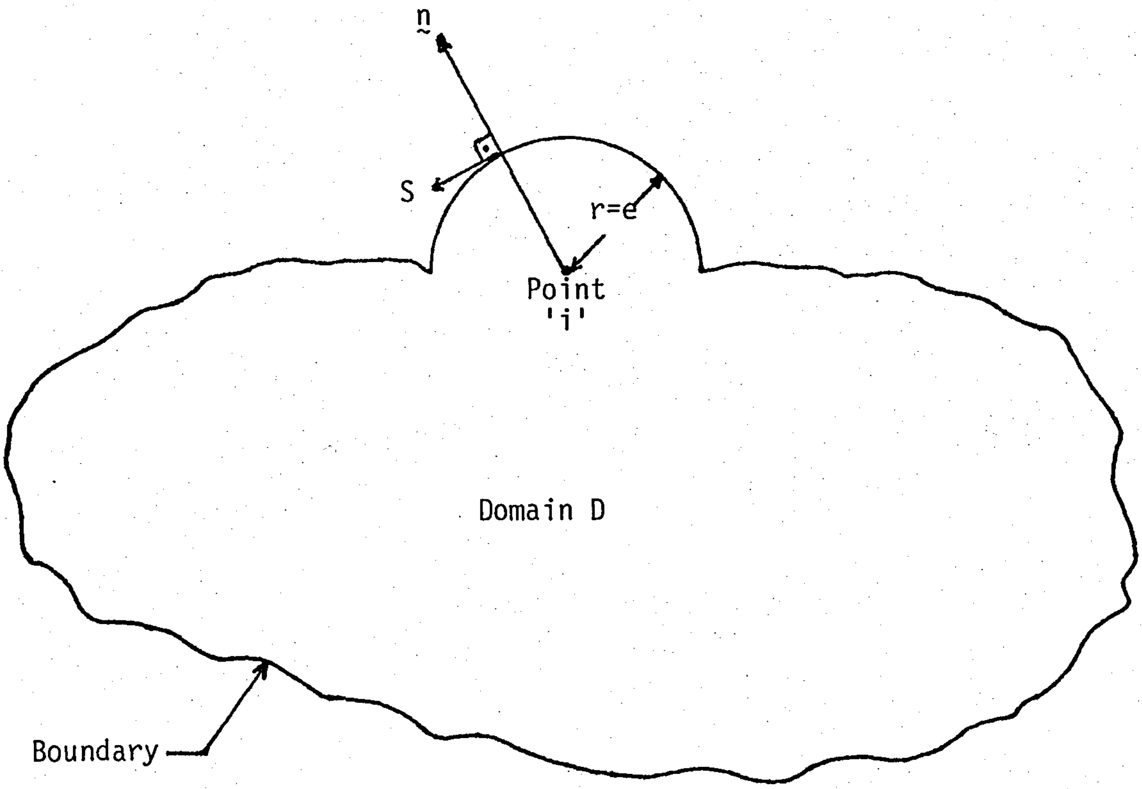


Figure 2.2.3 - Illustration of the point 'i' at the neighbourhood of the boundary.

The point 'i' is located at the center of the semi-circle. As the radius 'e' is reduced to zero, the point becomes a boundary point.

We take the second integral at the right hand side of Eq. (2.2.23). It is evaluated only near the surface S. In order to evaluate the integral at the boundary, consider the surface in two partitions, i.e.,

$$S = S' + S_e \tag{2.2.26a}$$

where S_e is the semi-circle surface and

$$S' = S - S_e \tag{2.2.26b}$$

is the remaining part of the whole surface. Thus,

$$\int_S uq^*dS = \int_{S'} uq^*dS + \int_{S_e} uq^*dS \quad . \tag{2.2.27}$$

Now, taking the limit we get

$$\begin{aligned}\lim_{e\to 0}(\int_{S_e} uq^*dS) &= \lim_{e\to 0}(\int_{S_e} u(-1/2\pi e)dS) \\ &= \lim_{e\to 0}(-(u/2\pi e)(\pi e)) \\ &= -\frac{1}{2}u \quad . \end{aligned} \tag{2.2.28}$$

Note that as e goes to zero, S_e approaches zero in the limit. Therefore,

$$S' \to S \quad ,$$

and

$$\int_{S'} uq^*dS$$

in the domain is equal to

$$\int_S uq^* dS$$

at the boundary. Thus,

$$\int_S uq^* dS$$

in the domain is equal to

$$\int_S uq^* dS - \frac{1}{2} u$$

at the boundary.

For the first integral of the right hand side of Eq. (2.2.23),

$$\int_S u^* q dS \quad ,$$

we perform a similar analysis by splitting S into two parts and writing the above equation as follows.

$$\int_S u^* q dS = \int_{S'} u^* q dS + \int_{S_e} u^* q dS \quad . \tag{2.2.29}$$

Substituting the expression for u* we have

$$\int_{S_e} u^* q dS = \int_{S_e} q \frac{1}{2\pi} \ln(1/e) dS \quad . \tag{2.2.30}$$

Now taking the limit

$$\lim_{e \to 0} \left(\int_{S_e} q \frac{1}{2\pi} \ln(1/e) ds \right) = 0 \tag{2.2.31}$$

and noting that as e goes to zero

$$S' \to S \ ,$$

we have

$$\int_{S'} uq^* dS$$

in the domain equal to

$$\int_{S} uq^* dS$$

at the boundary. Thus,

$$\int_{S} uq^* dS$$

in the domain is equal to

$$\int_{S} uq^* dS$$

at the boundary. As we take the limit as e goes to zero, the u value in Eq. (2.2.23) approaches the u value at the boundary point. So we have

$$u = \int_{S} u^* q dS - \int_{S} uq^* dS + \frac{1}{2} u + \int_{D} pu^* dD \ . \qquad (2.2.32)$$

Rearranging the Eq. (2.2.32) we obtain

$$\frac{1}{2} u = \int_{S} u^* q dS - \int_{S} uq^* dS + \int_{D} pu^* dD \ . \qquad (2.2.33)$$

Equation (2.2.33) is known as the GREEN's BOUNDARY FORMULA or BOUNDARY INTEGRAL EQUATION [15, p.51].

2.3 MATRIX FORMULATION

The boundary element technique can be interpreted in matrix form. Let us consider the boundary integral equation (2.2.33)

$$\frac{1}{2} u = \int_S u^* q dS - \int_S u q^* dS + \int_D p u^* dD \, . \qquad (2.3.1)$$

Let us assume that the body is two-dimensional and its boundary is divided into $\bar{n}$ 'segments' or 'boundary elements', as shown in Figure 2.3.1. The points where the unknown values are considered are called 'nodes'. The elements on which u and q are constant are called 'constant' elements in which the nodes are in the middle of each element (Figure 2.3.1a). The elements on which u and q vary linearly are called 'linear' elements and the nodes are at the intersection of the elements (Figure 2.3.1b).

i. Constant Elements

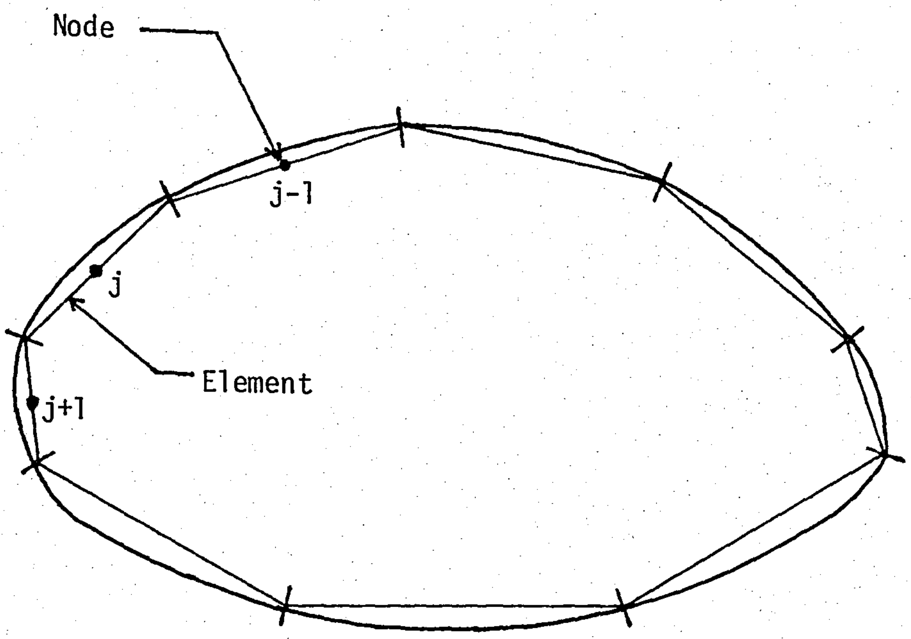
The boundary is divided into $\bar{n}$ elements. The values of u and q are assumed to be constant on each element and equal to the values at the mid-node of the element.

Before the application of any boundary conditions, Eq. (2.3.1) can be discretized as given below.

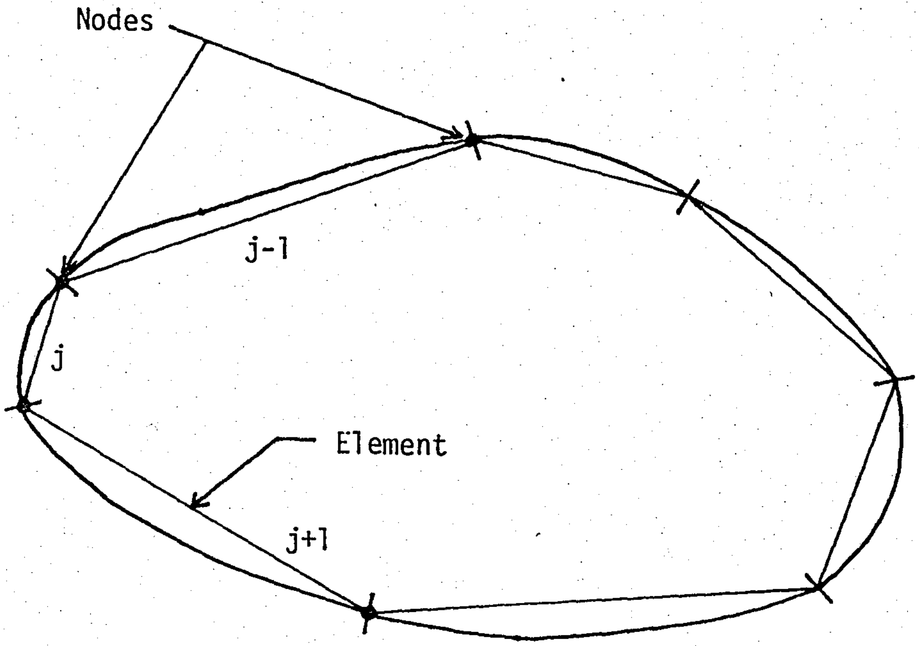
$$-\int_D (pu^*)_i dD + \frac{1}{2} u_i + \sum_{j=1}^{\bar{n}} \left(\int_{S_j} u_j q_i^* dS \right) = \sum_{j=1}^{\bar{n}} \left(\int_{S_j} u_i^* q_j dS \right) . \qquad (2.3.2)$$

It should be noted

$$u^* = u^*(\underset{\sim}{x}, \underset{\sim}{x}_i)$$



(a)



(b)

Figure 2.3.1 - Boundary elements: (a) constant,
(b) linear.

and

$$q^* = q^*(\underset{\sim}{x},\underset{\sim}{x}_i) \quad .$$

Here, $|\underset{\sim}{x}|$ is the distance to the j^{th} element and $|\underset{\sim}{x}_i|$ is the distance to the node 'i' where the distances are from the origin of a prescribed coordinate system.

Equation (2.3.2) applies for a particular node 'i'. The u_j and q_j values can be taken out of the integrals as they are assumed to be constant over each element. This gives

$$-\int_D (pu^*)_i dD + \frac{1}{2} u_i + \sum_{j=1}^{\overline{n}} \left(\int_{S_j} q_i^* dS\right) u_j = \sum_{j=1}^{\overline{n}} \left(\int_{S_j} u_i^* dS\right) q_j \ . \qquad (2.3.3)$$

The integrals $\int q^*dS$ relate the 'i' node with the boundary element 'j' over which the integral is carried out. We shall call these integrals $\hat{H}_{ij}$. Also, we shall denote the integrals $\int u^*dS$ on the right hand side of Eq. (2.3.3) as G_{ij}. Hence, we can write Eq. (2.3.3) as follows.

$$-\int_D (pu^*)_i dD + \frac{1}{2} u_i + \sum_{j=1}^{\overline{n}} \hat{H}_{ij} u_j = \sum_{j=1}^{\overline{n}} G_{ij} q_j \ . \qquad (2.3.4)$$

We can rearrange the Eq. (2.3.4). Let us now define

$$H_{ij} = \begin{cases} \hat{H}_{ij} & \text{when } i \neq j \\ \hat{H}_{ij} + \frac{1}{2} & \text{when } i = j \end{cases} \ . \qquad (2.3.5)$$

Equation (2.3.4) can now be written as follows.

$$B_i + \sum_{j=1}^{\overline{n}} H_{ij} u_j = \sum_{j=1}^{\overline{n}} G_{ij} q_j \qquad (2.3.6)$$

where,

$$B_i = -\int_D (pu^*)_i dD.$$

The whole set of equations can also be expressed in matrix form as given below.

$$\underset{\bar{n}x1}{\{B\}} + \underset{\bar{n}x\bar{n}}{[H]} \underset{\bar{n}x1}{\{u\}} = \underset{\bar{n}x\bar{n}}{[G]} \underset{\bar{n}x1}{\{q\}} \quad . \tag{2.3.7}$$

It should be noted that there are $\bar{n}$ unknowns in Eq. (2.3.7).

Evaluation of the Integrals

The integrals $\hat{H}_{ij}$ and G_{ij} can be calculated using the simple Gauss quadrature rule [16, p.420] for all points, except the one corresponding to the node under consideration. Let us choose the new coordinate system (Figure 2.3.2) as follows.

$$ds = |r_1| d\xi \tag{2.3.8}$$

where

$$|r_1| = |r_2| \; . \tag{2.3.9}$$

We then have

$$G_{ij} = \int_{-1}^{+1} u_i^* |r_1| d\xi \; , \tag{2.3.10}$$

$$\hat{H}_{ij} = \int_{-1}^{+1} q_i^* |r_1| d\xi \quad . \tag{2.3.11}$$

For the element 'j', we can take

$$\frac{\ell^j}{2} = |r_1| \quad . \tag{2.3.12}$$

Then the Eqs.(2.3.10) and (2.3.11) respectively reduces to

$$G_{ij} = \int_{-1}^{+1} u_i^*(\ell^j/2)d\xi \quad , \tag{2.3.13}$$

$$\hat{H}_{ij} = \int_{-1}^{+1} q_i^*(\ell^j/2)d\xi \quad . \tag{2.3.14}$$

By using the simple Gauss quadrature rule, we can write

$$G_{ij} = \sum_{t=1}^{tm} Z_t . u^*(\underset{\sim}{x}_i,\underset{\sim}{x}_t) . \frac{\ell^j}{2} \quad , \tag{2.3.15}$$

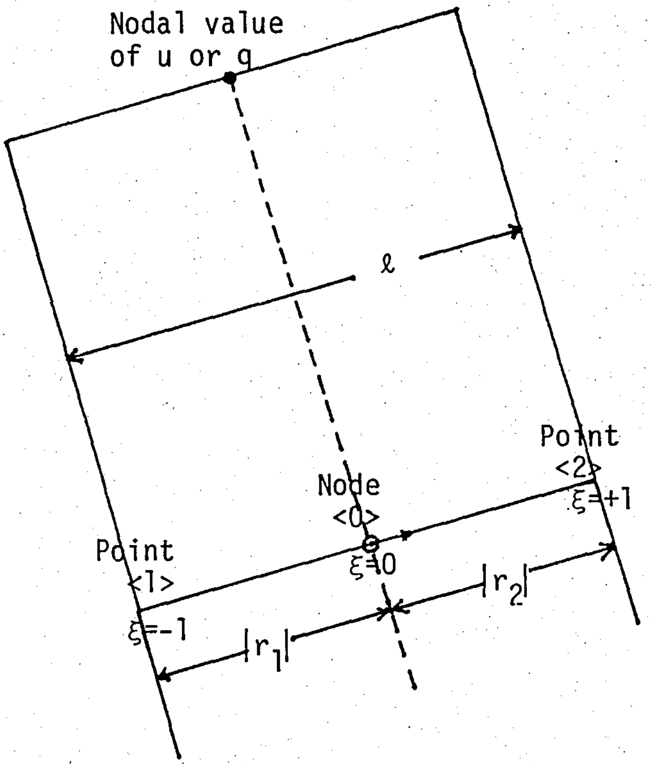


Figure 2.3.2 - Constant element coordinates.

$$\hat{H}_{ij} = \sum_{t=1}^{tm} Z_t \cdot q^*(\underset{\sim}{x}_i, \underset{\sim}{x}_t) \cdot \frac{\ell^j}{2} \quad . \tag{2.3.16}$$

Here, Z_t are the Gauss weighting coefficients. $|\underset{\sim}{x}_i|$ is the distance to the node 'i' and $|\underset{\sim}{x}_t|$ is the distance to the integration point 't' where the distances are from the origin of a prescribed coordinate system. tm is the total number of integration points on each 'j' element. It should be noted that

$$q^* = \frac{\partial u^*}{\partial n} \tag{2.3.17}$$

$$= \frac{\partial u^*}{\partial r} \cos(\underset{\sim}{n}, \underset{\sim}{r}) \tag{2.3.18}$$

where,

$$\cos(\underset{\sim}{n}, \underset{\sim}{r}) = \frac{d}{r} \tag{2.3.19}$$

as shown in Figure 2.3.3. Thus,

$$q^* = - \frac{d}{2\pi r^2} \quad . \tag{2.3.20}$$

For the particular case of constant elements, however, the $\hat{H}_{ii}$ and G_{ii} integrals can be easily computed analytically. The $\hat{H}_{ii}$ term, for instance, is identically zero for fundamental solutions with no S dependence, i.e.,

$$\begin{aligned} \hat{H}_{ii} &= \int_{S_i} q_i^* dS \\ &= \int_{S_i} \left(\frac{\partial u_i^*}{\partial r} \cdot \frac{\partial r}{\partial n}\right) dS \\ &= 0 \quad . \end{aligned} \tag{2.3.21}$$

This is due to the fact that $\underset{\sim}{n}$ and $\underset{\sim}{r}$ are orthogonal over the element.

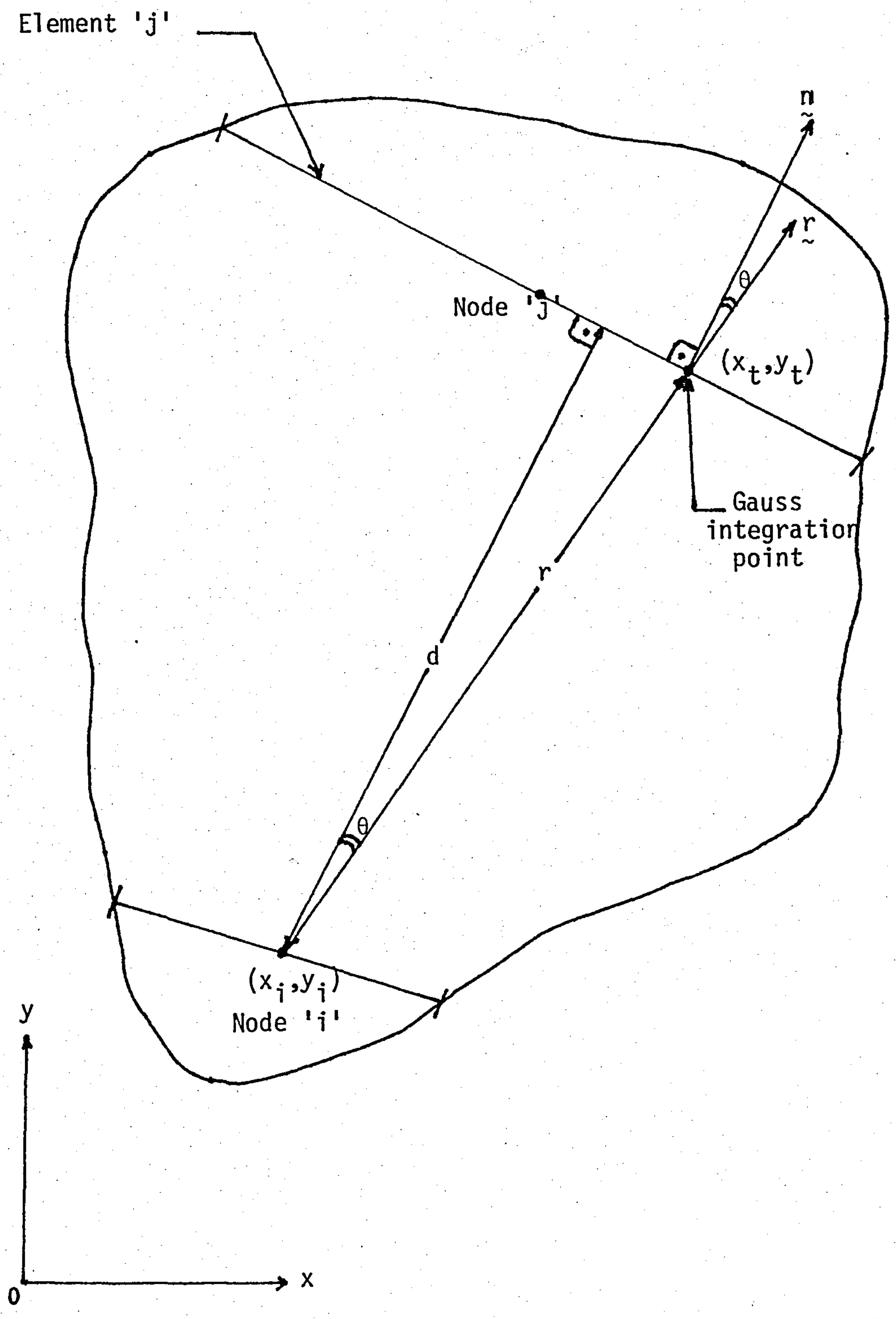


Figure 2.3.3 - The angle between the vectors $\underset{\sim}{n}$ and $\underset{\sim}{r}$.

The G_{ii} integral can be calculated analytically as follows.

$$G_{ii} = \int_{S_i} u_i^* dS$$

$$= \frac{1}{2\pi} \int_{S_i} \ln(1/r) dS \quad . \qquad (2.3.22)$$

With the use of the homogeneous coordinate ξ over an element (Figure 2.3.2) we get

$$G_{ii} = \frac{1}{2\pi} \int_{<1>}^{<2>} \ln(1/r) dS$$

$$= \frac{1}{\pi} \int_{<0>}^{<2>} \ln(1/r) dS \quad . \qquad (2.3.23)$$

On transforming the coordinate system as given below

$$dS = |r_1| d\xi$$

we get

$$G_{ii} = \frac{|r_1|}{\pi}[\ln(1/|r_1|) + \int_0^1 \ln(1/\xi) d\xi] \; . \qquad (2.3.24)$$

Noting that the last integral is equal to 1 we have

$$G_{ii} = \frac{1}{\pi} |r_1| [\ln(1/|r_1|) + 1] \qquad (2.3.25)$$

where,

$$|r_1| = \frac{\ell^i}{2} \quad .$$

ii. Linear Elements

Let us consider a linear variation for u and q (Figure 2.3.4). The nodes are now considered to be at the intersection between two straight elements such as those shown in Figure 2.3.1b .

Consider the Eq. (2.3.2)

$$B_i + \frac{1}{2} u_i + \sum_{j=1}^{\bar{n}} \left(\int_{S_j} u_j q_i^* dS \right) = \sum_{j=1}^{\bar{n}} \left(\int_{S_j} q_j u_i^* dS \right) . \qquad (2.3.26)$$

The integrals in the above equation are now more difficult to evaluate than in the constant element case because u and q vary linearly over the element.

The values of u and q at any point on the element can be defined in terms of their nodal values and two linear interpolation functions denoted as ϕ_1 and ϕ_2. Here, both ϕ_1 and ϕ_2 are functions of the coordinate ξ so that,

$$u(\xi) = \phi_1 u_1 + \phi_2 u_2 \qquad (2.3.27a)$$

$$= [\phi_1 \phi_2] \begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} \qquad (2.3.27b)$$

and

$$q(\xi) = \phi_1 q_1 + \phi_2 q_2 \qquad (2.3.28a)$$

$$= [\phi_1 \phi_2] \begin{Bmatrix} q_1 \\ q_2 \end{Bmatrix} . \qquad (2.3.28b)$$

The dimensionless coordinate ξ is given as follows.

$$\xi = x/((1/2)\ell) .$$

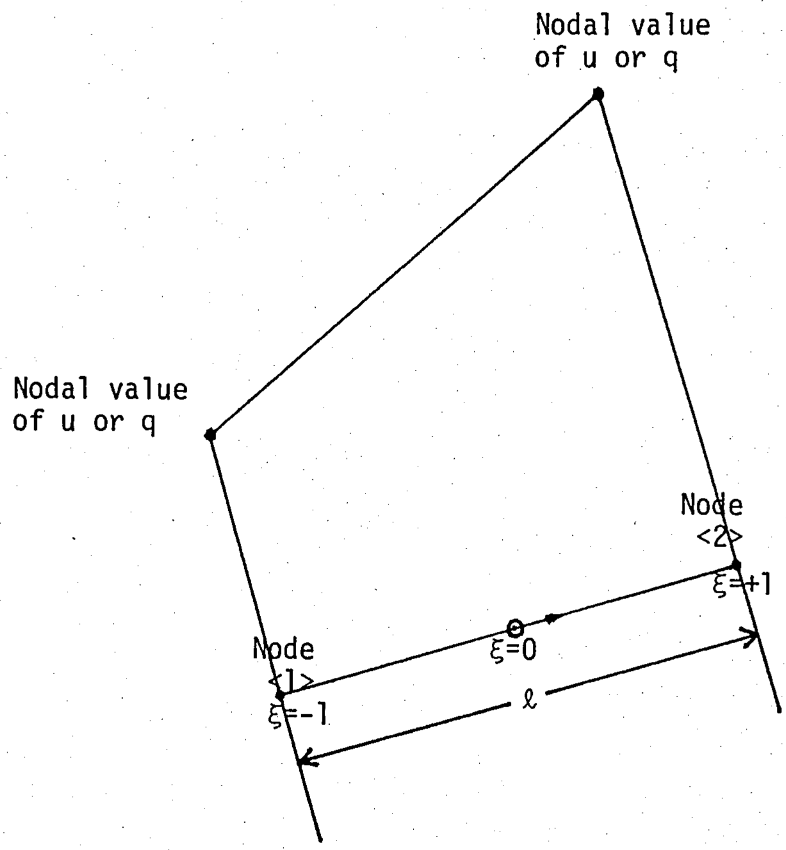


Figure 2.3.4 - Linear element coordinates.

The functions ϕ_1 and ϕ_2 are given as stated below.

$$\phi_1 = \frac{1}{2}(1 - \xi) \tag{2.3.29a}$$

and

$$\phi_2 = \frac{1}{2}(1 + \xi) \quad . \tag{2.3.29b}$$

The integrals along an element 'j', that appears on the left hand side of the Eq. (2.3.26) can be written as follows.

$$\int_{S_j} u(\xi) q_i^* dS = \int_{S_j} [\phi_1 \phi_2] q_i^* dS \begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} \tag{2.3.30a}$$

$$= [h_{ij}^1 \ h_{ij}^2] \begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} \tag{2.3.30b}$$

Here,

$$h^1_{ij} = \int_{S_j} \phi_1 q^*_i dS \ , \tag{2.3.31a}$$

$$h^2_{ij} = \int_{S_j} \phi_2 q^*_i dS \ . \tag{2.3.31b}$$

The h^m_{ij} are influence coefficients defining the interaction between the point 'i' under consideration and a particular node 'm' on an element 'j'.

Similarly, for the integrals on the right hand side of the Eq. (2.3.26), we can write

$$\int_{S_j} q(\xi) u^*_i dS = \int_{S_j} [\phi_1 \phi_2] u^*_i dS \begin{Bmatrix} q_1 \\ q_2 \end{Bmatrix} \tag{2.3.32a}$$

$$= [g^1_{ij} \ g^2_{ij}] \begin{Bmatrix} q_1 \\ q_2 \end{Bmatrix} \tag{2.3.32b}$$

where,

$$g^1_{ij} = \int_{S_j} \phi_1 u^*_i dS \ , \tag{2.3.33a}$$

$$g^2_{ij} = \int_{S_j} \phi_2 u^*_i dS \ . \tag{2.3.33b}$$

The g^m_{ij} are influence coefficients defining the interaction between the point 'i' under consideration and a particular node 'm' on an element 'j'.

To write the equation corresponding to node 'i' in discrete form we need to sum up the contributions from two adjoining elements, 'j-1' and 'j', into one term defining the nodal coefficient. This will give the following equation.

$$B_i + \frac{1}{2} u_i + [\hat{H}_{i1}\hat{H}_{i2}\cdots\hat{H}_{i\bar{n}}] \begin{Bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{\bar{n}} \end{Bmatrix} = [G_{i1}G_{i2}\cdots G_{i\bar{n}}] \begin{Bmatrix} q_1 \\ q_2 \\ \vdots \\ q_{\bar{n}} \end{Bmatrix} \qquad (2.3.$$

Here,

$$\hat{H}_{ij} = h^1_{ij} + h^2_{i(j-1)} \qquad (2.3.35)$$

The same applies for G_{ij}, i.e.,

$$G_{ij} = g^1_{ij} + g^2_{i(j-1)} \quad . \qquad (2.3.36)$$

Hence, Eq. (2.3.34) represents the assembled equation for node 'i' and it can be written as follows.

$$B_i + \frac{1}{2} u_i + \sum_{j=1}^{\bar{n}} \hat{H}_{ij} u_j = \sum_{j=1}^{\bar{n}} G_{ij} q_j \qquad (2.3.37)$$

or, more simply,

$$B_i + \sum_{j=1}^{\bar{n}} H_{ij} u_j = \sum_{j=1}^{\bar{n}} G_{ij} q_j \qquad (2.3.38)$$

where,

$$H_{ij} = \begin{cases} \hat{H}_{ij} & \text{when } i \neq j \\ \hat{H}_{ij} + \frac{1}{2} & \text{when } i = j \end{cases} \qquad (2.3.39)$$

When all the nodes are taken into consideration, Eq. (2.3.38) produces a $\bar{n}x\bar{n}$ system of equations which can be represented in matrix form as follows.

$$\underset{\bar{n}x1}{\{B\}} + \underset{\bar{n}x\bar{n}}{[H]} \underset{\bar{n}x1}{\{u\}} = \underset{\bar{n}x\bar{n}}{[G]} \underset{\bar{n}x1}{\{q\}} \quad . \qquad (2.3.40)$$

We can calculate the diagonal terms of [H] by using the fact that when a uniform potential is applied on the whole boundary the values of q must be zero. Let us also assume that there is no heat generation, i.e.,

$$\{B\} = \{0\} \quad . \tag{2.3.41}$$

Under these conditions Eq. (2.3.40) produces

$$[H]\{u\} = \{0\} \quad . \tag{2.3.42}$$

Equation (2.3.42) indicates that the sum of all the elements of [H] in a row ought to be zero, hence, the values of the coefficients in the diagonal can be easily calculated once the off-diagonal coefficients are all known, i.e.,

$$H_{ii} = - \sum_{\substack{j=1 \\ j\neq 1}}^{\bar{n}} H_{ij} \quad . \tag{2.3.43}$$

The result derived above is applicable for the general case, because [G] and [H] do not depend on the boundary conditions or heat generation.

In order to integrate the B_i integrals we need to discretize the domain D into a series of 'cells' or 'interior elements' as shown in Figure 2.3.5. The procedure is similar to that of the finite element method, but conceptually it is different because we do not deal with the u and q values at the interior points.

Let's consider $\bar{m}$ interior elements. We can then write

$$B_i = - \int_D (pu^*)_i dD \tag{2.3.44}$$

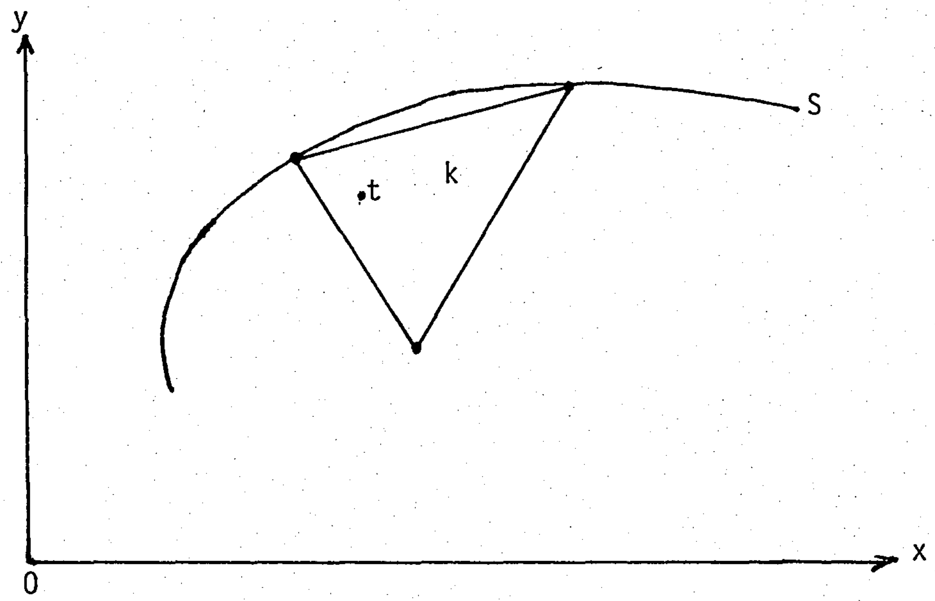


Figure 2.3.5 - Interior cell k and integration point t.

$$= \sum_{k=1}^{\bar{m}} \left(\int_{D_k} (pu^*)_i dD \right) \quad . \qquad (2.3.45)$$

Over each element a numerical integration formula can then be applied as follows.

$$B_i = - \sum_{k=1}^{\bar{m}} \left(\sum_{t=1}^{tn} Z_t . p_i . u^*(\underset{\sim}{x}_i , \underset{\sim}{x}_t) . \det[J] \right)_k \qquad (2.3.46)$$

Here, t is the integration point, Z_t is the weighting function, tn is the total number of integration points on each cell k, $|\underset{\sim}{x}_i|$ is the distance to the node 'i', and $|\underset{\sim}{x}_t|$ is the distance to the integration point 't' where the distances are from the origin of a prescribed system.

It should be noted that

$$\det[J] = 2(\text{Area})_{\text{triangular cell}} \quad .$$

With the use of Eq. (2.2.23) we can calculate the u value at any interior point as follows.

$$u = \int_S u^* q \, dS - \int_S u q^* dS + \int_D p u^* dD \quad . \tag{2.3.48}$$

The u values can be obtained directly from Eq. (2.3.48) by discretizing as follows.

$$u_i = \sum_{j=1}^{\bar{n}} q_j G_{ij} - \sum_{j=1}^{\bar{n}} u_j \hat{H}_{ij} - B_i \quad . \tag{2.3.49}$$

Here,

$$B_i = -\int_D (pu^*)_i dD$$

and definitions for G_{ij} and $\hat{H}_{ij}$ are given in Eqs. (2.3.10) and (2.3.11) respectively.

2.4 BOUNDARY CONDITIONS

The general matrix equation was found as given below.

$$\underset{\bar{n}x\bar{n}}{[G]} \; \underset{\bar{n}x1}{\{q\}} = \underset{\bar{n}x\bar{n}}{[H]} \; \underset{\bar{n}x1}{\{u\}} + \underset{\bar{n}x1}{\{B\}} \quad . \tag{2.4.1}$$

We can rearrange the equation (2.4.1) as follows.

$$\underset{\bar{n}x\bar{n}}{[A]} \; \underset{\bar{n}x1}{\{X\}} = \underset{\bar{n}x1}{\{F\}} \quad . \tag{2.4.2}$$

Now, let us analyse the boundary conditions involved and the forms which the matrix [A] and vectors {X} and {F} take, respectively.

1. The boundary conditions are all of the first kind, i.e.,

$$\{u\} = \{\bar{u}\} \qquad \text{on} \quad S \quad . \tag{2.4.3}$$

Inserting the boundary condition into Eq. (2.4.1) we obtain

$$[G]\{q\} = [H]\{\bar{u}\} + \{B\} \quad . \tag{2.4.4}$$

Here, it is easily seen that we can rearrange Eq. (2.4.4) by letting

$$[G] = [A] \quad , \tag{2.4.5}$$

$$\{q\} = \{X\} \tag{2.4.6}$$

and

$$[H]\{\bar{u}\} + \{B\} = \{F\} \quad . \tag{2.4.7}$$

Thus, the Eq. (2.4.3) becomes

$$[A]\{X\} = \{F\} \quad . \tag{2.4.8}$$

Formally, the solution is

$$\{X\} = [A]^{-1}\{F\} \quad . \tag{2.4.9}$$

2. The boundary conditions are all of the second kind, i.e., the flux is known on the boundary S. We can show this as follows.

$$\{q\} = \{\bar{q}\} \qquad \text{on} \quad S \quad . \tag{2.4.10}$$

On applying the boundarycondition to Eq. (2.4.1) we obtain

$$[G]\{\bar{q}\} = [H]\{u\} + \{B\} \quad . \tag{2.4.11}$$

Rearranging the above equation as

$$-[H]\{u\} = -[G]\{\overline{q}\} + \{B\} \tag{2.4.12}$$

and letting

$$-[H] = [A] \quad , \tag{2.4.13}$$

$$\{u\} = \{X\} \tag{2.4.14}$$

and

$$-[G]\{\overline{q}\} + \{B\} = \{F\} \tag{2.4.15}$$

we again have a simple matrix formulation of the problem as

$$[A]\{X\} = \{F\} \quad .$$

3. The boundary condition is of the third kind, i.e., there is convective heat transfer on the boundary which can be formulated as follows.

$$\{q\} = [\alpha](\{\beta\} - \{u\}) \quad \text{on} \quad S \tag{2.4.16}$$

Here $[\alpha]$ is a diagonal matrix and $\{\beta\}$ is a vector and both of them are known quantities. Applying this boundary condition to Eq. (2.4.1), we get

$$[G][\alpha](\{\beta\} - \{u\}) = [H]\{u\} + \{B\} \tag{2.4.17a}$$

or

$$-([H] + [G][\alpha])\{u\} = -([G][\alpha])\{\beta\} + \{B\} \tag{2.4.17b}$$

Further, if we let

$$-([H] + [G][\alpha]) = [A] \quad , \tag{2.4.18}$$

$$\{u\} = \{X\} \tag{2.4.19}$$

and

$$-([G][\alpha])\{\beta\} + \{B\} = \{F\} \tag{2.4.20}$$

we obtain, from Eq. (2.4.17b),

$$[A]\{X\} = \{F\} \quad . \tag{2.4.21}$$

We can solve the Eq. (2.4.21) for $\{X\}$, i.e.,

$$\begin{aligned}\{u\} &= \{X\} \\ &= [A]^{-1}\{F\} \quad . \end{aligned} \tag{2.4.22}$$

q can then be obtained from Eq. (2.4.16).

4. Radiation boundary conditions can be imposed by requiring that

$$q = \frac{\sigma}{K}(u_\infty^4 - u^4) \tag{2.4.23}$$

on the boundary S, where σ is the Stefan-Boltzmann constant and u_∞ is the ambient temperature. As it can be seen from Eq. (2.4.23), this kind of boundary condition is nonlinear. For simplicity, we shall try to linearize it.
Let us consider the nonlinear function

$$f(u) = u^4 \quad . \tag{2.4.24}$$

We can approximate this function with a linear expression (Figure 2.4.1),

$$P_1(u) = f(u_1) + (u - u_1)g(u_2,u_1) \tag{2.4.25}$$

with,

$$g(u_2,u_1) = \frac{f(u_2) - f(u_1)}{u_2 - u_1} \tag{2.4.26}$$

which is the slope of the line $P_1(u)$. Hence,

$$f(u) = P_1(u) + R_1 \tag{2.4.27}$$

where R_1 is the remainder arising from the above stated approximation.

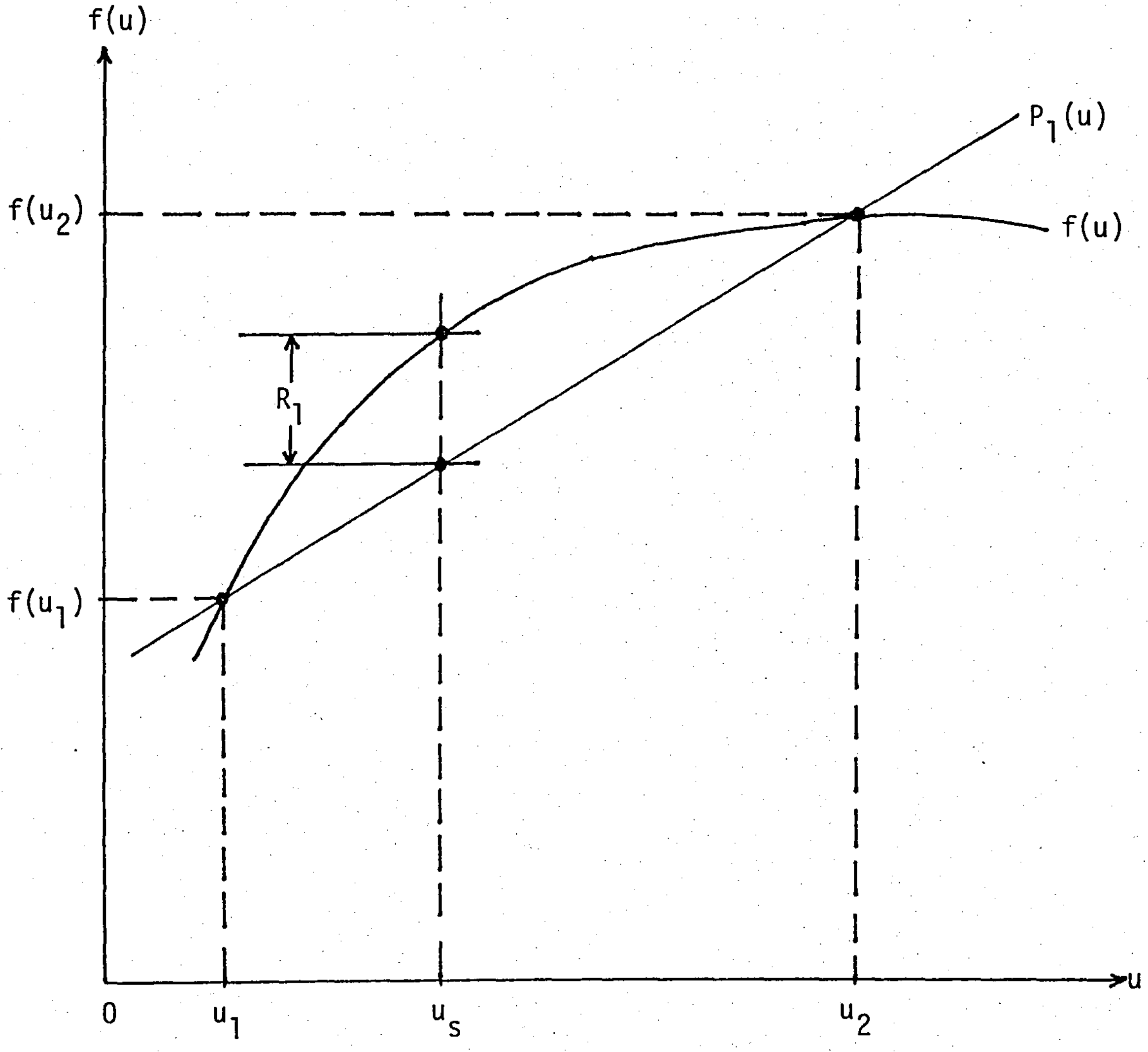


Figure 2.4.1 - Linear approximation to the nonlinear function f(u)

Meanwhile, we can make use of the mean-value theorem [17,p.13] by choosing a point u_s between u_1 and u_2 so that

$$f'(u_s) = \frac{f(u_2) - f(u_1)}{u_2 - u_1} \tag{2.4.28}$$

By evaluating the derivative of f(u) at the point u_s, we get

$$f'(u_s) = 4u_s^3 \tag{2.4.29}$$

$$= g(u_1,u_2) \quad . \tag{2.4.30}$$

Hence, the linear function $P_1(u)$ becomes

$$P_1(u) = f(u_1) + (u - u_1)4u_s^3 \tag{2.4.31}$$

By substituting it into the Eq. (2.4.27) we obtain

$$f(u) = f(u_1) + (u - u_1)4u_s^3 + R_1 \quad . \tag{2.4.32}$$

If we evaluate the function at the point u_s, we get

$$f(u_s) = f(u_1) + (u_s - u_1)4u_s^3 + R_1 \tag{2.4.33}$$

or,

$$u_s^4 = u_1^4 + (u_s - u_1)4u_s^3 + R_1 \tag{2.4.34}$$

Thus,

$$R_1 = u_s^4 - u_1^4 + (u_s - u_1)4u_s^3 \quad . \tag{2.4.35}$$

Substituting the above expression for R_1 into the Eq. (2.4.32), we get

$$f(u) = 4u_s^3.u - 3u_s^4 \tag{2.4.36}$$

which is the linear approximate expression for f(u). Applying the above approximation to Eq. (2.4.23), we obtain

$$q = \frac{\sigma}{K}\left(u_\infty^4 + 3u_s^4 - (4u_s^3)u\right) \quad . \tag{2.4.37}$$

Thus, the linear form of the boundary condition can be written as follows.

$$q = \alpha(\beta - u) \tag{2.4.38}$$

Here,

$$\beta = \frac{3u_s^4 + u_\infty^4}{4u_s^3} \tag{2.4.39a}$$

and

$$\alpha = \frac{\sigma}{K}(4u_s^3) \quad . \tag{2.4.39b}$$

The Eq. (2.4.38) is of the same form as the third kind of boundary condition except u_s which is not known. In finding the values of u, we will use an iterative procedure [18], which starts by taking u_s equal to u_∞. After selecting the u_s value we calculate the values of α and β from the Eqs. (2.4.39). Then we find the u value using the procedure for the third kind of boundary condition. We denote this value of u as $u^{(1)}$ and assign it as the new value of u_s. The same procedure is repeate and a new value of u is found and denoted as $u^{(2)}$. The new valu of u_s can be chosen as $u^{(2)}$, and the procedure is repeated as before until a convergence criterion in the form

$$|u^{(n+1)} - u^{(n)}|$$

reaches a certain value.

The problems which have various types of nonlinear boundary conditions can be solved in the same way. An example is the case in which the convective heat transfer coefficient h_o is the function of the temperature u, i.e.,

$$q = \frac{h_o(u)}{K}(u_\infty - u) \quad . \tag{2.4.40}$$

5. The boundary conditions are 'mixed'. In this case, more than one kind of boundary conditions prevail on the boundary. One can meet various versions of this kind. An example will be given to show how they are treated.

<u>Example</u>

Consider a triangular plate as shown in Figure 2.4.2 on which the boundary conditions are imposed as follows.

$$u = \overline{u} \qquad \text{on} \quad S_1 ,$$

$$q = \overline{q} \qquad \text{on} \quad S_2 ,$$

and

$$q = \alpha(\beta - u) \qquad \text{on} \quad S_3$$

where the boundary S is the sum of S_1, S_2 and S_3. The problem is to find the temperature u on S_2 and S_3 and the flux on S_1 and S_3.

We consider S_1, S_2 and S_3 as constant elements with nodes 1, 2 and 3 in the middle of each of them respectively. Considering the boundary integral equation

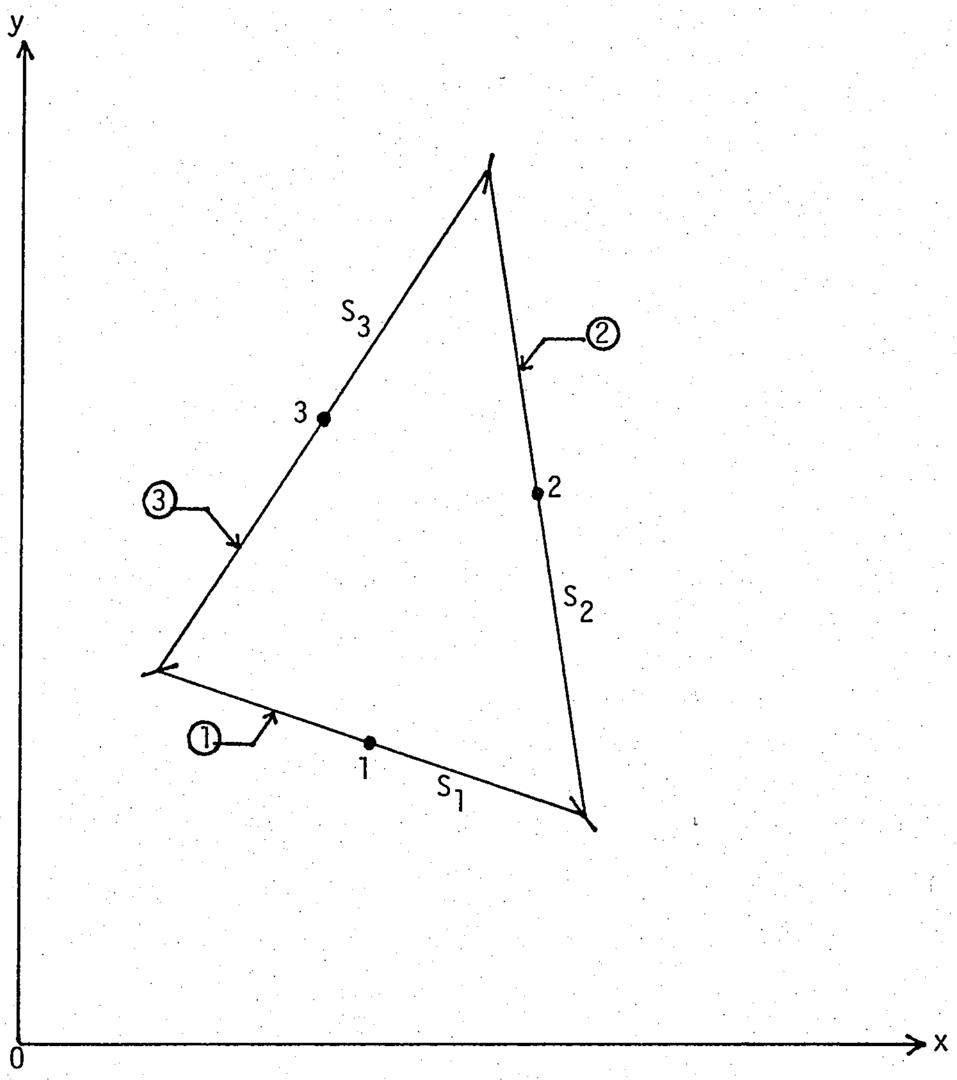


Figure 2.4.2 - An illustration for the boundary conditions of the 'mixed' kind.

$$\sum_{j=1}^{\overline{n}} G_{ij}q_j = \sum_{j=1}^{\overline{n}} H_{ij}u_j + B_i$$

where 'i' and 'j' denote nodes and elements, respectively. We insert the boundary conditions for each node. Thus, for i = 1

$$G_{11}q_1 + G_{12}\overline{q}_2 + G_{13}\alpha_3(\beta_3 - u_3) = H_{11}\overline{u}_1 + H_{12}u_2 + H_{13}u_3 + B_1. \tag{2.4.41}$$

Rearranging the Eq. (2.4.41), we get

$$G_{11}q_1 - H_{12}u_2 - (H_{13} + G_{13}\alpha_3)u_3 = H_{11}\bar{u}_1 - G_{12}\bar{q}_2 + G_{13}\alpha_3\beta_3 + B_1 \ . \qquad (2.4.42a)$$

Similarly, for i = 2

$$G_{21}q_1 - H_{22}u_2 - (H_{23} + G_{23}\alpha_3)u_3 = H_{21}\bar{u}_1 - G_{22}\bar{q}_2 - G_{23}\alpha_3\beta_3 + B_2 \qquad (2.4.42b)$$

and for i = 3

$$G_{31}q_1 - H_{32}u_2 - (H_{33} + G_{33}\alpha_3)u_3 = H_{31}\bar{u}_1 - G_{32}\bar{q}_2 - G_{33}\alpha_3\beta_3 + B_3 \ . \qquad (2.4.42c)$$

If we write in matrix form, we have

$$\begin{bmatrix} G_{11} & -H_{12} & -(H_{13}+G_{13}\alpha_3) \\ G_{21} & -H_{22} & -(H_{23}+G_{23}\alpha_3) \\ G_{31} & -H_{32} & -(H_{33}+G_{33}\alpha_3) \end{bmatrix} \begin{bmatrix} q_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} H_{11}\bar{u}_1-G_{12}\bar{q}_2-G_{13}\alpha_3\beta_3+B_1 \\ H_{21}\bar{u}_1-G_{22}\bar{q}_2-G_{23}\alpha_3\beta_3+B_2 \\ H_{31}\bar{u}_1-G_{32}\bar{q}_2-G_{33}\alpha_3\beta_3+B_3 \end{bmatrix} \qquad (2.4.4$$

Hence, we can solve for the unknown quantities q_1, u_2 and u_3.

2.5 NUMERICAL PROCEDURE

Numerical procedure for the boundary integral element method may be outlined as follows.

STEP(1): Discretization of the boundary.

STEP(2): Division of the domain into interior cells. (if $p \neq 0$)

STEP(3): Interpolation of the temperature u and the flux q over the boundary elements.

STEP(4): Evaluation of the influence matrices [G] and [H].

STEP(5): Introduction of the boundary conditions.

STEP(6): Decomposition of the modified influence matrix [A] into the triangular form by Gaussian Forward Elimination.

STEP(7): Evaluation of the source vector {B} and the right hand side vector in the system of algebraic equations

$$[A]\{X\} = \{F\} \quad .$$

STEP(8): Solution of the algebraic equations for the unknown values of u and q on the boundary.

STEP(9): Evaluation of the internal temperature values.

III. SAMPLE PROBLEMS

In this section, 5 sample problems are solved and their results are given on tables. The second and third problems involve internal heat generation whereas the others don't have internal heat generation. The cross-section of an industrial furnace is considered in the problem 3. The fourth problem involves radiation boundary condition. Finally, the case of hollow cylinder is solved in problem 5.

Sample Problem 1

Consider the steady-state heat conduction in a square region, without heat generation. The mathematical formulation of the heat conduction problem is

$$\nabla^2 u = 0 \qquad \text{in} \qquad 0 \le x \le 1 \ , \quad 0 \le y \le 1 \quad .$$

Find the temperature at the internal and boundary points, and the flux at boundary points for each of the cases shown below.

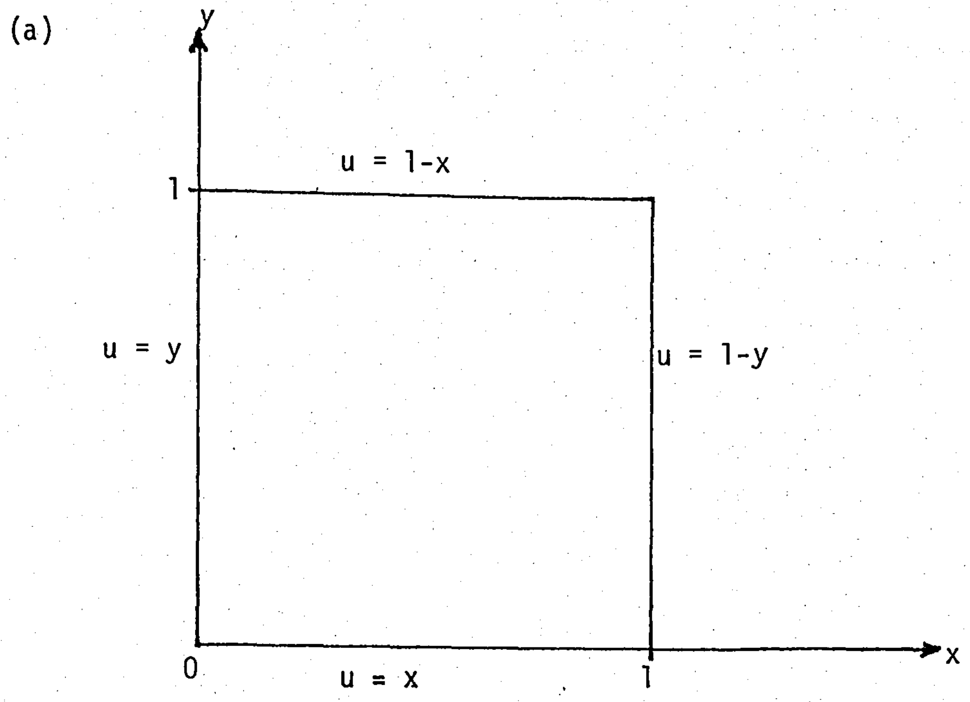


Figure 3.1 - Boundary conditions for the problem 1a.

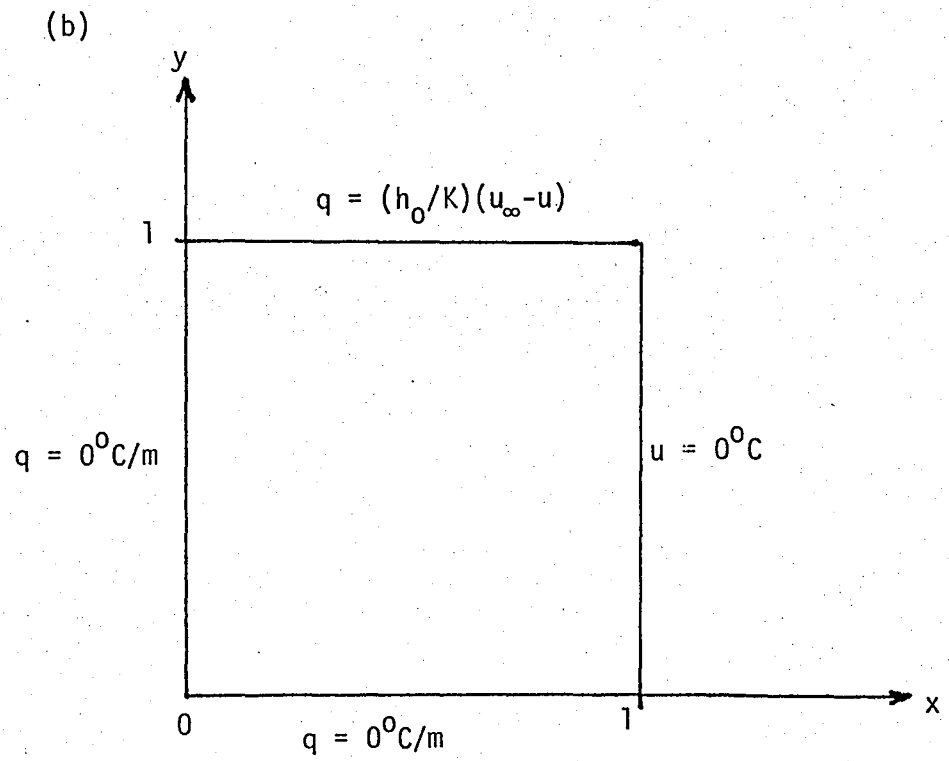


Figure 3.2 - Boundary conditions for the problem 1b.

TABLE 3.1 - Results for the Problem 1a.

BIEM: 20 Linear Boundary Elements (CPU[†] = 3.902 sec.)						
BOUNDARY NODES	COORDINATES (m)		TEMPERATURE u (^oC)		FLUX[††] $q = \partial u/\partial n$ ($^oC/m$)	
	X	Y	EXACT (App.B)	BIEM	EXACT	BIEM
1	0.001	0.000	0.001	0.001	-1.000	-1.018
3	0.500	0.000	0.500	0.500	0.000	0.000
5	0.999	0.000	0.999	0.999	1.000	1.018
7	1.000	0.250	0.750	0.750	0.500	0.492
9	1.000	0.750	0.250	0.250	-0.500	-0.492
11	0.999	1.000	0.001	0.001	-1.000	-1.018
13	0.500	1.000	0.500	0.500	0.000	0.000
15	0.001	1.000	0.999	0.999	1.000	1.018
17	0.000	0.750	0.750	0.750	0.500	0.492
19	0.000	0.250	0.250	0.250	-0.500	-0.492

INTERNAL POINTS	COORDINATES (m)		TEMPERATURE u (^oC)	
	X	Y	EXACT	BIEM
1	0.250	0.250	0.375	0.375
3	0.750	0.750	0.375	0.375
5	0.500	0.500	0.500	0.500

For 1b, the convective heat transfer coefficient h_o, thermal conductivity K and ambient temperature u_∞ are given as follows.

[†] *Central Processor unit time for UNIVAC 1106.*

[††] *Flux is '+' when there is heat input to the region.*

$h_o = 20 \ \ W/m^2.^oC$,

$K = 1 \ \ W/m.^oC$,

and $u_\infty = 1 \ ^oC$.

TABLE 3.2 - Results for the Problem 1b.

BIEM: 20 Linear Boundary Elements (CPU = 5.437 sec.)
FDM : 81 Grid Points (CPU = 1.459 sec.)

BOUNDARY NODES	COORDINATES (m)		TEMPERATURE u (^oC)			FLUX $q = \partial u/\partial n$ ($^oC/m$)	
	X	Y	EXACT (APP.B)	FDM (APP.D)	BIEM	EXACT	BIEM
1	0.001	0.000	0.467	0.466	0.464	0.000	0.000
3	0.500	0.000	0.339	0.339	0.338	0.000	0.000
5	0.999	0.000	0.001	0.000	0.004	0.000	0.000
7	1.000	0.250	0.000	0.000	0.000	-0.858	-0.771
9	1.000	0.750	0.000	0.000	0.000	-2.164	0.399
11	0.999	1.000	0.018	0.000	0.216	19.640	15.691
13	0.500	1.000	0.930	0.936	0.938	1.400	1.245
15	0.001	1.000	0.956	0.957	0.959	0.880	0.821
17	0.000	0.750	0.756	0.754	0.756	0.000	0.000
19	0.000	0.250	0.500	0.499	0.496	0.000	0.000

INTERNAL POINTS	COORDINATES (m)		TEMPERATURE u (^oC)	
	X	Y	EXACT	BIEM
1	0.250	0.250	0.467	0.465
3	0.750	0.750	0.444	0.437
5	0.500	0.500	0.464	0.463
7	0.781	0.969	0.782	0.792
9	0.844	0.969	0.715	0.783
11	0.781	0.781	0.439	0.427

(c)

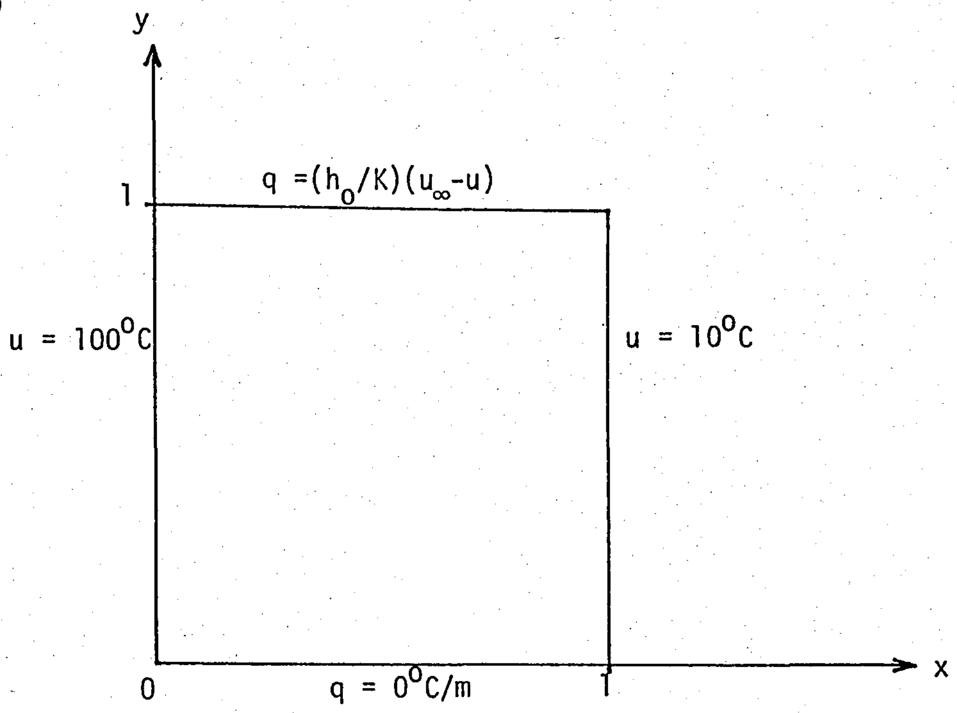


Figure 3.3 - Boundary conditions for the problem 1c.

In this case, the convective heat transfer coefficient h_o, thermal conductivity K and ambient temperature u_∞ are given as follows.

$$h_o = 50 \quad W/m^2.{}^oC \quad ,$$

$$K = 1 \quad W/m.{}^oC \quad ,$$

and

$$u_\infty = 20^oC \quad .$$

TABLE 3.3 - Results for the Problem 1c.

BIEM: 16 Constant Boundary Elements (CPU = 2.012 sec.)

FDM : 81 Grid Points (CPU = 0.915 sec.)

BOUNDARY NODES	COORDINATES (m)		TEMPERATURE u (^oC)		FLUX $q = \partial u/\partial n$ ($^oC/m$)
	X	Y	FDM (APP.D)	BIEM	BIEM
1	0.125	0.000	87.239	87.945	0.000
3	0.625	0.000	40.407	40.385	0.000
5	1.000	0.125	10.000	10.000	-83.223
7	1.000	0.625	10.000	10.000	-59.829
9	0.875	1.000	19.027	19.253	37.356
11	0.375	1.000	22.221	22.040	-101.982
13	0.000	0.875	100.000	100.000	434.079
15	0.000	0.375	100.000	100.000	111.574

INTERNAL POINTS	COORDINATES (m)		TEMPERATURE u (^oC)	
	X	Y	FDM	BIEM
1	0.250	0.250	73.748	74.039
3	0.750	0.750	23.442	23.359
5	0.500	0.500	45.999	46.062

Sample Problem 2

Consider the steady-state heat conduction in a square region with heat generation. The mathematical formulation of the heat conduction problem is

$$\nabla^2 u + \frac{q'''}{K} = 0 \qquad \text{in} \quad 0 \le x \le 1 \quad , \qquad 0 \le y \le 1$$

where q''' is the volumetric heat generation and K is the thermal conductivity.

Find the temperature at the internal and boundary points and the flux at boundary points for each of the cases shown below.

(a)

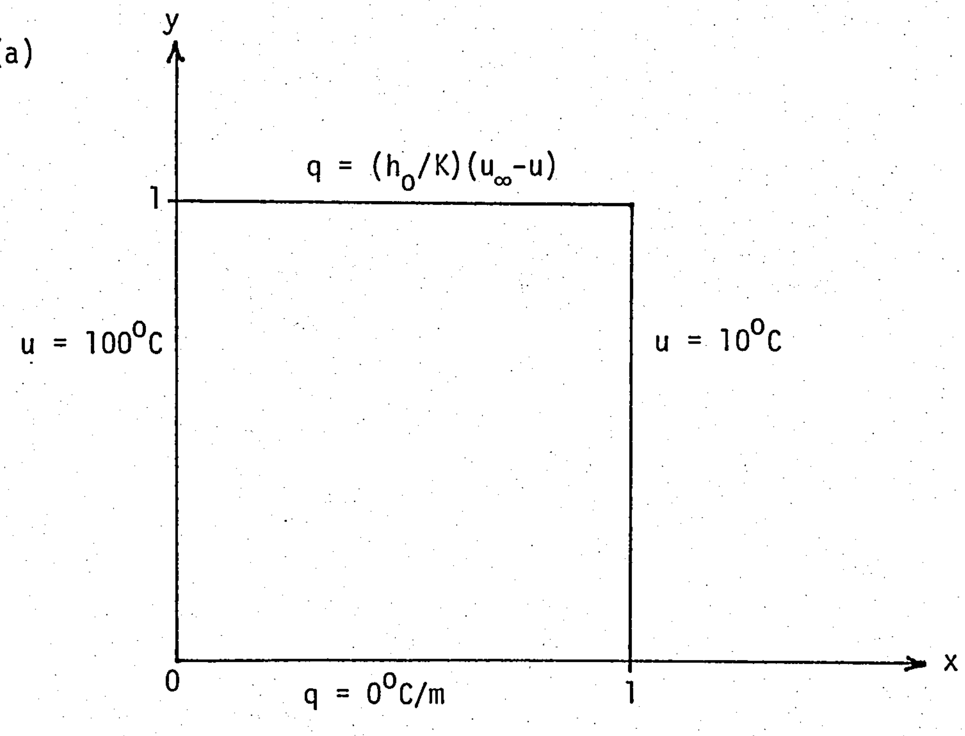


Figure 3.4 - Boundary conditions for the problem 2a.

In this case, the convective heat transfer coefficient h_o, thermal conductivity K, ambient temperature u_∞ and volumetric heat generation q''' are given as follows.

$h_o = 50\ W/m^2.{}^oC$, $u_\infty = 20^oC$,

$K = 4\ W/m.{}^oC$,   and $q''' = 100\ 000\ W/m^3$.

TABLE 3.4 - Results for the Problem 2a.

BIEM: 20 Linear Boundary Elements and 32 Internal Triangular Elements (CPU = 8.396 sec.)
FDM : 81 Grid Points (CPU = 1.189 sec.)

BOUNDARY NODES	COORDINATES (m)		TEMPERATURE u (^oC)		FLUX $q = \partial u/\partial n$ ($^oC/m$)
	X	Y	FDM (APP.D)	BIEM	BIEM
1	0.001	0.000	100.0	165.4	0.0
3	0.500	0.000	2246.0	2963.5	0.0
5	0.999	0.000	10.0	76.6	0.0
7	1.000	0.250	10.0	10.0	-11471.9
9	1.000	0.750	10.0	10.0	- 8938.7
11	0.999	1.000	10.0	35.1	-188.7
13	0.500	1.000	634.6	645.8	- 7823.1
15	0.001	1.000	100.0	111.7	- 1146.1
17	0.000	0.750	100.0	100.0	- 8847.3
19	0.000	0.250	100.0	100.0	-11307.9

INTERNAL POINTS	COORDINATES (m)		TEMPERATURE u (^oC)	
	X	Y	FDM	BIEM
1	0.250	0.250	2203.2	2189.0
3	0.750	0.750	1519.0	1528.6
5	0.500	0.500	2604.3	2605.5

(b)

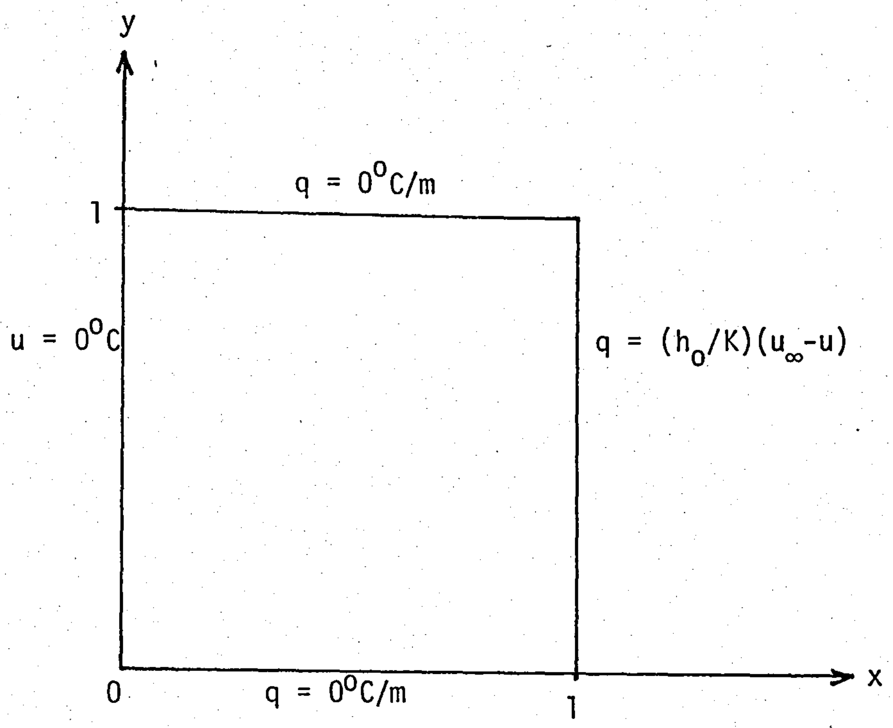


Figure 3.5 - Boundary conditions for the problem 2b.

In this case, the convective heat transfer coefficient h_o, thermal conductivity K, ambient temperature u_∞ and volumetric heat generation q"' are given as follows.

$h_o = 0.1\ W/m^2.^oC$,

$K = 1\ W/m.^oC$,

$u_\infty = 1\ ^oC$,

and $q''' = 10\ W/m^3$.

It should be noted that linear boundary elements are used for boundary integral element method solution. This problem is solved for 12 different configurations as shown in Figure 3.6. The number at each corner in each of the configuration is located on the side where to the node is assumed to belong.

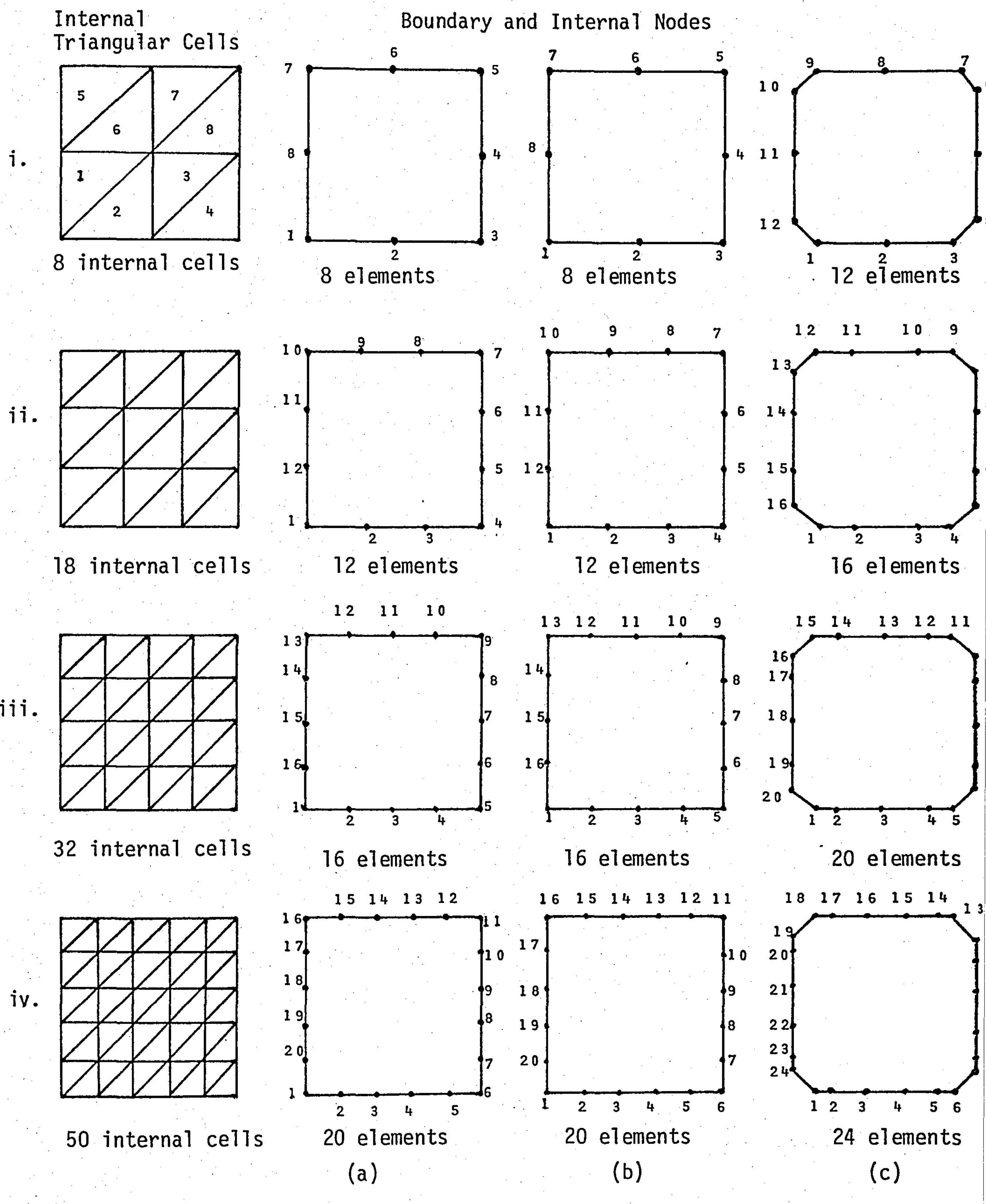


Figure 3.6 - The alternative forms of the nodes and the internal cells for the problem 2b.

TABLE 3.5 - Results for the Problem 2b.

NODES	COORDINATES (m)		TEMPERATURE u (^oC)
	X	Y	EXACT (APP.B)
1	0.25	0.25	2.10
2	0.75	0.25	4.41
3	0.75	0.75	4.41
4	0.25	0.75	2.10
5	0.50	0.50	3.57
6	0.50	0.00	3.57
7	1.00	0.50	4.64
8	0.50	1.00	3.57

NODES	T E M P E R A T U R E u (^oC)		
	BIEM. i-a. CPU= 1.771 sec.	BIEM. i-b. CPU= 1.799 sec.	BIEM. i-c. CPU= 3.087 sec.
1	3.47	1.88	2.05
2	5.74	3.91	4.26
3	5.74	3.91	4.26
4	3.47	1.87	2.05
5	4.81	3.15	3.41
6	4.92	3.06	3.46
7	5.89	4.08	4.42
8	4.92	3.06	3.46

NODES	T E M P E R A T U R E u (^oC)		
	BIEM. ii-a. CPU= 4.297 sec.	BIEM ii-b. CPU= 4.373 sec.	BIEM. ii-c. CPU= 5.477 sec.
1	2.70	1.93	2.04
2	5.02	4.12	4.32
3	5.01	4.11	4.31
4	2.71	1.94	2.05
5	4.16	3.37	3.52
6	4.11	3.16	3.38
7	5.24	4.34	4.54
8	4.11	3.16	3.38

Table 3.5 continued...

NODES	TEMPERATURE u (^0C)		
	BIEM. iii-a. CPU= 7.032 sec.	BIEM. iii-b. CPU= 6.331 sec.	BIEM. iii-c. CPU= 7.797 sec.
1	2.42	1.99	2.07
2	4.75	4.23	4.36
3	4.75	4.23	4.36
4	2.42	1.99	2.07
5	3.88	3.43	3.53
6	3.93	3.42	3.54
7	4.97	4.45	4.58
8	3.93	3.42	3.54

NODES	TEMPERATURE u (^0C)		
	BIEM. iv-a. CPU= 11.890 sec.	BIEM. iv-b. CPU= 10.543 sec.	BIEM. iv-c. CPU= 12.112 sec.
1	2.29	2.02	2.08
2	4.64	4.28	4.38
3	4.63	4.28	4.38
4	2.30	2.02	2.08
5	3.78	3.48	3.55
6	3.76	3.41	3.50
7	4.86	4.51	4.60
8	3.76	3.41	3.50

Sample Problem 3

Consider the steady-state heat conduction with heat generation in part of a cross-section of an industrial furnace shown in Figure 3.7 is considered. The mathematical formulation of the problem is

$$\nabla^2 u + \frac{q'''}{K} = 0 \quad .$$

Surfaces AB and DE are thermally insulated. There is heat exchange on surfaces AF and FE by convection with medium having temperature of $u_\infty = 500^\circ K$. The convective heat transfer coefficient h_0, thermal conductivity K and volumetric heat generation q"' are assumed to be constant and given as follows.

$h_0 = 40 \quad W/m^2.^\circ K \quad , \quad$ and $\quad q''' = 2 \times 10^3 \quad W/m^3$.

$K = 2 \quad W/m.^\circ K \quad ,$

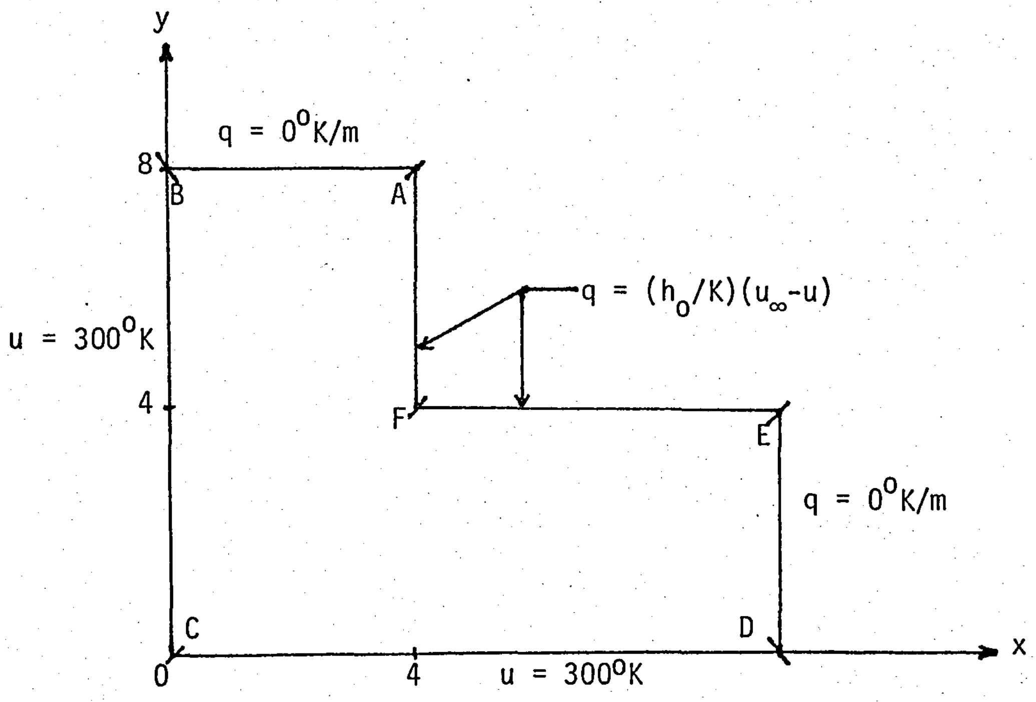


Figure 3.7 - Boundary conditions for the problem 3.

TABLE 3.6 - Results for the Problem 3.

BIEM: 23 Linear Boundary Elements and 28 Internal Triangular Elements (CPU = 9.951 sec.)
FDM : 75 Grid Points (CPU = 0.470 sec.)

BOUNDARY NODES	COORDINATES (m)		TEMPERATURE u (^oK)		FLUX $q = \partial u/\partial n$ ($^oK/m$)
	X	Y	FDM (APP.D)	BIEM	BIEM
1	0.001	0.000	300.0	300.0	-433.4
3	4.000	0.000	300.0	300.0	-2162.8
5	8.000	0.000	300.0	300.0	-2013.6
7	10.000	0.001	300.0	391.2	0.0
9	10.000	3.999	596.5	686.0	0.0
11	8.000	4.000	596.7	595.1	-1902.4
13	4.000	4.000	645.7	678.4	-3568.8
15	4.000	7.999	597.0	605.5	-2110.4
17	2.000	8.000	2467.0	2507.6	0.0
19	0.000	7.999	300.0	300.0	-2236.3
21	0.000	4.000	300.0	300.0	-2168.5
23	0.000	0.001	300.0	300.0	- 433.4

INTERNAL POINTS	COORDINATES (m)		TEMPERATURE u (^oK)	
	X	Y	FDM	BIEM
1	2.000	6.000	2491.0	2448.2
3	2.000	2.000	2450.0	2483.2
5	6.000	2.000	2489.0	2477.1

Sample Problem 4

Consider the steady-state heat conduction ina square region without heat generation. The mathematical formulation of the problem is

$$\nabla^2 u = 0 \quad \text{in} \quad 0 \leq x \leq 1 \quad , \quad 0 \leq y \leq 1 \quad .$$

Find the temperature at the internal and boundary points and the flux at the boundary points for each of the cases as shown below.

(a)

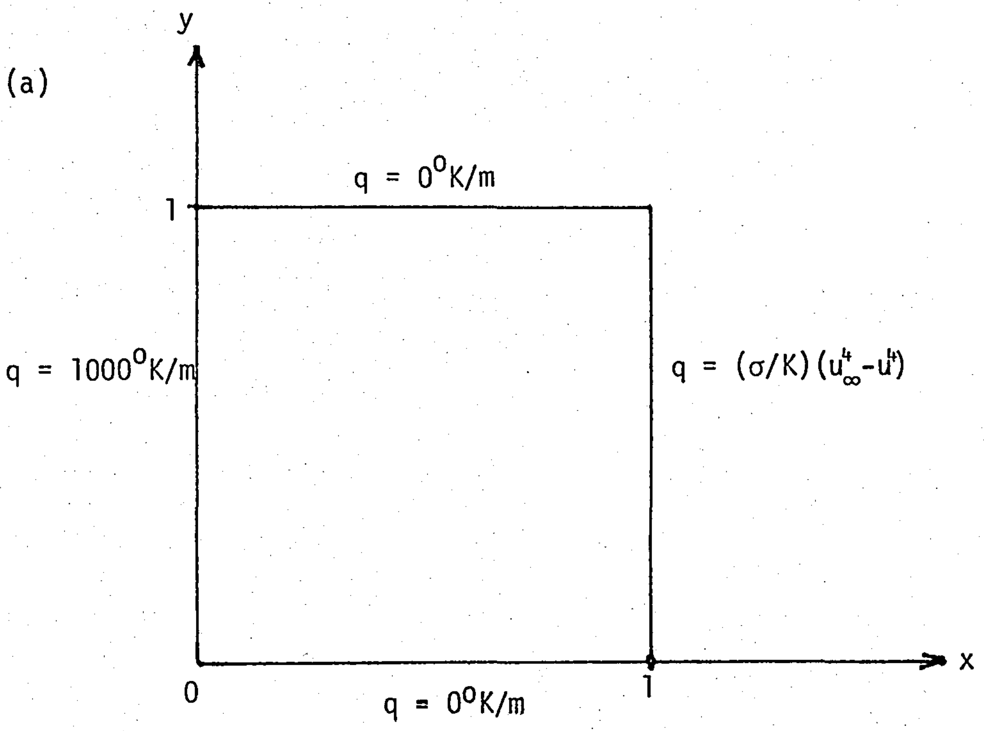


Figure 3.8 - Boundary conditions for the problem 4a.

In this case, Stephan-Boltzmann constant σ, thermal conductivity K and the ambient temperature u_∞ are given as follows.

$\sigma = 5.6697\text{x}10^{-8} \quad W.m^2/^0K^4$,

$K = 1 \quad W/m.^0K$,

and $u_\infty = 350 \quad ^0K$.

TABLE 3.7 - Results for the Problem 4a.

BIEM: 20 Linear Boundary Element (CPU = 15.725 sec.)
Iteration Number = 4

BOUNDARY NODES	COORDINATES (m)		TEMPERATURE u (^oK)		FLUX $q = \partial u/\partial n$ ($^oK/m$)	
	X	Y	EXACT (APP.B)	BIEM	EXACT	BIEM
1	0.001	0.000	1424.0	1418.6	0.0	0.0
3	0.500	0.000	925.0	924.4	0.0	0.0
5	0.999	0.000	426.0	430.1	0.0	0.0
7	1.000	0.250	425.0	425.1	-1000.0	-1000.7
9	1.000	0.750	425.0	425.1	-1000.0	-1000.7
11	0.999	1.000	426.0	430.1	0.0	0.0
13	0.500	1.000	925.0	924.4	0.0	0.0
15	0.001	1.000	1424.0	1418.6	0.0	0.0
17	0.000	0.750	1425.0	1423.6	1000.0	1000.0
19	0.000	0.250	1425.0	1423.6	1000.0	1000.0

INTERNAL POINTS	COORDINATES (m)		TEMPERATURE u (^oK)	
	X	Y	EXACT	BIEM
1	0.250	0.250	1175.0	1173.9
3	0.750	0.750	675.0	674.8
5	0.500	0.500	925.0	924.4

(b)

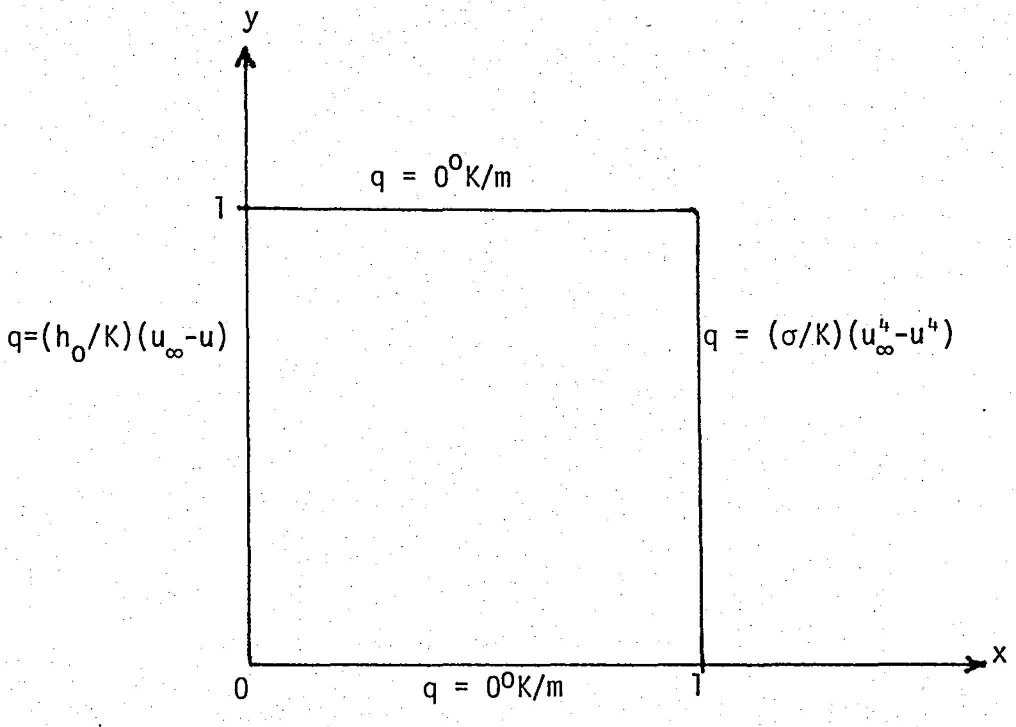


Figure 3.9 - Boundary conditions for the problem 4b.

In this case, Stephan-Boltzmann constant σ, thermal conductivity K, convective heat transfer coefficient h_0 and the ambient temperature u_∞ are given as follows.

$$\sigma = 5.6697\text{x}10^{-8} \quad \text{W.m}^2/{}^0\text{K}^4 \quad ,$$

$$K = 1 \quad \text{W/m.}^0\text{K} \quad ,$$

$$h_0 = 20 \quad \text{W/m}^2.{}^0\text{K} \quad ,$$

and

$$u_\infty = \begin{cases} 600^0\text{K} & \text{at } x = 0 \\ 300^0\text{K} & \text{at } x = 1 \end{cases} \quad .$$

TABLE 3.8 - Results for the Problem 4b.

BIEM: 20 Linear Boundary Elements (CPU = 15.573 sec.)
Iteration Number = 4

BOUNDARY NODES	COORDINATES (m)		TEMPERATURE u (^oK)		FLUX q = ∂u/∂n ($^oK/m$)	
	X	Y	EXACT (APP.B)	BIEM	EXACT	BIEM
1	0.001	0.000	587.1	586.1	0.0	0.0
3	0.500	0.000	461.1	461.1	0.0	0.0
5	0.999	0.000	335.0	336.1	0.0	0.0
7	1.000	0.250	334.8	334.8	-252.6	-253.1
9	1.000	0.750	334.8	334.8	-252.6	-253.1
11	0.999	1.000	335.0	336.1	0.0	0.0
13	0.500	1.000	461.1	461.1	0.0	0.0
15	0.001	1.000	587.1	586.1	0.0	0.0
17	0.000	0.750	587.4	587.3	252.6	253.1
19	0.000	0.250	587.4	587.3	252.6	253.1

INTERNAL POINTS	COORDINATES (m)		TEMPERATURE u (^oK)	
	X	Y	EXACT	BIEM
1	0.250	0.250	524.2	524.2
3	0.750	0.750	397.9	397.9
5	0.500	0.500	461.1	461.1

(c)

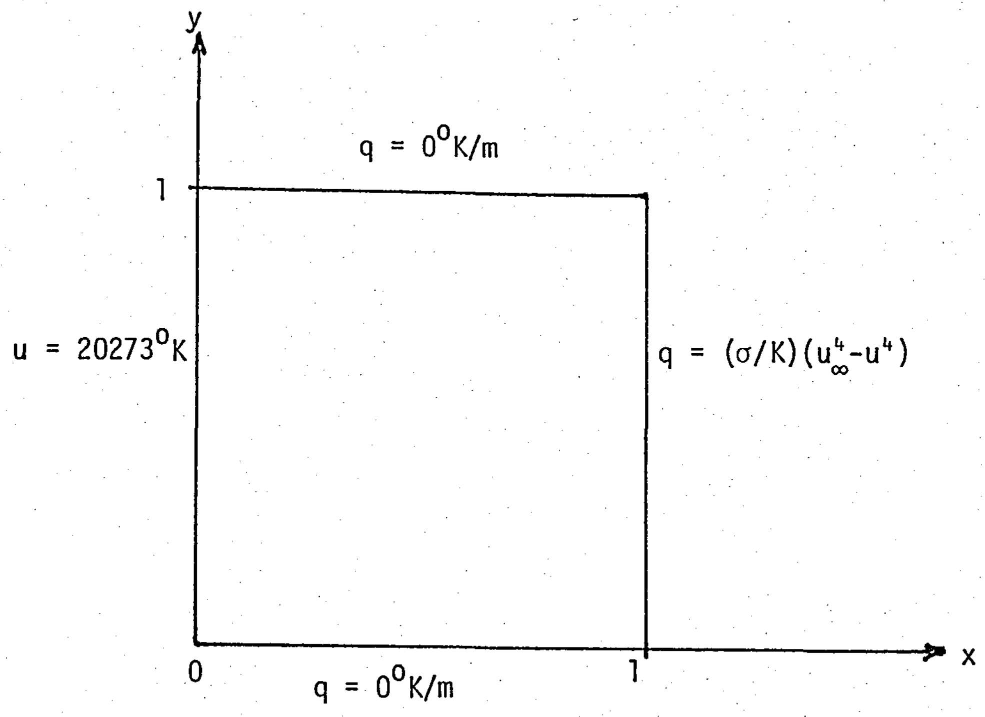


Figure 3.10 - Boundary conditions for the problem 4c.

In this case, Stephan-Boltzmann constant σ, thermal conductivity K and the ambient temperature u_∞ are given as follows.

$\sigma = 5.6697 \times 10^{-8}$ W.m^2/^{0}K^4 ;

K = 1 W/m.^{0}K ,

and

$u_\infty = 273^0$K .

TABLE 3.9 - Results for the Problem 4c.

BIEM: 20 Linear Boundary Elements (CPU = 40.112 sec.)
Iteration Number = 10

BOUNDARY NODES	COORDINATES (m)		TEMPERATURE u (^oK)		FLUX q = ∂u/∂n ($^oK/m$)	
	X	Y	EXACT	BIEM	EXACT	BIEM
1	0.001	0.000	20253.0	20173.7	0.0	0.0
3	0.500	0.000	10517.9	10521.1	0.0	0.0
5	0.999	0.000	782.3	868.4	0.0	0.0
7	1.000	0.250	762.8	769.4	-19510.2	-19555.0
9	1.000	0.750	762.8	769.4	-19510.2	-19555.0
11	0.999	1.000	782.3	868.4	0.0	0.0
13	0.500	1.000	10517.9	10521.1	0.0	0.0
15	0.001	1.000	20253.0	20173.7	0.0	0.0
17	0.000	0.750	20273.0	20273.0	19510.2	19558.0
19	0.000	0.250	20273.0	20273.0	19510.2	19558.0

INTERNAL POINTS	COORDINATES (m)		TEMPERATURE u (^oK)	
	X	Y	EXACT	BIEM
1	0.250	0.250	15395.4	15395.1
3	0.750	0.750	5640.3	5647.1
5	0.500	0.500	10517.9	10521.1

Sample Problem 5

Consider the steady-state heat conduction without heat generation in a hollow cylinder as shown in Figure 3.11, where the domain is given as follows.

$1 \leq r \leq 2$

Find the steady-state flux at the boundaries while the boundary surfaces at r_1 = 1 m and r_2 = 2 m are kept at uniform temperatures $u_1 = 100^oC$ and $u_2 = 20^oC$, respectively.

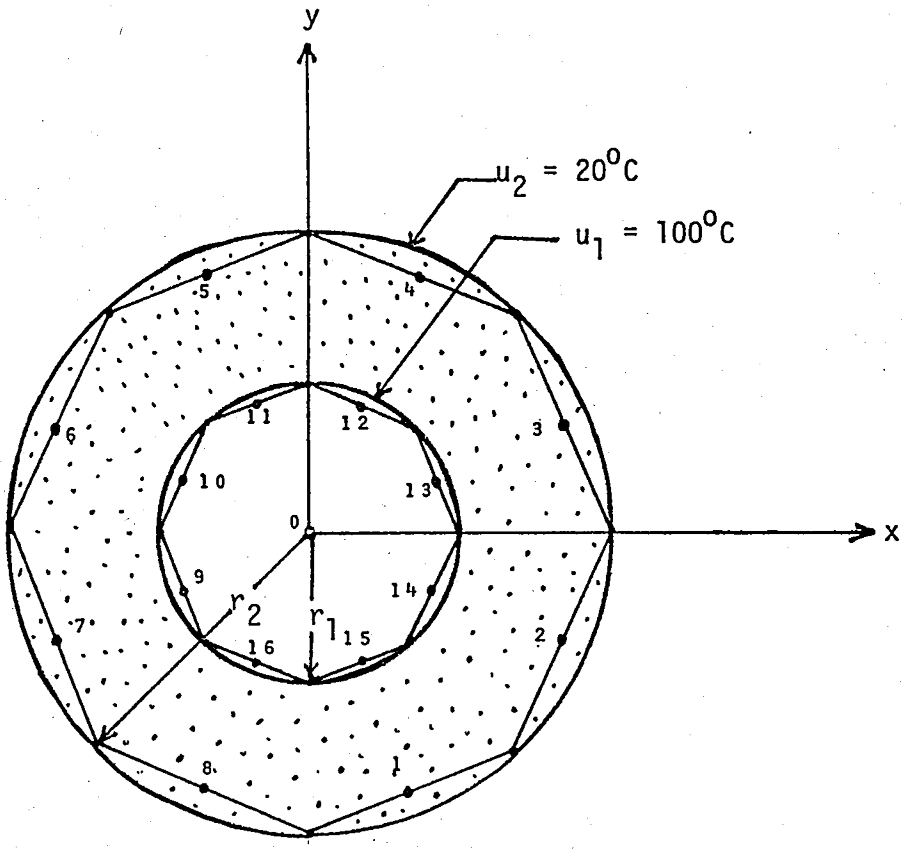


Figure 3.11 - Boundary conditions and boundary elements for the problem 5.

TABLE 3.10 - Results for the Problem 5.

BIEM: 16 Constant Boundary Elements (CPU = 2.623 sec.)				
BOUNDARY NODES	COORDINATES (m)		FLUX $q = \partial u/\partial n$ ($^oC/m$)	
	X	Y	EXACT (APP.B)	BIEM
1	0.707	-1.707	-62.463	-58.352
4	0.707	1.707	-62.463	-58.352
7	-1.707	-0.707	-62.463	-58.352
10	-0.854	0.354	124.925	120.243
13	0.854	0.354	124.925	120.243
16	-0.354	-0.854	124.925	120.243

INTERNAL POINTS	COORDINATES (m)		TEMPERATURE u (^oC)	
	X	Y	EXACT	BIEM
1	1.386	-0.574	53.203	44.888
3	0.574	1.386	53.203	44.888

IV. DISCUSSION OF THE RESULTS

In formulation of the steady-state heat conduction in the domain, the internal heat generation is an important term, since it causes to take the internal elements into consideration because of the domain integral which exists only in this case. However, it should be also remembered that the internal elements are regarded just because of a numerical technique that simplifies the evaluation of the domain integral easily. Thus, the internal heat generation does not create complexity.

Several problems were solved to test the validity and performance of our study where available exact solutions were used for some problems whereas the remaining problems were solved by finite difference technique. It may be seen that the results obtained by the boundary integral element method are in good agreement with the corresponding results obtained by exact solution and finite difference method, at the boundaries. However, this is not the case at interior points near the boundary. At these locations, the results are less accurate than the results on the boundary nodes. This is due to the fact that the numerical accuracy decreases as the distance r between the 'source' point and the 'observation' point goes to zero.

If linear boundary elements are used, unfortunately the corner points can have two values for the temperature or flux depending on the side under consideration as shown in Figure 4.1.

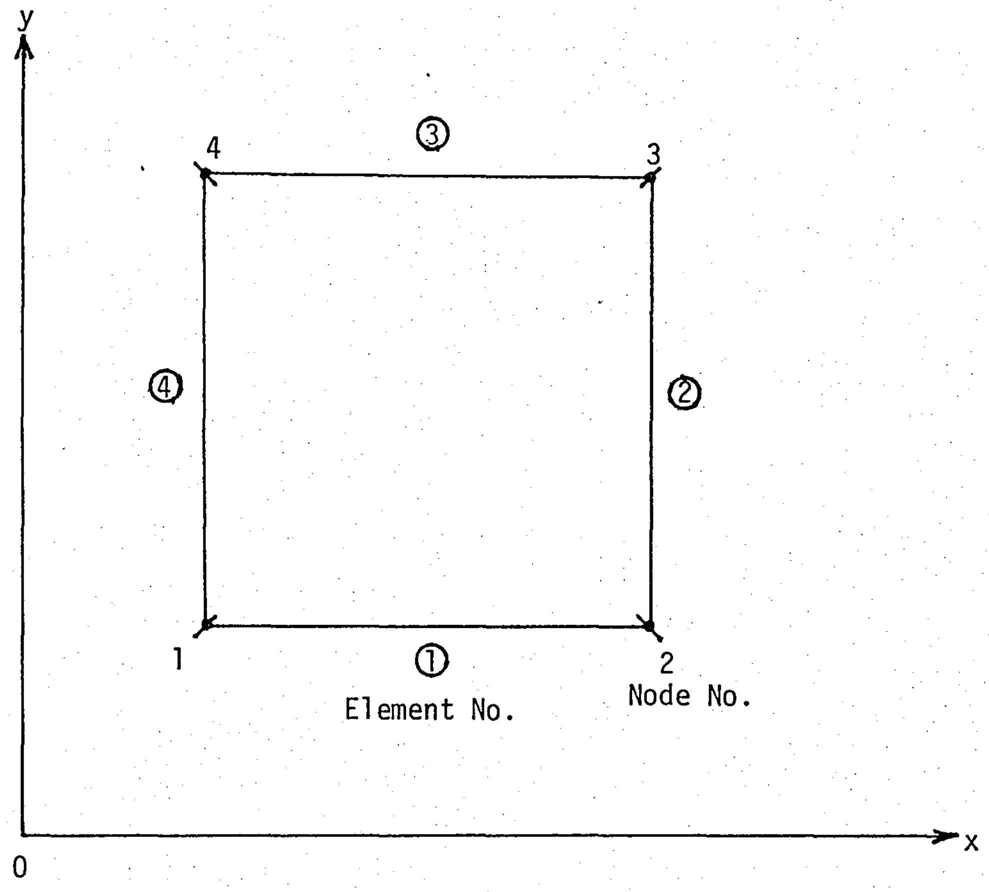


Figure 4.1 - The nodes at the corners.

A simple way to avoid the corner problem is to assume that there are two points very near to each other but which belong to different sides as shown in Figure 4.2. The two points near the corner are joined by a line segment which is considered as one of the elements approximating the boundary contour. It should be noted that the length of the line segment is taken to be too small to let the nodes to have different results. The results obtained by this method show a good agreement with the exact results.

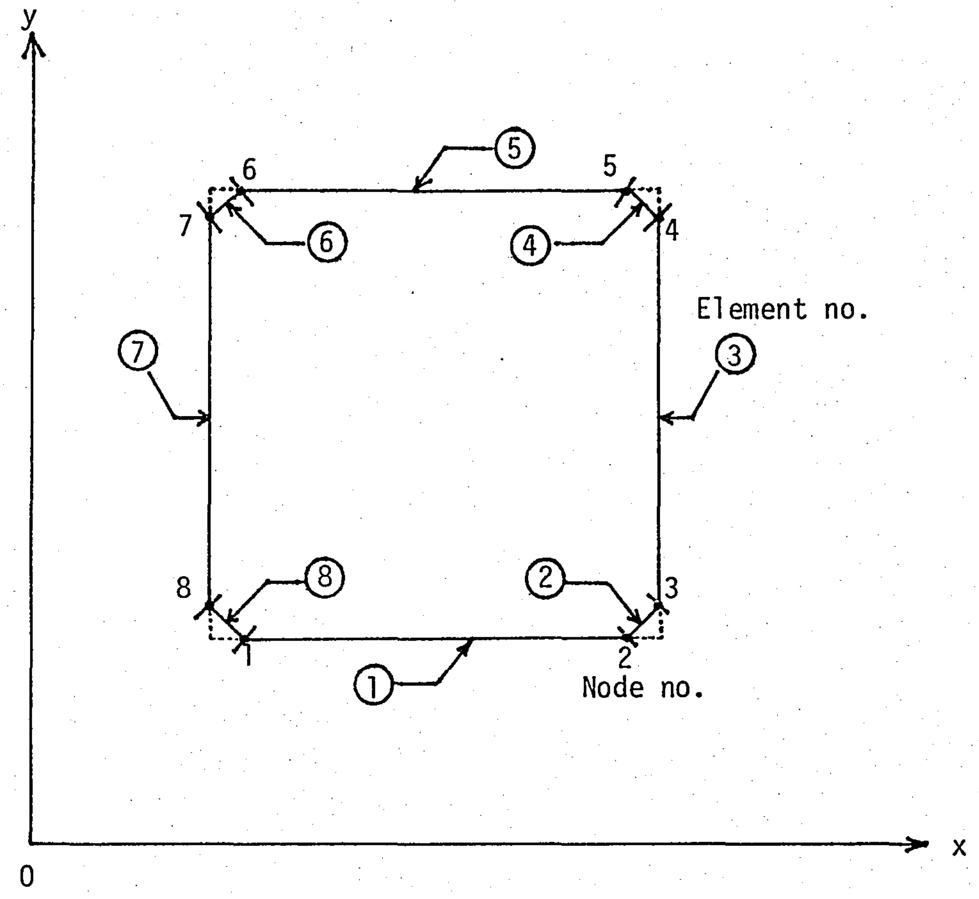


Figure 4.2 - The nodes near the corners.

For the case of radiative boundary conditions, one dimensional problems of simple square plate are selected in order to make a comparison to exact solutions. The nonlinearity of this kind of boundary condition creates difficulty. However, this difficulty is overcome by a linearizing technique. There was no problem of convergency, and an error of 10 percent in temperature is observed in the case of sample problem (5c), which is the largest error with respect to the errors of other two cases (5a) and (5b).

The method was also used to study problems with more than one surface, such as the case of hollow cylinder. The results at interior points are less accurate than the results at the boundary nodes. It should be noted that this inaccuracy is a result of the approximation of the actual boundary contour by finite segments.

V. CONCLUSIONS AND RECOMMENDATIONS

The solutions for the examples indicate that the present boundary integral element method is accurate and general for solving most of the conduction problems of practical importance.

The iterative boundary integral element method has been shown to be appropriate for use in numerically solving a variety of steady-state heat conduction problems.

The boundary integral element method in its present form has no inherent limitations as to the geometric complexity, kind of boundary condition.

The method is most suitable for calculating temperature and flux at the system boundaries and at a few individual interior points. This feature makes the method superior in this respect to available numerical methods, where the solution involves all interior points.

As in most of the practical calculations of heat transfer, boundary fluxes and temperatures are the only needed information. However, complete temperature distribution is directly obtainable with minimum effort.

Another primary advantage of using the boundary integral equation for the numerical solutions rather than the original differential equation is the space reduction of the problem. If the problem is three-dimensional in space, the boundary integral equation is a two-dimensional one which requires less effort and time for its solution.

It should be mentioned here that temperatures calculated at interior points near the boundary are generally not very accurate. This, of course, does not represent a drawback for the method, since temperatures and fluxes at the boundary are obtainable directly without reference to the interior point.

If linear elements are used in the boundary, the corner problem appears. This problem is solved by assuming that there are two points very near each other but which belong to different sides.

Although the examples cited in the present work are all of a two-dimensional nature, the method is also suitable for three-dimensional cases.

Finally, if the present method can be extended to include the transient heat conduction problems and to some specific problems of convective heat transfer, this will in turn make the boundary integral element method a more competitive numerical technique over the already existing methods.

REFERENCES

1. Morse, P.M. and Feshbach, H., Methods of Theoretical Physics, Part II, McGraw-Hill, New York, 1961.

2. Jawson, M.A., "Integral Equation Methods in Potential Theory", Proceedings Royal Society, Vol. 275A, 1963, p. 23.

3. Macmillan, W.D., The Theory of Potential, Dover, New York, 1958.

4. Mikhlin, S.G., Integral Equations, Pergamon Press, Oxford, 1957.

5. Mikhlin, S.G., Multidimensional Singular Integrals and Integral Equations, Pergamon Press, Oxford, 1965.

6. Mikhlin, S.G., Approximate Solutions of Differential and Integral Equations, Pergamon Press, Oxford, 1965.

7. Cruse, T.A. and Rizzo, F.J., "Boundary Integral Equation Method", Computational Applications in Applied Mechanics, ASME Proceedings AMD-Vol. 11, 1975.

8. Jawson, M.A. and Symm, G.T., Integral Equation Methods of Potential Theory and Elastostatics, Academic Press, New York, 1970.

9. Alarkon, E. and Martin, A., "Boundary Elements in Potential and Elasticity Theory", Computers and Structures, Vol. 10, 1979, p. 351.

10. Rizzo, F.J. and Shippy, D.J., "A Method of Solution for Certain Problems of Transient Heat Conduction", AIAA Journal, Vol. 8, 1971, p. 2004.

11. Kreyszig, E., Advanced Engineering Mathematics, John Wiley and Sons, New York, 1979.

12. Wylie, C.A., Advanced Engineering Mathematics, McGraw-Hill, New York, 1975.

13. Aşkar, A., Method in Applied Algebra and Analysis, Boğaziçi University, Istanbul, 1981.

14. Green, C.D., Integral Equation Methods, Barnes and Noble, New York, 1969.

15. Brebbia, C.A., The Boundary Element Method for Engineers, Pentech Press, London, 1978.

16. Huebner, K.H., The Finite Element Method for Engineers, John Wiley and Sons, New York, 1975.

17. Carnahan, B., Luther, H.A., and Wilkes, J.O., Applied Numerical Methods, John Wiley and Sons, New York, 1969.

18. Khader, M.S. and Hanna, M.C., "An Iterative Boundary Integral Numerical Solution for General Steady Heat Conduction Problems", Transactions of the ASME, Vol. 103, Feb. 1981, p. 26.

19. Özişik, M.N., Basic Heat Transfer, McGraw-Hill, New York, 1977.

20. Forsythe, G.E., Malcolm, M.A. and Moler, C.B., Computer Methods for Mathematical Computations, Prentice-Hall, New Jersey, 1977.

21. Finlayson, B.A., The Method of Weighted Residuals and Variational Principles, Academic Press, New York, 1972.

APPENDICES

APPENDIX A

FUNDAMENTAL SOLUTION

The fundamental solution, u*, is the solution of the equation

$$L^*u^* = \begin{cases} 0 & \text{when} \quad r > 0 \\ -\delta_i & \text{when} \quad r = 0 \end{cases} , \tag{A.1}$$

where r is the distance from the point of application of the unit potential to the point under consideration as shown in Figure 2.2.2.

For r > 0, taking symmetry into consideration, the Laplace equation in polar coordinates becomes

$$-\frac{1}{r}\frac{d}{dr}\left(r\frac{du^*}{dr}\right) = 0 \tag{A.2}$$

or,

$$\frac{d}{dr}\left(r\frac{du^*}{dr}\right) = 0 \quad . \tag{A.3}$$

Integrating the Eq. (A.3) twice, we get

$$u^* = C_1 \ln r + C_2 \quad . \tag{A.4}$$

Now, let us seek C_1 and C_2 by integrating the equation

$$L^*u^* = -\delta \qquad (A.5)$$

over a disk of arbitrary small radius 'e' centered at (x,y) as shown in Figure A.1.

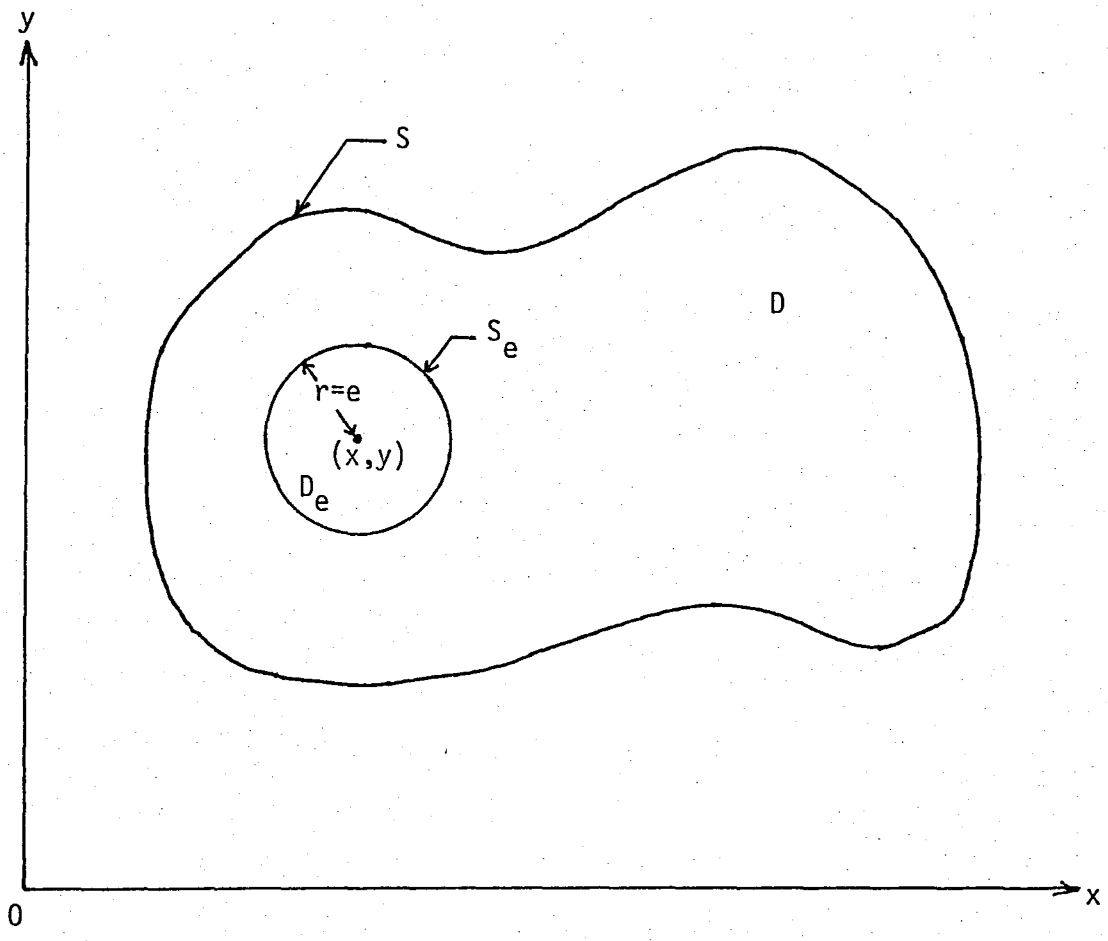


Figure A.1 - Point surrounded by a disk.

Then, the Eq. (A.1) can be written as follows.

$$\nabla^2 u^* = -\delta \qquad (A.6)$$

for $r \leq e$. Integrating the Eq. (A.6) over D_e, we get

$$\int_{D_e} \nabla^2 u^* dD = -\int_{D_e} \delta dD \quad . \qquad (A.7)$$

By using Green's first identity [11, p.451],

$$\int_D (\underset{\sim}{\nabla} a . \underset{\sim}{\nabla} b + a\nabla^2 b) dD = \int_S a \frac{\partial b}{\partial n} dS \quad , \qquad (A.8)$$

the right hand side of the Eq. (A.7) can be written as follows.

$$\int_{D_e} \nabla^2 u^* dD = \int_{S_e} (du^*/dr) dS \quad . \qquad (A.9)$$

Then, the left hand side of the Eq. (A.7) becomes

$$-\int_{D_e} \delta dD = \int_{S_e} (du^*/dr) dS \quad . \qquad (A.10)$$

From the property of the Dirac delta function,

$$\int_{D_e} \delta dD = 1 \quad , \qquad (A.11)$$

the Eq. (A.10) becomes

$$\int_{S_e} [du^*/dr]_{r=e} dS = -1 \quad . \qquad (A.12)$$

By using the Eq. (A.4), the Eq. (A.12) can be written as follows.

$$\int_{S_e} \frac{C_1}{e} dS = -1 \quad . \qquad (A.13)$$

From Figure A.1, we can see that

$$\int_{S_e} dS = 2\pi e \quad . \qquad (A.14)$$

Thus, the constant C_1 is found as follows.

$$C_1 = -\frac{1}{2\pi} \quad . \tag{A.15}$$

The other constant C_2 remains arbitrary and therefore we can set it equal to

$$C_2 = \frac{1}{2\pi} \ln 1 \tag{A.16}$$

$$= 0 \tag{A.17}$$

Thus, the fundamental solution becomes

$$u^* = \frac{1}{2\pi} \ln(1/r) \tag{A.18}$$

for two-dimensional Laplacian operator [15, p.48].

APPENDIX B
EXACT SOLUTIONS

Exact solution of the problem (1a)

The mathematical formulation of the problem is

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \quad \text{in} \quad 0 \leq x \leq 1 \;, \quad 0 \leq y \leq 1 \qquad \text{(B.1)}$$

$$u = y \quad \text{at} \quad x = 0 \;, \qquad \text{(B.2a)}$$

$$u = 1 - y \quad \text{at} \quad x = 1 \;, \qquad \text{(B.2b)}$$

$$u = x \quad \text{at} \quad y = 0 \;, \qquad \text{(B.2c)}$$

$$u = 1 - x \quad \text{at} \quad y = 1 \;. \qquad \text{(B.2d)}$$

For simplicity, let

$$u = C_1x + C_2y + C_3xy \;. \qquad \text{(B.3)}$$

It can be shown that the above expression for u satisfies the Laplace equation.

We can use the boundary conditions to find the unknown constants. The first boundary condition was

$$u(0,y) = y \;. \qquad \text{(B.4)}$$

Thus, inserting the Eq. (B.4) into Eq. (B.3) we get

$$y = C_2 y \quad . \tag{B.5}$$

So,

$$C_2 = 1 \quad . \tag{B.6}$$

Similarly, using the second boundary condition,

$$u(1,y) = 1 - y \quad , \tag{B.7}$$

we get

$$1 - y = C_1 + y(1 + C_3) \quad . \tag{B.8}$$

So,

$$C_1 = 1 \tag{B.9}$$

and

$$C_3 = -2 \quad . \tag{B.10}$$

Once the constants are found, one can write the complete expression for u, i.e.,

$$u = x + y - 2xy \quad . \tag{B.11}$$

It can be shown that the above expression for u satisfies the other boundary conditions, e.g. for the third boundary condition

$$u(x,0) = x \tag{B.12}$$

we have

$$\begin{aligned} x &= x + 0 - 2(0)x \\ &= x \quad . \end{aligned}$$

Also, for the fourth boundary condition

$$u(x,1) = 1 - x \tag{B.13}$$

we have

$$1 - x = x + 1 - 2(x)1$$
$$= 1 - x \quad .$$

Exact solution of the problem (1b)

The mathematical formulation of the problem is

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \qquad \text{in} \qquad 0 \leq x \leq 1 \quad , \quad 0 \leq y \leq 1 \tag{B.14}$$

$$u = 0 \qquad \text{at} \qquad x = 1 \quad , \tag{B.15a}$$

$$\partial u/\partial x = 0 \qquad \text{at} \qquad x = 0 \quad , \tag{B.15b}$$

$$\partial u/\partial y = 0 \qquad \text{at} \qquad y = 0 \quad , \tag{B.15c}$$

$$\partial u/\partial y = 20(1 - u) \qquad \text{at} \qquad y = 1 \quad . \tag{B.15d}$$

Let us use 'Separation of Variables' method [19,p.91] ,

$$u(x,y) = X(x)Y(y) \quad , \tag{B.16}$$

then the Eq. (B.1) becomes

$$\frac{X''}{X} = \frac{Y''}{Y} \quad . \tag{B.17}$$

We can write

$$X''/X = -\lambda^2 \tag{B.18a}$$

and

$$-Y''/Y = -\lambda^2 \tag{B.18b}$$

The solution of the Eq. (B.18a) is

$$X(x) = C_1 \sin\lambda_n(x) + C_2 \cos\lambda_n(x) \quad . \tag{B.19}$$

Applying the new forms of the boundary conditions (B.15a) and (B.15b),

$$X(1) = 0 \tag{B.20a}$$

and

$$X'(0) = 0 \quad , \tag{B.20b}$$

we get

$$X(x) = C_2 \cos\lambda_n x \tag{B.21}$$

where

$$\lambda_n = (\frac{2n + 1}{2})\pi \qquad n = 0,1,2,\ldots \quad . \tag{B.22}$$

The solution of the Eq. (B.18b) is

$$Y(y) = C_3 \sinh\lambda_n y + C_4 \cosh\lambda_n y \quad . \tag{B.23}$$

Applying the new form of the boundary condition (B.15c) ,

$$Y'(0) = 0 \quad , \tag{B.24}$$

we get

$$Y(y) = C_4 \cosh\lambda_n y \quad .$$

Then, the solution of the Eq. (B.14) is

$$u(x,y) = \sum_{n=0}^{\infty} A_n \cosh\lambda_n y \cos\lambda_n x \quad . \tag{B.25}$$

Using the boundary condition (B.15d), we get

$$A_n = \frac{40\sin\lambda_n}{\lambda_n(\lambda_n \sinh\lambda_n + 20\cosh\lambda_n)(1 + (1/2\lambda_n)\sin 2\lambda_n)} \quad . \tag{B.26}$$

Then the exact solution of the problem is

$$u(x,y) = \sum_{n=0}^{\infty} \left[\frac{(40\sin\lambda_n)\cos\lambda_n y\cos\lambda_n x}{\lambda_n(\lambda_n\sinh\lambda_n + 20\cosh\lambda_n)(1 + (1/2\lambda_n)\sin 2\lambda_n)}\right]. \qquad (B.27)$$

Here,

$$\lambda_n = \left(\frac{2n+1}{2}\right) \qquad n = 0,1,2,\ldots \quad .$$

Exact solution of the problem (2b)

The mathematical formulation of the problem is

$$\frac{d^2u}{dx^2} + 10 = 0 \qquad \text{in} \quad 0 \le x \le 1 \quad , \quad 0 \le y \le 1 \qquad (B.28)$$

$$u = 0 \qquad \text{at} \qquad x = 0 \quad , \qquad (B.29a)$$

$$du/dx = 0.1(1 - u) \qquad \text{at} \qquad x = 1 \quad . \qquad (B.29b)$$

Let us integrate the Eq. (B.28) two times. Then, the Eq. (B.28) becomes

$$u = -5x^2 + C_1x + C_2 \qquad (B.30)$$

Applying the boundary conditions (B.29a) and (B.29b), we get

$$C_1 = 106/11 \qquad (B.31a)$$

and

$$C_2 = 0 \quad . \qquad (B.31b)$$

Then the exact solution of the problem is

$$u = -5x^2 + \frac{106}{11}x \quad . \qquad (B.32)$$

Exact solution of the problem (4a)

The mathematical formulation of the problem is

$$\frac{d^2u}{dx^2} = 0 \qquad \text{in} \qquad 0 \leq x \leq 1 \quad , \quad 0 \leq y \leq 1 \qquad (B.33)$$

$$du/dx = 1000 \qquad \text{at} \qquad x = 0 \quad , \qquad (B.34a)$$

$$du/dx = 5.7\text{x}10^{-8}(350^4 - u^4) \qquad \text{at} \qquad x = 1 \quad . \qquad (B.34b)$$

Let us integrate the Eq. (B.33) two times. Then, the Eq. (B.33) becomes

$$u = C_1 + C_2 x \quad . \qquad (B.35)$$

Applying the boundary conditions (B.34a) and (B.34b), we get

$$C_1 = 1425 \qquad (B.36a)$$

and

$$C_2 = -1000 \quad . \qquad (B.36b)$$

Then the exact solution of the problem is

$$u = 1425 - 1000x \quad . \qquad (B.37)$$

Exact solution of the problem (4b)

The mathematical formulation of the problem is

$$\frac{d^2u}{dx^2} = 0 \qquad \text{in} \qquad 0 \leq x \leq 1 \quad , \quad 0 \leq y \leq 1 \qquad (B.38)$$

$$du/dx = 20(600 - u) \qquad \text{at} \qquad x = 0 \quad , \qquad (B.39a)$$

$$du/dx = 5.7\text{x}10^{-8}(300^4 - u^4) \qquad \text{at} \qquad x = 1 \quad . \qquad (B.39b)$$

Let us integrate the Eq. (B.38) two times. Then, the Eq. (B.38) becomes

$$u = C_1 + C_2 x \quad . \tag{B.40}$$

Applying the boundary conditions (B.39a) and (B.39b), we get

$$C_1 = 587.37 \tag{B.41a}$$

and

$$C_2 = -252.58 \quad . \tag{B.41b}$$

Then the exact solution of the problem is

$$u = 587.37 - 252.58x \quad . \tag{B.42}$$

Exact solution of the problem (4c)

The mathematical formulation of the problem is

$$\frac{d^2u}{dx^2} = 0 \qquad \text{in} \quad 0 \leq x \leq 1 \quad , \quad 0 \leq y \leq 1 \tag{B.43}$$

$$u = 20273 \qquad \text{at} \quad x = 0 \quad , \tag{B.44a}$$

$$du/dx = 5.7\text{x}10^{-8}(273^4 - u^4) \quad \text{at} \quad x = 1 \quad . \tag{B.44b}$$

Let us integrate the Eq. (B.43) two times. Then, the Eq. (B.43) becomes

$$u = C_1 + C_2 x \quad . \tag{B.45}$$

Applying the boundary conditions (B.44a) and (B.44b), we get

$$C_1 = 20273 \tag{B.46a}$$

and

$$C_2 = -19510.22 \quad . \tag{B.46b}$$

Then the exact solution of the problem is

$$u = 20273 - 19510.22x \quad . \qquad \text{(B.47)}$$

Exact solution of the problem (5)

The mathematical formulation of the problem [19,p.44] is

$$\frac{1}{r}\frac{d}{dr}[r\frac{du}{dr}] = 0 \qquad \text{in} \qquad 1 \leq r \leq 2 \qquad \text{(B.48)}$$

$$u = 20 \qquad \text{at} \qquad r = 2 \quad , \qquad \text{(B.49a)}$$

$$u = 100 \qquad \text{at} \qquad r = 1 \quad . \qquad \text{(B.49b)}$$

Let us integrate the Eq. (B.48) two times. Then, the Eq. (B.48) becomes

$$u = C_1 \ln r + C_2 \quad . \qquad \text{(B.50)}$$

Applying the boundary conditions (B.49a) and (B.49b), we get

$$C_1 = -80/\ln 2 \qquad \text{(B.51a)}$$

and

$$C_2 = 100 \quad . \qquad \text{(B.51b)}$$

Then the exact solution of the problem is

$$u = 100 - (80/\ln 2)\ln r \quad . \qquad \text{(B.52)}$$

APPENDIX C

COMPUTER PROGRAM FOR BOUNDARY INTEGRAL ELEMENT METHOD

This computer program solves the two-dimensional Poisson's equation ($\nabla^2 u + p = 0$) using constant or linear elements. Flowchart for the computer program can be seen in Figure C.1.

The main program defines the maximum dimensions of the system of equations which in this case is 40. It also allocates the input channel 5 and the output channel 6 for the Fortran statement. It calls the 11 following subroutines,

INPUT	:	Reads the program input.
GHCAL (GHCALC)	:	Computes GG and HH matrices for linear (constant) elements.
GHPCAL	:	Evaluates GGP and HHP matrices by reordering GG and HH matrices according to the type of the boundary condition at node 'i'.
BCAL	:	Calculates BB(I) for the source point (XSRCE, YSRCE).
INTE (INTEC)	:	Computes the integrals along a linear (constant) element which does not include the node under consideration.

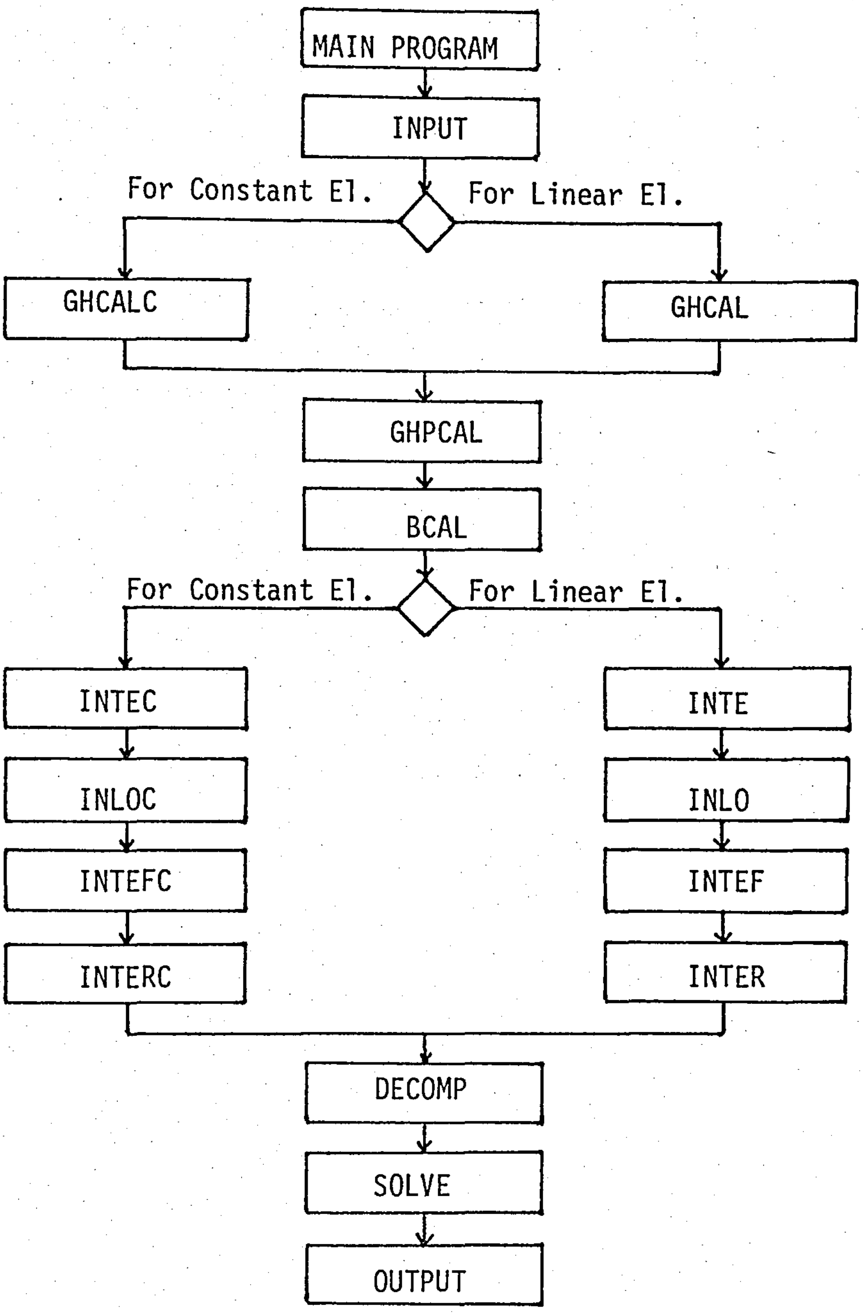


Figure C.1 - Flowchart for the computer program.

INLO (INLOC) :	Computes the integrals along a linear (constant) element including the node under consideration.
INTEF (INTEFC) :	Writes matrices GELEM (AH) and HELEM (BG) onto disc file (10) for internal points.
INTER (INTERC) :	Computes the temperature value at internal points for linear (constant) elements.
DECOMP and SOLVE :	Almost any computer library has subroutines based on variants of Gaussian elimination with partial pivoting for solving systems of simultaneous linear equations. The details of implementation of various subroutines available are quite different. These details can have important effects on the execution time of a particular subroutine, but if the subroutine is properly written, they should have little effect on its accuracy.

We can describe two such subroutines, DECOMP and SOLVE. DECOMP carries out that part of Gaussian elimination which depends only on the matrix. It saves the multipliers and the pivot information. SOLVE uses these results to obtain the solution for any right hand side.

DECOMP also returns an estimate of the condition of the matrix. Such an estimate is a much more reliable and useful measure of nearness to singularity than quantities such as the determinant or the smallest pivot.

The estimate is a lower bound for the actual condition, but it is computed in such a way that it is almost always within a factor of n of the actual condition, and it is usually much closer. In other words, for almost all matrices, DECOMP returns a quantity COND with

$$\frac{\text{cond}(A)}{n} \leq \text{COND} \leq \text{cond}(A) \quad .$$

In those situations where COND < cond(A)/n, it still measures the sensitivity of solutions for most right hand sides.

Roundoff error usually prevents DECOMP, or any other Gaussian elimination subroutine, from determining whether or not the input matrix is singular. If an exact zero pivot occurs during the elimination, DECOMP sets COND to 10^{32} to signal that it has detected singularity. The value 10^{32} is between BETAT and BETAU on all current floating-point systems, so it is between the reciprocal of the machine accuracy and the overflow level.

However, the occurrence of a zero pivot does not necessarily mean that the matrix is singular, nor does a singular matrix necessarily produce a zero pivot. In fact, the most common source of zero pivots is some kind of bug in the calling program.

It should be realized that, with partial pivoting, any matrix has a triangular factorization. DECOMP actually works faster when zero pivots occur. The only difficulty with a zero pivot is that SOLVE will divide by it during the back substitution. So SOLVE should not be used whenever DECOMP has set COND to a value much larger than BETAT.

Some of the subroutines available in computer libraries incorporate a technique as iterative improvement or iterative refinement.

This is a process which involves computation of the residual using high precision arithmetic and solution of a system of equations with the residual as the right hand side to obtain a correction for the computed solution. The corrected result often has a smaller error but does not necessarily have a smaller residual. Furthermore, the size of the correction is another measure of the sensitivity of the solution to errors in the data and the computation.

We decide against including an iterative improvement program for several reasons. First, the solution obtained without improvement is satisfactory for most applications. Second, the errors in the input data usually affect the solution more than the round-off introduced during its computation. Third, our condition estimator supplies the same kind of information available from the size of the correction. Finally, and possibly most important, the availability and use of the required high precision arithmetic varies from computer to computer. A general linear equation solver which efficiently incorporates iterative improvement cannot be written in standard Fortran.

To comment upon some details in DECOMP and SOLVE, we need to examine how Fortran systems store matrices. If a program contains statement,

```
DIMENSION A(3,5)
```

then

```
3*5 = 15
```

locations will be reserved in memory for the elements of A. They will be stored in the following order,

```
A(1,1) A(2,1) A(3,1) A(1,2) A(2,2) .......
```

In other words the elements of each column are stored together. The elements of each row are seperated from each other by a number of locations equal to the first subscript in the dimension statement.

Many of the common matrix operations are most naturally described in terms of rows. For example, in Gaussian elimination, a multiple of one row is subtracted from another row. When implemented in Fortran, such operations typically have the innermost loops varying the second index of arrays. This has two potentially adverse effects on program efficiency. Subscript calculations may be more costly because they involve information contained in the dimension statement. Operating systems which automatically move data between high speed and secondary memory units during computation may have to do an excessive amount of work. For these reasons, we have implemented Gaussian elimination in a somewhat unconventional manner with all the inner loops varying the first index. Such an implementation can be significantly more efficient with certain types of operating systems.

Most, but not all, Fortran dialects have provision for variable dimensions on arrays which are subroutine parameters. In a main program, one may specify

```
      DIMENSION A(40,40)
```

but intend to actually work with an N by N matrix where N may vary from problem to problem. Subroutines such as DECOMP and SOLVE need both N, the actual working order, and the quantity 40 used in the dimension statement because that is the memory increment between successive elements of a row. This dimension information is called NDIM in DECOMP and SOLVE [20, p.48].

OUTPUT	:	Outputs the results.

TERMINOLOGY

The general variables used by the program, together with their meaning are given below.

M	:	Number of different surfaces.
NC(K)	:	Last nodes in these surfaces.
LMICMI	:	Indicates the type of the elements. LMICM = 1 means that constant boundary elements are used. LMICMI = 2 means that linear boundary elements are used.
NONL	:	Indicates the type of boundary conditions at the element nodes. NONL = 1 means that there are nonlinear boundary conditions at the element nodes. NONL = 0 means that linear boundary conditions at the element nodes.
EPSMAX	:	Maximum tolerance for the iteration procedure.
NST	:	The first node which has nonlinear boundary condition.
NLA	:	The last node which has nonlinear boundary condition.
THC	:	Thermal conductivity.
N	:	Number of nodes.
KODE(I)	:	Indicates the type of boundary conditions at the element nodes. If KODE(I) = 1; then ALPHA(I) = 0, BETA(I) = value of 'temperature'. If KODE(I) = 2; then ALPHA(I) = 0, BETA(I) = value of 'flux'.

	If KODE(I) = 3; then ALPHA(I) = value of heat transfer coefficient, BETA(I) = ambient temperature. Note that flux is '+' if there is heat input to the region.
KODEP :	Check the source (heat generation) term. If KODEP = 0, then there is no source term. If KODEP = 1, then there is source term.
KODEI :	Checks if internal temperature needed.
LINT :	Number of internal points where the funtion is calculated.
NPOIN :	Number of points for internal elements.
CX, CY :	Internal point coordinates where the value of u is required.
X, Y :	Coordinates of the extreme points of the boundary elements.
NELEM :	Number of internal elements.
EXISP,ETASP:	Numerical integration points for internal triangles.
WEIGP :	Weights for internal triangles.

```
C*****************************************************************
C
C     PROGRAM BIEM
C
C     SOLVES 2-DIMENSIONAL POISSON'S EQUATION
C        BY THE BIEM
C
C     LAPLACIAN ( U ) + P = 0
C
C
C     LINEAR OR CONSTANT VARIATION
C        ALONG THE SEGMENTS
C
C
C
C
C
      COMMON/BEM1/X(41),Y(41),CX(9),CY(9),SOL(9),BB(40),M,NC(5)
      COMMON/BEM2/GG(40,40),HH(40,40),KODE(40),FI(41),DFI(41)
      COMMON/BEM3/NDIM,N,LINT,NI,NO,KODEI,KODEP,NPOIN,NELEM
      COMMON/BEM4/EXISP(7),ETASP(7),WEIGP(7),XM(40),YM(40)
      COMMON/BEM5/PTERM(40),LNODS(3,50),COORD(2,40)
      COMMON/BEM6/WORK(40),IPVT(40),COND,NONL,LMICMI
      COMMON/BEM7/GGP(40,40),HHP(40,40),ALPHA(40),BETA(40)
      COMMON/BEM8/EFI(41)
C
C     INITIALIZATION OF PROGRAM PARAMETERS
C     NDIM=MAX. DIMENSION OF THE SYSTEM OF EQUATIONS
C
C
      NDIM=40
C
C     ASSIGN DATA SET NUMBER FOR INPUT,NI AND OUTPUT,NO
C
      NI=5
      NO=6
C
C     INPUT
C
      CALL INPUT
C
C     CHECK NONL IF ANY NONLINEAR BOUNDARY CONDITION
C        IS PRESENT
C
      IF(NONL.NE.1)GO TO 7
C
C     A ITERATIVE SCHEME CAN BE USED TO SOLVE THE PROBLEMS
C        WHICH POSSESS NONLINEAR BOUNDARY CONDITIONS
C
C     EPSMAX=MAXIMUM TOLERANCE
C
C     THE ITERATION PROCEDURE STARTS BY TAKING
C        EFI(JK)=TAMB (AMBIENT TEMP.)
C
C     NST=THE FIRST NODE WHICH HAS NONLINEAR B.C.
C
C     NLA=THE LAST NODE WHICH HAS NONLINEAR B.C.
C
C     THC= THERMAL CONDUCTIVITY
C
C
      READ(NI,80)EPSMAX,TAMB,NST,NLA,THC
   80 FORMAT(2F10.0,2I5,F10.0)
      WRITE(NO,81)EPSMAX,TAMB,NST,NLA,THC
   81 FORMAT(//,5X,'EPSMAX=',F5.3,5X,'TAMB=',F5.1,5X,'NST=',
     .I3,5X,'NLA=',I3,5X,'THC=',F5.2,/)
C
C
      DO 11 JK=NST,NLA
   11 EFI(JK)=TAMB
      KK=NST
C
C     NOTE THAT WE CAN WRITE
C          DO 12 KK=NST,NLA
C     INSTEAD OF
C          KK=NST
C
      ITER=0
    3 ITER=ITER+1
      IF(KK.NE.NST)GO TO 1
      IF(ITER-1)1,2,1
    2 GOLD=TAMB
      GO TO 16
    1 GOLD=FI(KK)
   16 CONTINUE
C
C
C     THE NEW FORMS OF THE NONLINEAR BOUNDARY CONDITIONS
C
C
C
      DO 21 II=NST,NLA
      ALPHA(II)=0.04*5.6697*((EFI(II)/100.)**3)/THC
   21 BETA(II)=(3.*(EFI(II)**4.)+(TAMB**4.))/(4.*(EFI(II)**3.))
C
C
C
    7 CONTINUE
C
C
C
C
C     CHECK M AND LMICMI IF IT IS CONSTANT/LINEAR
C        VARIATION ALONG THE SEGMENTS
C
C
      IF(M-1)31,31,32
   31 IF(LMICMI-1)33,32,33
C
C     EVALUATE GG AND HH MATRICES FOR CONSTANT ELEMENTS
C
   32 CALL GHCALC
      GO TO 34
C
C     EVALUATE GG AND HH MATRICES FOR LINEAR ELEMENTS
C
   33 CALL GHCAL
C
C     EVALUATE GGP AND HHP MATRICES (BY REARRANGING)
C
   34 CALL GHPCAL
C
C
C     DECOMPOSE GG MATRIX BY USING DECOMP
C
C
C
C
C
C
      CALL DECOMP (NDIM,N,GGP,COND,IPVT,WORK)
C
C     PRINT THE CONDITION NO. OF THE COEFFICIENT MATRIX
C
      WRITE (NO,931) COND
      CONDP1=COND+1
      IF (CONDP1.EQ.COND) WRITE (NO,932)
      IF (CONDP1.EQ.COND) STOP
C
C     CLEAR THE BB VECTOR FOR THE SOURCE TERM
C
      DO 560 I=1,N
  560 BB(I)=0.
C
C     CHECK KODEP IF ANY SOURCE TERM IS PRESENT
C
      IF (KODEP.EQ.0) GO TO 778
C
C     COMPUTE THE BB VECTOR OF THE SOURCE TERM
C     LOOP OVER THE BOUNDARY NODES FOR CONSTANT/LINEAR ELEMENTS
C
      DO 510 I=1,N
C
C
C
      IF(M-1)51,51,52
   51 IF(LMICMI-1)53,52,53
   52 CALL BCAL (XM(I),YM(I),BB(I))
      GO TO 510
C
C
C
   53 CALL BCAL ( X(I), Y(I),BB(I))
  510 CONTINUE
```

```
C
C          EVALUATE THE DFI VECTOR
C          IT ORIGINALLY CONTAINS THE RHS OF THE EQUATIONS
C          AFTER SOLUTION,WILL CONTAIN THE VALUES OF THE SYSTEM OF EQNS
C
      778  DO 160 I=1,N
           DFI(I)=-BB(I)
           DO 160 J=1,N
           DFI(I)=DFI(I)+HHP(I,J)*BETA(J)
      160  CONTINUE
C
C          SOLVE FOR THE UNKNOWNS
C             (BY BACK-SUBSTITUTION)
C
           CALL SOLVE (NDIM,N,GGP,DFI,IPVT)
C
C          REORDER DFI VECTOR TO OBTAIN :
C             FI =BOUNDARY POTENTIAL (TEMP.) VALUES
C             DFI=BOUNDARY POTENTIAL DERIVATIVES (FLUX)
C
           DO 250 I=1,N
           GO TO (10,20,30),KODE(I)
       10  FI(I)=BETA(I)
           DFI(I)=DFI(I)
           GO TO 250
       20  FI(I)=DFI(I)
           DFI(I)=BETA(I)
           GO TO 250
       30  FI(I)=DFI(I)
           DFI(I)=ALPHA(I)*(BETA(I)-FI(I))
      250  CONTINUE
C
C          CHECK KODEI WHETHER TO EVALUATE AT THE INTERNAL POINTS
C
           IF (KODEI.EQ.0) GO TO 790
C
           IF(M-1)41,41,42
       41  IF(LMICMI-1)43,42,43
C
C          FOR CONSTANT ELEMENTS
C
C
       42  CALL INTEFC
           CALL INTERC
           GO TO 790
C
C          FOR LINEAR ELEMENTS
C
       43  CALL INTEF
           CALL INTER
      790  CONTINUE
C
           IF(NONL.NE.1)GO TO 12
C
C
C          THE PROCEDURE IS REPEATED AS BEFORE UNTIL A CONVERGENCE
C          CRITERION IN THE FORM ( HOLDT-GOLD ) REACHES A CERTAIN TOLER.
C
C
C
           DO 99 MM=KK,NLA
       99  FFI(MM)= FI(MM)
           HOLDT= FI(KK)
           EPS=ABS(HOLDT-GOLD)
           IF(ITER.GT.80) GO TO 13
           IF(EPS-EPSMAX)13,3,3
       13  CONTINUE
C
           WRITE(NO,101)ITER,EPS
      101  FORMAT(///,20X,'ITER =',I3,15X,'EPS =',E15.7,/)
C
C          PRINT THE RESULTS
C
C
       12  CONTINUE
C
           CALL OUTPUT
C
      931  FORMAT (2X,'CONDITION NO=',E15.5)
      932  FORMAT (2X,'MATRIX IS SINGULAR TO WORKING PRECISION')
C
```

```
      STOP
      END
C
C
C*******************************************************************
      SUBROUTINE INPUT
C
      COMMON/BEM1/X(41),Y(41),CX(9),CY(9),SOL(9),BB(40),M,NC(5)
      COMMON/BEM2/GG(40,40),HH(40,40),KODE(40),FI(41),DFI(41)
      COMMON/BEM3/NDIM,N,LINT,NI,NO,KODEI,KODEP,NPOIN,NELEM
      COMMON/BEM4/EXISP(7),ETASP(7),WEIGP(7),XM(40),YM(40)
      COMMON/BEM5/PTERM(40),LNODS(3,50),COORD(2,40)
      COMMON/BEM6/WORK(40),IPVT(40),COND,NONL,LMICMI
      COMMON/BEM7/GGP(40,40),HHP(40,40),ALPHA(40),BETA(40)
C
C     READ BASIC PARAMETERS
C     M=NUMBER OF DIFFERENT SURFACES
C     NC(K)=LAST NODES IN THESE SURFACES
C
C     LMICMI=1 ,CONSTANT BOUNDARY ELEMENTS
C     LMICMI=2 ,LINEAR BOUNDARY ELEMENTS
C
C
C     NONL=1 ,NONLINEAR BOUNDARY CONDITIONS
C     NONL=0 ,LINEAR BOUNDARY CONDITIONS
C
C
      READ(NI,200)M,LMICMI,NONL
C
      WRITE (NO,100)
  100 FORMAT (, ,,120(,*,))
C
      READ(NI,200)  (NC(K),K=1,M)
      WRITE(NO,201)M,LMICMI,NONL,(NC(K),K=1,M)
  201 FORMAT(//5X,,M=,,I1,5X,,LMICMI=,,I1,5X,,NONL=,,I1,//5X,
     .,NC(K):,,5(I5))
C
      READ(NI,200)N,KODEI,KODEP
      WRITE (NO,300)N,KODEI,KODEP
  300 FORMAT(/,5X,,N=,,I3,2X,,KODEI=,,I1,2X,,KODEP=,,I1)
  200 FORMAT(5I5)
C
C     CHECK IF INTERNAL POTENTIALS NEEDED
C
      IF (KODEI.EQ.0) GO TO 777
C
C     READ NO. OF INTERNAL POINTS AND COORDINATES
C
      READ (NI,115) LINT
  115 FORMAT (I5)
      DO 1 I=1,LINT
    1 READ (NI,400) J,CX(I),CY(I)
  400 FORMAT (I5,2F10.0)
C
C     READ COORDINATES OF EXTREME POINTS OF THE BOUNDARY
C     ELEMENTS IN ARRAY X AND Y
C
  777 WRITE (NO,500)
  500 FORMAT (//2X,,COORDINATES OF THE EXTREME POINTS OF THE ,,
     .    , BOUNDARY ELEMENTS,,//4X,,POINT,,10X,,X,,18X,,Y,)
      DO 10 I=1,N
      READ (NI,400) J,X(I),Y(I)
   10 WRITE (NO,700) I,X(I),Y(I)
  700 FORMAT (5X,I3,2(5X,E14.7))
C
C     READ BOUNDARY CONDITIONS
C
C     IF KODE(I)=1 ,ALPHA(I)=0.
C                   BETA(I)=VALUE OF TEMPERATURE
C     IF KODE(I)=2 ,ALPHA(I)=0.
C                   BETA(I)=VALUE OF  FLUX
C     IF KODE(I)=3 ,ALPHA(I)=VALUE OF HEAT TRANSFER COEFFICIENT
C                   BETA(I)=VALUE OF AMBIENT TEMPERATURE
C                   NOTE, Q=ALPHA*(BETA-U)
C
C
      WRITE (NO,800)
  800 FORMAT (//2X,,BOUNDARY CONDITIONS,//5X,,NODE(I),,3X,
     . ,KODE(I),,5X,,ALPHA(I),,13X,,BETA(I),)
      DO 20 I=1,N
      READ (NI,900) J,KODE(I),ALPHA(I),BETA(I)
  900 FORMAT (2I5,2F10.0)
```

```
      C
      C
      C
    20 WRITE (NO,950) I,KODE(I),ALPHA(I),BETA(I)
   950 FORMAT (5X,I3,8X,I1,8X,E14.7,6X,E14.7)
      C
      C     CHECK KODEP,
      C                  KODEP=0 ,NO SOURCE TERM
      C                  KODEP=1 ,SOURCE TERM
      C
           IF (KODEP.EQ.0) GO TO 999
      C
      C     READ,
      C          NPOIN=NO. OF POINTS FOR INTERNAL ELEMENTS
      C          NELEM=NO. OF INTERNAL ELEMENTS
      C
      C
           READ (NI,510) NPOIN,NELEM
   510 FORMAT (4I5)
      C
      C
      C     WRITE NPOIN AND NELEM
      C
           WRITE (NO,511)
   511 FORMAT (1H0,5X,,INTERNAL SOURCE TERM DATA,)
           WRITE (NO,515) NPOIN,NELEM
   515 FORMAT (1H0,6X,,NPOIN =,,I3,5X,,NELEM =,,I3)
      C
      C     READ AND WRITE NUMERICAL INTEGRATION POINTS
      C     AND WEIGHTS FOR INTERNAL ELEMENTS
      C
           WRITE (NO,560)
   560 FORMAT (1H0,7X,,EXISP,,10X,,ETASP,,10X,,WEIGP,)
           DO 565 IGAUS=1,7
           READ (NI,570) EXISP(IGAUS),ETASP(IGAUS),WEIGP(IGAUS)
   565 WRITE (NO,570) EXISP(IGAUS),ETASP(IGAUS),WEIGP(IGAUS)
   570 FORMAT (3F15.8)
      C
      C     READ AND WRITE THE TRIANGULAR INTERNAL ELEMENT
      C     NODAL CONNECTIONS
      C
      C
           WRITE (NO,520)
   520 FORMAT (1H0,2X,,EL,,3X,,NODES,)
           DO 525 JELEM=1,NELEM
           READ (NI,510) IELEM,(LNODS(INODE,JELEM),INODE=1,3)
   525 WRITE (NO,510)  JELEM,(LNODS(INODE,JELEM),INODE=1,3)
      C
      C     READ AND WRITE THE NODAL COORDINATES
      C     FOR THE INTERNAL TRIANGULAR ELEMENTS
      C
      C
           WRITE (NO,530)
   530 FORMAT (1H0,6X,,NODE,,6X,,X-COORD,,10X,,Y-COORD,)
           DO 535 JPOIN=1,NPOIN
           READ (NI,540) IPOIN,(COORD(IDIME,JPOIN),IDIME=1,2)
   535 WRITE (NO,540) JPOIN,(COORD(IDIME,JPOIN),IDIME=1,2)
   540 FORMAT (I10,2F15.5)
      C
      C     READ AND WRITE THE INTERNAL NODAL SOURCE VALUES,PTERM(I)
      C
      C
           WRITE (NO,545)
   545 FORMAT(1H0,6X,,NODE,,8X,,PTERM,)
           DO 550 JPOIN=1,NPOIN
           READ (NI,555) IPOIN,PTERM(JPOIN)
   550 WRITE (NO,555) JPOIN,PTERM(JPOIN)
   555 FORMAT (I10,F15.5)
      C
      C
   999 RETURN
           END
      C
      C*************************************************************************
           SUBROUTINE GHCAL
      C
           COMMON/BEM1/X(41),Y(41),CX(9),CY(9),SOL(9),BB(40),M,NC(5)
           COMMON/BEM2/GG(40,40),HH(40,40),KODE(40),FI(41),DFI(41)
           COMMON/BEM3/NDIM,N,LINT,NI,NO,KODEI,KODEP,NPOIN,NELEM
           COMMON/BEM4/EXISP(7),ETASP(7),WEIGP(7),XM(40),YM(40)
           COMMON/BEM5/PTERM(40),LNODS(3,50),COORD(2,40)
           COMMON/BEM6/WORK(40),IPVT(40),COND,NONL,LMICMI
      COMMON/BEM7/GGP(40,40),HHP(40,40),ALPHA(40),BETA(40)
      DIMENSION GELEM(2),HELEM(2)
C
C     CLEAR GG AND HH MATRICES
C
C
      DO 10 J=1,N
      DO 10 I=1,N
      GG(I,J)=0.
   10 HH(I,J)=0.
      X(N+1)=X(1)
      Y(N+1)=Y(1)
C
C     COMPUTE GG AND HH MATRICES
C
C
      DO 110 I=1,N
      NF=I+1
      NS=I+N-2
      DO 50 JJ=NF,NS
      IF (JJ-N) 30,30,20
   20 J=JJ-N
      GO TO 40
   30 J=JJ
C
   40 CALL INTE (X(I),Y(I),X(J),Y(J),X(J+1),Y(J+1),GELEM,HELEM)
C
      IF(J-N)42,43,43
   42 HH(I,J+1)=HH(I,J+1)+HELEM(2)
C
C
      GG(I,J+1)=GG(I,J+1)+GELEM(2)
      GO TO 44
   43 HH(I,1)=HH(I,1)+HELEM(2)
      GG(I,1)=GG(I,1)+GELEM(2)
   44 HH(I,J)=HH(I,J)+HELEM(1)
      GG(I,J)=GG(I,J)+GELEM(1)
   50 HH(I,I)=HH(I,I)-HELEM(1)-HELEM(2)
      NF=I+N-1
      NS=I+N
      DO 95 JJ=NF,NS
      IF (JJ-N) 70,70,60
   60 J=JJ-N
      GO TO 80
   70 J=JJ
C
   80 CALL INLO (X(J),Y(J),X(J+1),Y(J+1),GELEM)
C
      IF (JJ-NF) 82,82,83
   82 CH=GELEM(1)
      GELEM(1)=GELEM(2)
      GELEM(2)=CH
   83 IF (J-N) 85,90,90
   85 GG(I,J+1)=GG(I,J+1)+GELEM(2)
      GO TO 95
   90 GG(I,1)=GG(I,1)+GELEM(2)
   95 GG(I,J)=GG(I,J)+GELEM(1)
  110 CONTINUE
C
      RETURN
      END
C
C***********************************************************************
      SUBROUTINE GHPCAL
C
C     EVALUATES GGP AND HHP MATRICES BY
C     REORDERING GG AND HH MATRICES
C     ACC. TO THE TYPE OF THE B.C. AT NODE J
C
C
      COMMON/BEM1/X(41),Y(41),CX(9),CY(9),SOL(9),BB(40),M,NC(5)
      COMMON/BEM2/GG(40,40),HH(40,40),KODE(40),FI(41),DFI(41)
      COMMON/BEM3/NDIM,N,LINT,NI,NO,KODEI,KODEP,NPOIN,NELEM
      COMMON/BEM4/EXISP(7),ETASP(7),WEIGP(7),XM(40),YM(40)
      COMMON/BEM5/PTERM(40),LNODS(3,50),COORD(2,40)
      COMMON/BEM6/WORK(40),IPVT(40),COND,NONL,LMICMI
      COMMON/BEM7/GGP(40,40),HHP(40,40),ALPHA(40),BETA(40)
C
C
      DO 250 J=1,N
      GO TO (10,20,30),KODE(J)
   10 DO 100 I=1,N
      GGP(I,J)=GG(I,J)
  100 HHP(I,J)=HH(I,J)
      GO TO 250
   20 DO 200 I=1,N
      GGP(I,J)=-HH(I,J)
  200 HHP(I,J)=-GG(I,J)
      GO TO 250
   30 DO 300 I=1,N
      GGP(I,J)=-HH(I,J)-GG(I,J)*ALPHA(J)
  300 HHP(I,J)=-GG(I,J)*ALPHA(J)
  250 CONTINUE
C
C
      RETURN
      END
C
C*********************************************************************
      SUBROUTINE BCAL (XSRCE,YSRCE,BSRCE)
C
C
      COMMON/BEM1/X(41),Y(41),CX(9),CY(9),SOL(9),BB(40),M,NC(5)
      COMMON/BEM2/GG(40,40),HH(40,40),KODE(40),FI(41),DFI(41)
      COMMON/BEM3/NDIM,N,LINT,NI,U0,KODEI,KODEP,NPOIN,NELEM
      COMMON/BEM4/EXISP(7),ETASP(7),WEIGP(7),XM(40),YM(40)
      COMMON/BEM5/PTERM(40),LNODS(3,50),COORD(2,40)
      COMMON/BEM6/WORK(40),IPVT(40),COND,NONL,LMICMI
      COMMON/BEM7/GGP(40,40),HHP(40,40),ALPHA(40),BETA(40)
      DIMENSION LNODE(3),XNODE(3),YNODE(3),PNODE(3),SHAPE(3)
C
C     THIS SUBROUTINE CALCULATES BB(I)
C     FOR THE SOURCE POINT (XSRCE,YSRCE)
C     BSRCE=BB(I)
C
C
C     LOOP OVER INTERNAL TRIANGULAR ELEMENTS
C
      BSRCE=0.
      DO 515 IELEM=1,NELEM
C
C
      DO 520 INODE=1,3
      LNODE(INODE)=LNODS(INODE,IELEM)
      XNODE(INODE)=COORD(1,LNODE(INODE))
      YNODE(INODE)=COORD(2,LNODE(INODE))
  520 PNODE(INODE)=PTERM(LNODE(INODE))
C
C     CALCULATE DETERMINANT OF JACOBIAN MATRIX
C
C
      DJACB=(XNODE(2)-XNODE(1))*(YNODE(3)-YNODE(1))-(YNODE(2)-
     .YNODE(1))*(XNODE(3)-XNODE(1))
C
C     LOOP OVER GAUSS INTEGRATION POINTS
C     QUINTIC INTEGRATION, NGAUS=7
C
C
      DO 525 IGAUS=1,7
C
C     CALCULATE SHAPE FUNCTIONS AT INTEGRATION POINT
C
C
      SHAPE(1)=1-EXISP(IGAUS)-ETASP(IGAUS)
      SHAPE(2)=EXISP(IGAUS)
      SHAPE(3)=ETASP(IGAUS)
C
C     CALCULATION AT INTEGRATION POINT
C
      PGAUS=0.
      XGAUS=0.
      YGAUS=0.
      DO 530 INODE=1,3
      PGAUS=PGAUS+SHAPE(INODE)*PNODE(INODE)
      XGAUS=XGAUS+SHAPE(INODE)*XNODE(INODE)
  530 YGAUS=YGAUS+SHAPE(INODE)*YNODE(INODE)
C
C     CALCULATE DISTANCE BETWEEN I AND INTEGRATION POINT
C
      RA=SQRT((XGAUS-XSRCE)**2+(YGAUS-YSRCE)**2)
C
C     CALCULATE BSRCE=BB(I)
C
```

```
      BSRCE=BSRCE+DJACB*WEIGP(IGAUS)* ALOG(1/RA)*PGAUS
C
C
  525 CONTINUE
  515 CONTINUE
  999 RETURN
      END
C
C*****************************************************************
      SUBROUTINE INTE (XP,YP,X1,Y1,X2,Y2,GELEM,HELEM)
C
C     THIS SUBROUTINE COMPUTES THE INTEGRALS ALONG A LINEAR ELEMENT
C     WHICH DOES NOT INCLUDE THE NODE UNDER CONSIDERATION
C     DIST=DISTANCE FROM THE POINT UNDER CONSIDERATION TO THE BOUND.
C     RA=DISTANCE FROM THE POINT UNDER CONSIDERATION TO THE
C         INTEGRATION POINTS IN THE BOUNDARY ELEMENTS
C
C
C
C
      DIMENSION GELEM(2),HELEM(2)
      DIMENSION XCO(4),YCO(4),GI(4),OME(4)
      DATA GI/0.86113631,-0.86113631, 0.33998104,-0.33998104/
      DATA OME/0.34785485, 0.34785485, 0.65214515, 0.65214515/
      AX=(X2-X1)/2
      BX=(X2+X1)/2
      AY=(Y2-Y1)/2
      BY=(Y2+Y1)/2
C
C
      IF (AX) 10,20,10
   10 TA=AY/AX
      DIST=  ABS((TA*XP-YP+Y1-TA*X1)/  SQRT(TA**2+1))
      GO TO 30
   20 DIST=  ABS(XP-X1)
   30 SIG=(X1-XP)*(Y2-YP)-(X2-XP)*(Y1-YP)
      IF (SIG) 31,32,32
   31 DIST=-DIST
C
   32 DO 90 I=1,2
      GELEM(I)=0.
   90 HELEM(I)=0.
      DO 40 I=1,4
      XCO(I)=AX*GI(I)+BX
      YCO(I)=AY*GI(I)+BY
      RA=  SQRT((XP-XCO(I))**2+(YP-YCO(I))**2)
      GZ=  ALOG(1/RA)*OME(I)*  SQRT(AX**2+AY**2)
      HZ=DIST*OME(I)*  SQRT(AX**2+AY**2)/RA**2
      GELEM(1)=GELEM(1)-(GI(I)-1)*GZ/2
      GELEM(2)=GELEM(2)+(GI(I)+1)*GZ/2
      HELEM(1)=HELEM(1)+(GI(I)-1)*HZ/2
   40 HELEM(2)=HELEM(2)-(GI(I)+1)*HZ/2
C
C
      RETURN
      END
C
C*****************************************************************
      SUBROUTINE INLO (X1,Y1,X2,Y2,GELEM)
C
C     THIS SUBROUTINE COMPUTES THE INTEGRALS ALONG A LINEAR ELEMENT
C     INCLUDING THE NODE UNDER CONSIDERATION
C
      DIMENSION GELEM(2)
      SEP=  SQRT ((X2-X1)**2+(Y2-Y1)**2)
      GELEM(1)=SEP*(1.5-ALOG(SEP))/2
      GELEM(2)=SEP*(0.5-ALOG(SEP))/2
      RETURN
      END
C
C*****************************************************************
      SUBROUTINE INTEF
C
C     FOR INTERNAL POINTS,
C         WRITES GELEM AND HELEM ONTO DISC FILE(10)
C
C
      COMMON/BEM1/X(41),Y(41),CX(9),CY(9),SOL(9),BB(40),M,NC(5)
      COMMON/BEM2/GG(40,40),HH(40,40),KODE(40),FI(41),DFI(41)
      COMMON/BEM3/NDIM,N,LINT,NI,NO,KODEI,KODEP,NPOIN,NELEM
      COMMON/BEM4/EXISP(7),ETASP(7),WEIGP(7),XM(40),YM(40)
      COMMON/BEM5/PTERM(40),LNODS(3,50),COORD(2,40)
      COMMON/BEM6/WORK(40),IPVT(40),COND,NONL,LMICMI
      COMMON/BEM7/GGP(40,40),HHP(40,40),ALPHA(40),BETA(40)
C
C
      DIMENSION GELEM(2),HELEM(2)
C
C     PREPARE DISC FOR WRITING
C
C
      REWIND 10
C
C     LOOP OVER THE INTERNAL POINTS
C
C
      DO 20 K=1,LINT
C
C     LOOP OVER THE BOUNDARY ELEMENTS
C
C
      DO 30 J=1,N
C
      CALL INTE (CX(K),CY(K),X(J),Y(J),X(J+1),Y(J+1),GELEM,HELEM)
C
C     WRITE ONTO DISC FILE(10)
C
      WRITE (10) GELEM,HELEM
   30 CONTINUE
   20 CONTINUE
C
      RETURN
      END
C
C*****************************************************************
      SUBROUTINE INTER
C
C     THIS SUBROUTINE COMPUTES THE POTENTIAL VALUE AT INTERNAL POIN
C
C
      COMMON/BEM1/X(41),Y(41),CX(9),CY(9),SOL(9),BB(40),M,NC(5)
      COMMON/BEM2/GG(40,40),HH(40,40),KODE(40),FI(41),DFI(41)
      COMMON/BEM3/NDIM,N,LINT,NI,NO,KODEI,KODEP,NPOIN,NELEM
      COMMON/BEM4/EXISP(7),ETASP(7),WEIGP(7),XM(40),YM(40)
      COMMON/BEM5/PTERM(40),LNODS(3,50),COORD(2,40)
      COMMON/BEM6/WORK(40),IPVT(40),COND,NONL,LMICMI
      COMMON/BEM7/GGP(40,40),HHP(40,40),ALPHA(40),BETA(40)
      DIMENSION GELEM(2),HELEM(2)
C
C     PREPARE DISC(10) FOR READING
C
      REWIND 10
C
C     LOOP OVER THE INTERNAL POINTS
C
      DO 40 K=1,LINT
      SOL(K)=0.
C
C     CHECK KODEP
C
      IF (KODEP.EQ.0) GO TO 998
C
      CALL BCAL (CX(K),CY(K),BSRCE)
      SOL(K)=BSRCE
C
C     LOOP OVER THE BOUNDARY ELEMENTS
C
C
C
  998 DO 30 J=1,N
C
C     READ DISC FILE(10)
C
      READ (10) GELEM,HELEM
C
C
      IF (J-N) 32,33,33
   32 SOL(K)=SOL(K)+DFI(J)*GELEM(1)+DFI(J+1)*GELEM(2)-
     .FI(J)*HELEM(1)-FI(J+1)*HELEM(2)
      GO TO 30
   33 SOL(K)=SOL(K)+DFI(J)*GELEM(1)+DFI(1)*GELEM(2)-
     .FI(J)*HELEM(1)-FI(1)*HELEM(2)
   30 CONTINUE
   40 SOL(K)=SOL(K)/(2*3.1415926)
C
```

```
      RETURN
      END
C
C*********************************************************************
      SUBROUTINE DECOMP (NDIM,N,A,COND,IPVT,WORK)
C
      DIMENSION A(NDIM,N),WORK(N)
      INTEGER IPVT(N)
C
C     DECOMPOSES A REAL MATRIX BY GAUSSIAN ELIMINATION
C     AND ESTIMATES THE CONDITION OF THE MATRIX
C
C     USE SOLVE TO COMPUTE SOLUTIONS TO LINEAR SYSTEMS
C     INPUT:
C          NDIM=DECLARED ROW DIMENSION OF THE ARRAY CONTAINING A
C          N=ORDER OF THE MATRIX
C          A=MATRIX TO BE TRIANGULARIZED
C     OUTPUT:
C          A CONTAINS AN UPPER TRIANGULAR MATRIX U AND A PERMUTED
C     VERSION OF A LOWER TRIANGULAR MATRIX I-L SO THAT
C     (PERMUTATION MATRIX)*A=L*U
C
C
C
C     COND=AN ESTIMATE OF THE CONDITION OF A
C     FOR THE LINEAR SYSTEM A*X=B ,CHANGES IN A AND B
C     MAY CAUSE CHANGES COND TIMES AS LARGE IN X.
C     IF COND+1.0.EQ.COND, A IS SINGULAR TO WORKING PRECISION
C     COND IS SET TO 1.0E+32 IF EXACT SINGULARITY IS DETECTED
C
C     IPVT=THE PIVOT VECTOR
C          IPVT(K)=THE INDEX OF THE K-TH PIVOT ROW
C          IPVT(N)=(-1)**(NO. OF INTERCHANGES)
C
C     WORK SPACE ,
C        THE VECTOR WORK MUST BE DECLARED AND INCLUDED IN THE CALL.
C        ITS INPUT CONTENTS ARE IGNORED.ITS OUTPUT CONTENTS ARE
C        USUALLY UNIMPORTANT
C
C     THE DETERMINANT OF A CAN BE OBTAINED ON OUTPUT BY
C          DET(A)=IPVT(N)*A(1,1)*A(2,2)*.....*A(N,N)
C
C
C
C
      IPVT(N)=1
      IF (N.EQ.1) GO TO 80
      NM1=N-1
C
C
C     COMPUTE 1-NORM OF A.
C
      ANORM=0.0
      DO 10 J=1,N
      T=0.0
      DO 5 I=1,N
      T=T+ ABS(A(I,J))
    5 CONTINUE
      IF (T.GT.ANORM) ANORM=T
   10 CONTINUE
C
C     GAUSSIAN ELIMINATION WITH PARTIAL PIVOTING
C
C
C
      DO 35 K=1,NM1
      KP1=K+1
C
C     FIND PIVOT
C
C
C
      M=K
      DO 15 I=KP1,N
      IF ( ABS(A(I,K)).GT. ABS(A(M,K))) M=I
   15 CONTINUE
      IPVT(K)=M
      IF (M.NE.K) IPVT(N)=-IPVT(N)
      T=A(M,K)
      A(M,K)=A(K,K)
      A(K,K)=T
C
C     SKIP STEP IF PIVOT IS ZERO
C
```

```
C
      IF (T.EQ.0.0) GO TO 35
C
C     COMPUTE MULTIPLIERS
C
C
      DO 20 I=KP1,N
      A(I,K)=-A(I,K)/T
   20 CONTINUE
C
C     INTERCHANGE AND ELIMINATE BY COLUMNS
C
      DO 30 J=KP1,N
      T=A(M,J)
      A(M,J)=A(K,J)
      A(K,J)=T
      IF (T.EQ.0.0) GO TO 30
      DO 25 I=KP1,N
      A(I,J)=A(I,J)+A(I,K)*T
   25 CONTINUE
   30 CONTINUE
   35 CONTINUE
C
C       COND=(1-NORM OF A)*(AN ESTIMATE OF 1-NORM OF A-INVERSE)
C     ESTIMATE OBTAINED BY ONE STEP OF INVERSE ITERATION FOR THE
C     SMALL SINGULAR VECTOR.THIS INVOLVES SOLVING TWO SYSTEMS
C     OF EQUATIONS, (A-TRANSPOSE)*Y=E AND A*Z=Y WHERE E IS
C     A VECTOR OF +1 OR -1 CHOSEN TO CAUSE GROWTH IN Y.
C     ESTIMATE=(1-NORM OF Z)/(1-NORM OF Y)
C
C     SOLVE (A-TRANSPOSE)*Y=E
C
C
      DO 50 K=1,N
      T=0.0
      IF (K.EQ.1) GO TO 45
      KM1=K-1
      DO 40 I=1,KM1
      T=T+A(I,K)*WORK(I)
   40 CONTINUE
   45 EK=1.0
      IF (T.LT.0.0) EK=-1.0
      IF(A(K,K).EQ.0.0) GO TO 90
      WORK(K)=-(EK+T)/A(K,K)
   50 CONTINUE
      DO 60 KB=1,NM1
      K=N-KB
      T=0.0
      KP1=K+1
      DO 55 I=KP1,N
      T=T+A(I,K)*WORK(K)
   55 CONTINUE
      WORK(K)=T
      M=IPVT(K)
      IF (M.EQ.K) GO TO 60
      T=WORK(M)
      WORK(M)=WORK(K)
      WORK(K)=T
   60 CONTINUE
C
C
      YNORM=0.0
      DO 65 I=1,N
      YNORM=YNORM+ ABS(WORK(I))
   65 CONTINUE
C
C     SOLVE A*Z=Y
C
C
      CALL SOLVE (NDIM,N,A,WORK,IPVT)
C
C
      ZNORM=0.0
      DO 70 I=1,N
      ZNORM=ZNORM+ ABS(WORK(I))
   70 CONTINUE
C
C     ESTIMATE CONDITION
C
C
      COND=ANORM*ZNORM/YNORM
      IF (COND.LT.1.0) COND=1.0
      RETURN
C
C     1-BY-1
C
C
   80 COND=1.0
      IF (A(1,1).NE.0.0) RETURN
C
C     EXACT SINGULARITY
C
   90 COND=1.0E+32
      RETURN
      END
C
C*******************************************************************************
      SUBROUTINE SOLVE (NDIM,N,A,B,IPVT)
C
      DIMENSION A(NDIM,N),B(N)
      INTEGER IPVT(N)
C
C     SOLUTION OF LINEAR SYSTEM, A*X=B
C       DO NOT USE IF DECOMP HAS DETECTED SINGULARITY
C     INPUT,
C       NDIM=DECLARED ROW DIMENSION OF ARRAY CONTAINING A
C       N=ORDER OF MATRIX
C       A=TRIANGULARIZED MATRIX OBTAINED FROM DECOMP
C       B=RIGHT HAND SIDE VECTOR
C       IPVT=PIVOT VECTOR OBTAINED FROM DECOMP
C     OUTPUT,
C       B=SOLUTION VECTOR, X
C
C
C
C
C     FORWARD ELIMINATION
C
C
C
C
      IF (N.EQ.1) GO TO 50
      NM1=N-1
      DO 20 K=1,NM1
      KP1=K+1
      M=IPVT(K)
      T=B(M)
      B(M)=B(K)
      B(K)=T
      DO 10 I=KP1,N
      B(I)=B(I)+A(I,K)*T
   10 CONTINUE
   20 CONTINUE
C
C     BACK SUBSTITUTION
C
      DO 40 KB=1,NM1
      KM1=N-KB
      K=KM1+1
      B(K)=B(K)/A(K,K)
      T=-B(K)
      DO 30 I=1,KM1
      B(I)=B(I)+A(I,K)*T
   30 CONTINUE
   40 CONTINUE
   50 B(1)=B(1)/A(1,1)
      RETURN
      END
C
C
C*******************************************************************************
      SUBROUTINE OUTPUT
      COMMON/BEM1/X(41),Y(41),CX(9),CY(9),SOL(9),BB(40),M,NC(5)
      COMMON/BEM2/GG(40,40),HH(40,40),KODE(40),FI(41),DFI(41)
      COMMON/BEM3/NDIM,N,LINT,NI,NO,KODEI,KODEP,NPOIN,NELEM
      COMMON/BEM4/EXISP(7),ETASP(7),WEIGP(7),XM(40),YM(40)
      COMMON/BEM5/PTERM(40),LNODS(3,50),COORD(2,40)
      COMMON/BEM6/WORK(40),IPVT(40),COND,NONL,LMICMI
      COMMON/BEM7/GGP(40,40),HHP(40,40),ALPHA(40),BETA(40)
      COMMON/BEM8/EFI(41)
      WRITE (NO,100)
  100 FORMAT(' ',120('*')//1X,'RESULTS'//2X,'BOUNDARY NODES'//2X
     .,'NODE',10X,'X',17X,'Y',12X,'APP. POT.',6X,
     .1X,'APP. POT. DERIV.')
```

```
C
C
      DO 201 I=1,N
C
      IF(M-1)31,31,10
   31 IF(LMICMI-1)33,10,33
C
   10 WRITE(NO,200) I,XM(I),YM(I),FI(I),DFI(I)
      GO TO 201
   33 WRITE(NO,200) I,X(I),Y(I),FI(I),DFI(I)
  201 CONTINUE
  200 FORMAT(1X,I3,4X,E14.7,3(4X,E14.7))
C
C     CHECK KODEI
C
      IF (KODEI.EQ.0) GO TO 777
C
      WRITE (NO,300)
  300 FORMAT(//,2X,'INTERNAL POINTS',//1X,'NODE',8X,'X',20X,
     .'Y',11X,'APP. POT.')
      DO 20 K=1,LINT
   20 WRITE(NO,400)K,CX(K),CY(K),SOL(K)
  400 FORMAT(I4,2X,E14.7,3(5X,E14.7))
  777 WRITE (NO,500)
  500 FORMAT (' ',120('*'))
      RETURN
      END
C***************************************************************
      SUBROUTINE GUCALC
      COMMON/BEM1/X(41),Y(41),CX(9),CY(9),SOL(9),BB(40),M,NC(5)
      COMMON/BEM2/GG(40,40),HH(40,40),KODE(40),FI(41),DFI(41)
      COMMON/BEM3/NDIM,N,LINT,NI,NO,KODEI,KODEP,NPOIN,NELEM
      COMMON/BEM4/EXISP(7),ETASP(7),WEIGP(7),XM(40),YM(40)
      COMMON/BEM5/PTERM(40),LNODS(3,50),COORD(2,40)
      COMMON/BEM6/WORK(40),IPVT(40),COND,NONL,LMICMI
      COMMON/BEM7/GGP(40,40),HHP(40,40),ALPHA(40),BETA(40)
C
C     COMPUTE GG AND HH MATRICES FOR CONSTANT ELEMENTS
C
C
      X(N+1)=X(1)
      Y(N+1)=Y(1)
      DO 11 I=1,N
      XM(I)=(X(I)+X(I+1))/2.
   11 YM(I)=(Y(I)+Y(I+1))/2.
      IF(M-1)15,15,12
   12 XM(NC(1))=(X(NC(1))+X(1))/2.
      YM(NC(1))=(Y(NC(1))+Y(1))/2.
      DO 13 K=2,M
      XM(NC(K))=(X(NC(K))+X(NC(K-1)+1))/2.
   13 YM(NC(K))=(Y(NC(K))+Y(NC(K-1)+1))/2.
   15 DO 110 I=1,N
      DO 110 J=1,N
C
C
C
      IF(M-1)16,16,17
   17 IF(J-NC(1))19,18,19
   18 KKK=1
      GO TO 23
   19 DO 22 K=2,M
      IF(J-NC(K))22,21,22
   21 KKK=NC(K-1)+1
      GO TO 23
   22 CONTINUE
   16 KKK=J+1
C
C
   23 IF(I-J)20,25,20
   20 CALL INTEC (XM(I),YM(I),X(J),Y(J),X(KKK),Y(KKK),HH(I,J),GG(
      GO TO 110
   25 CALL INLOC (X(J),Y(J),X(KKK),Y(KKK),GG(I,J))
      HH(I,J)=3.1415926
  110 CONTINUE
C
C
      RETURN
      END
C
C***************************************************************
      SUBROUTINE INTEC (XP,YP,X1,Y1,X2,Y2,HH,GG)
C
```

```
C     THIS SUBROUTINE COMPUTES THE VALUES OF THE HH AND GG MATRIX
C     OFF DIAGONAL ELEMENTS BY MEANS OF NUMERICAL INTEGRATION
C     ALONG THE CONSTANT ELEMENTS.
C
      DIMENSION XCO(4),YCO(4),GI(4),OME(4)
      DATA GI/0.86113631,-0.86113631, 0.33998104,-0.33998104/
      DATA OME/0.34785485, 0.34785485, 0.65214515, 0.65214541/
      AX=(X2-X1)/2.
      BX=(X2+X1)/2.
      AY=(Y2-Y1)/2.
      BY=(Y2+Y1)/2.
C
C
      IF (AX) 10,20,10
   10 TA=AY/AX
      DIST= ABS((TA*XP-YP+Y1-TA*X1)/ SQRT(TA**2+1))
      GO TO 30
   20 DIST= ABS(XP-X1)
   30 SIG=(X1-XP)*(Y2-YP)-(X2-XP)*(Y1-YP)
      IF (SIG) 31,32,32
   31 DIST=-DIST
C
   32 GG=0.
      HH=0.
      DO 40 I=1,4
      XCO(I)=AX*GI(I)+BX
      YCO(I)=AY*GI(I)+BY
      RA= SQRT((XP-XCO(I))**2+(YP-YCO(I))**2)
      GZ= ALOG(1/RA)*OME(I)* SQRT(AX**2+AY**2)
      HZ= DIST*OME(I)* SQRT(AX**2+AY**2)/RA**2
C
      GG=GG+GZ
   40 HH=HH-HZ
      RETURN
      END
C*********************************************************************
      SUBROUTINE INLOC (X1,Y1,X2,Y2,GG)
C
C     THIS SUBROUTINE COMPUTES THE VALUES OF THE DIAGONAL
C     ELEMENTS OF THE GG MATRIX FOR CONSTANT BOUNDARY ELEMENTS.
C
      AX=(X2-X1)/2.
      AY=(Y2-Y1)/2.
      SEP=SQRT(AX**2+AY**2)
      GG=2.*SEP*(ALOG(1./SEP)+1.)
C
      RETURN
      END
C
C
C*********************************************************************
      SUBROUTINE INTEFC
C
C     THIS SUBROUTINE IS FOR THE CONSTANT BOUNDARY ELEMENTS
C     FOR INTERNAL POINTS,
C       WRITES AH AND BG ONTO DISC FILE(10)
C
      COMMON/BEM1/X(41),Y(41),CX(9),CY(9),SOL(9),BB(40),M,NC(5)
      COMMON/BEM2/GG(40,40),HH(40,40),KODE(40),FI(41),DFI(41)
      COMMON/BEM3/NDIM,N,LINT,NI,NO,KODEI,KODEP,NPOIN,NELEM
      COMMON/BEM4/EXISP(7),ETASP(7),WEIGP(7),XM(40),YM(40)
      COMMON/BEM5/PTERM(40),LNODS(3,50),COORD(2,40)
      COMMON/BEM6/WORK(40),IPVT(40),COND,NONL,LMICMI
      COMMON/BEM7/GGP(40,40),HHP(40,40),ALPHA(40),BETA(40)
C
C     PREPARE DISC FOR WRITING
C
C
C
      REWIND 10
C
C     LOOP OVER THE INTERNAL POINTS
C
C
      DO 20 K=1,LINT
C
C     LOOP OVER THE BOUNDARY ELEMENTS
C
C
      DO 30 J=1,N
C
C
```

```
      IF(M-1)16,16,17
   17 IF(J-NC(1))19,18,19
   18 KKK=1
      GO TO 23
   19 DO 22 KL=2,M
      IF(J-NC(KL))22,21,22
   21 KKK=NC(KL-1)+1
      GO TO 23
   22 CONTINUE
   16 KKK=J+1
C
C
   23 CALL INTEC (CX(K),CY(K),X(J),Y(J),X(KKK),Y(KKK),AH,BG)
C
C     WRITE ONTO DISC FILE(10)
C
      WRITE(10) AH,BG
   30 CONTINUE
   20 CONTINUE
C
C
      RETURN
      END
C
C*****************************************************************************
C
      SUBROUTINE INTERC
C
C     THIS SUBROUTINE COMPUTES THE POTENTIAL VALUE AT INTERNAL
C     ELEMENTS FOR CONSTANT BOUNDARY ELEMENTS
C
      COMMON/BEM1/X(41),Y(41),CX(9),CY(9),SOL(9),BB(40),M,NC(5)
      COMMON/BEM2/GG(40,40),HH(40,40),KODE(40),FI(41),DFI(41)
      COMMON/BEM3/NDIM,N,LINT,NI,NO,KODET,KODEP,NPOIN,NELEM
      COMMON/BEM4/EXISP(7),ETASP(7),WEIGP(7),XM(40),YM(40)
      COMMON/BEM5/PTERM(40),LNODS(3,50),COORD(2,40)
      COMMON/BEM6/WORK(40),IPVT(40),COND,NONL,LMICMI
      COMMON/BEM7/GGP(40,40),HHP(40,40),ALPHA(40),BETA(40)
C
C     PREPARE DISC(10) FOR READING
C
      REWIND 10
C
C     LOOP OVER THE INTERNAL POINTS
C
      DO 40 K=1,LINT
      SOL(K)=0.
C
C     CHECK KODEP
C
      IF (KODEP.EQ.0) GO TO 998
C
C
      CALL BCAL (CX(K),CY(K),BSRCE)
      SOL(K)=BSRCE
C
C     LOOP OVER THE BOUNDARY ELEMENTS
C
  998 DO 30 J=1,N
C
C     READ DISC FILE(10)
C
      READ(10) AH,BG
C
   30 SOL(K)=SOL(K)+DFI(J)*BG-FI(J)*AH
   40 SOL(K)=SOL(K)/(2.*3.1415926)
C
C
      RETURN
      END
   DATA AND RESULTS FOR PROBLEM (1c)
   M=1      LMICMI=1      NONL=0
   NC(K)?    16
   N= 16  KODEI=1  KODEP=0

 COORDINATES OF THE EXTREME POINTS OF THE  BOUNDARY ELEMENTS
   POINT           X                    Y
     1         .0000000             .0000000
     2         .2500000+000         .0000000
     3         .5000000+000         .0000000
     4         .7500000+000         .0000000
     5         .1000000+001         .0000000
     6         .1000000+001         .2500000+000
     7         .1000000+001         .5000000+000
     8         .1000000+001         .7500000+000
     9         .1000000+001         .1000000+001
    10         .7500000+000         .1000000+001
    11         .5000000+000         .1000000+001
    12         .2500000+000         .1000000+001
    13         .0000000             .1000000+001
    14         .0000000             .7500000+000
    15         .0000000             .5000000+000
    16         .0000000             .2500000+000

 BOUNDARY CONDITIONS
    NODE(I)    KODE(I)       ALPHA(I)            BETA(I)
     1           2         .0000000            .0000000
     2           2         .0000000            .0000000
     3           2         .0000000            .0000000
     4           2         .0000000            .0000000
     5           1         .0000000            .1000000+002
     6           1         .0000000            .1000000+002
     7           1         .0000000            .1000000+002
     8           1         .0000000            .1000000+002
     9           3         .5000000+002        .2000000+002
    10           3         .5000000+002        .2000000+002
    11           3         .5000000+002        .2000000+002
    12           3         .5000000+002        .2000000+002
    13           1         .0000000            .1000000+003
    14           1         .0000000            .1000000+003
    15           1         .0000000            .1000000+003
    16           1         .0000000            .1000000+003
 CONDITION NO=      .30337+003
*****************************************************************************
RESULTS
 BOUNDARY NODES
NODE          X                  Y                  APP. POT.(u)      APP. POT. DERIV.(
 1       .1250000+000       .0000000            .8794546+002       .0000000
 2       .3750000+000       .0000000            .6307473+002       .0000000
 3       .6250000+000       .0000000            .4038484+002       .0000000
 4       .8750000+000       .0000000            .1945775+002       .0000000
 5       .1000000+001       .1250000+000        .1000000+002      -.8322287+002
 6       .1000000+001       .3750000+000        .1000000+002      -.7182383+002
 7       .1000000+001       .6250000+000        .1000000+002      -.5982903+002
 8       .1000000+001       .8750000+000        .1000000+002      -.6924791+002
 9       .8750000+000       .1000000+001        .1925288+002       .3735592+002
10       .6250000+000       .1000000+001        .2076315+002      -.3815775+002
11       .3750000+000       .1000000+001        .2203964+002      -.1019820+003
12       .1250000+000       .1000000+001        .2806613+002      -.4033065+003
13       .0000000           .8750000+000        .1000000+003       .4340790+003
14       .0000000           .6250000+000        .1000000+003       .1371114+003
15       .0000000           .3750000+000        .1000000+003       .1115742+003
16       .0000000           .1250000+000        .1000000+003       .1083507+003

 INTERNAL POINTS
NODE          X                  Y                  APP. POT.
 1      .2500000+000       .2500000+000        .7403933+002
 2      .7500000+000       .2500000+000        .2929521+002
 3      .7500000+000       .7500000+000        .2335871+002
 4      .2500000+000       .7500000+000        .5794830+002
 5      .5000000+000       .5000000+000        .4606160+002
```

```
DATA AND RESULTS FOR PROBLEM (3)

M=1       LMICMI=2      NONL=0

NC(K):    23

N= 23   KODEI=1   KODEP=1

COORDINATES OF THE EXTREME POINTS OF THE  BOUNDARY ELEMENTS

 POINT              X                    Y
   1          .1000000-002         .0000000
   2          .2000000+001         .0000000
   3          .4000000+001         .0000000
   4          .6000000+001         .0000000
   5          .8000000+001         .0000000
   6          .9999000+001         .0000000
   7          .1000000+002         .1000000-002
   8          .1000000+002         .2000000+001
   9          .1000000+002         .3999000+001
  10          .9999000+001         .4000000+001
  11          .8000000+001         .4000000+001
  12          .6000000+001         .4000000+001
  13          .4000000+001         .4000000+001
  14          .4000000+001         .6000000+001
  15          .4000000+001         .7999000+001
  16          .3999000+001         .8000000+001
  17          .2000000+001         .8000000+001
  18          .1000000-002         .8000000+001
  19          .0000000             .7999000+001
  20          .0000000             .6000000+001
  21          .0000000             .4000000+001
  22          .0000000             .2000000+001
  23          .0000000             .1000000-002

BOUNDARY CONDITIONS

  NODE(I)    KODE(I)       ALPHA(I)             BETA(I)
    1          1         .0000000            .3000000+003
    2          1         .0000000            .3000000+003
    3          1         .0000000            .3000000+003
    4          1         .0000000            .3000000+003
    5          1         .0000000            .3000000+003
    6          1         .0000000            .3000000+003
    7          2         .0000000            .0000000
    8          2         .0000000            .0000000
    9          2         .0000000            .0000000
   10          3         .2000000+002        .5000000+003
   11          3         .2000000+002        .5000000+003
   12          3         .2000000+002        .5000000+003
   13          3         .2000000+002        .5000000+003
   14          3         .2000000+002        .5000000+003
   15          3         .2000000+002        .5000000+003
   16          2         .0000000            .0000000

   17          2         .0000000            .0000000
   18          2         .0000000            .0000000
   19          1         .0000000            .3000000+003
   20          1         .0000000            .3000000+003
   21          1         .0000000            .3000000+003
   22          1         .0000000            .3000000+003
   23          1         .0000000            .3000000+003
   INTERNAL SOURCE TERM DATA
    NPOIN = 24      NELEM = 28

     EXISP            ETASP            WEIGP
   .33333333        .33333333        .11250000
   .10128651        .10128651        .06296959
   .79742699        .10128651        .06296959
   .10128651        .79742699        .06296959
   .47014206        .47014206        .06619708
   .05971587        .47014206        .06619708
   .47014206        .05971587        .06619708
```

```
EL    NODES
 1     1     8     7
 2     1     2     8
 3     2     9     8
 4     2     3     9
 5     3    10     9
 6     3     4    10
 7     4    11    10
 8     4     5    11
 9     5    12    11
10     5     6    12
11     7    14    13
12     7     8    14
13     8    15    14
14     8     9    15
15     9    16    15
16     9    10    16
17    10    17    16
18    10    11    17
19    11    18    17
20    11    12    18
21    13    20    19
22    13    14    20
23    14    21    20
24    14    15    21
25    19    23    22
26    19    20    23
27    20    24    23
28    20    21    24
         NODE        X-COORD           Y-COORD
           1           .00000           .00000
           2          2.00000           .00000
           3          4.00000           .00000
           4          6.00000           .00000
           5          8.00000           .00000
           6         10.00000           .00000
           7           .00000          2.00000
           8          2.00000          2.00000
           9          4.00000          2.00000
          10          6.00000          2.00000
          11          8.00000          2.00000
          12         10.00000          2.00000
          13           .00000          4.00000
          14          2.00000          4.00000
          15          4.00000          4.00000
          16          6.00000          4.00000
          17          8.00000          4.00000
          18         10.00000          4.00000
          19           .00000          6.00000
          20          2.00000          6.00000
          21          4.00000          6.00000
          22           .00000          8.00000
          23          2.00000          8.00000
          24          4.00000          8.00000
         NODE          PTERM
           1       1000.00000
           2       1000.00000
           3       1000.00000
           4       1000.00000

           5       1000.00000
           6       1000.00000
           7       1000.00000
           8       1000.00000
           9       1000.00000
          10       1000.00000
          11       1000.00000
          12       1000.00000
          13       1000.00000
          14       1000.00000
          15       1000.00000
          16       1000.00000
          17       1000.00000
```

```
      18      1000.00000
      19      1000.00000
      20      1000.00000
      21      1000.00000
      22      1000.00000
      23      1000.00000
      24      1000.00000
 CONDITION NO=       .10223+005
**********************************************************************************
RESULTS
 BOUNDARY NODES
 NODE           X                       Y                       APP. POT.            APP. POT. DERIV.
  1        .1000000-002            .0000000                 .3000000+003           -.4333732+003
  2        .2000000+001            .0000000                 .3000000+003           -.2045554+004
  3        .4000000+001            .0000000                 .3000000+003           -.2162779+004
  4        .6000000+001            .0000000                 .3000000+003           -.2109500+004
  5        .8000000+001            .0000000                 .3000000+003           -.2013638+004
  6        .9999000+001            .0000000                 .3000000+003           -.2228507+004
  7        .1000000+002            .1000000-002             .3912278+003            .0000000
  8        .1000000+002            .2000000+001             .2494661+004            .0000000
  9        .1000000+002            .3999000+001             .6859674+003            .0000000
 10        .9999000+001            .4000000+001             .6021926+003           -.2043852+004
 11        .8000000+001            .4000000+001             .5951219+003           -.1902438+004
 12        .6000000+001            .4000000+001             .5837759+003           -.1675517+004
 13        .4000000+001            .4000000+001             .6784403+003           -.3568805+004
 14        .4000000+001            .6000000+001             .5803632+003           -.1607264+004
 15        .4000000+001            .7999000+001             .6055205+003           -.2110411+004
 16        .3999000+001            .8000000+001             .6922326+003            .0000000
 17        .2000000+001            .8000000+001             .2507613+004            .0000000
 18        .1000000-002            .8000000+001             .3915403+003            .0000000
 19        .0000000                .7999000+001             .3000000+003           -.2236263+004
 20        .0000000                .6000000+001             .3000000+003           -.2034954+004
 21        .0000000                .4000000+001             .3000000+003           -.2168542+004
 22        .0000000                .2000000+001             .3000000+003           -.2044660+004
 23        .0000000                .1000000-002             .3000000+003           -.4333830+003

 INTERNAL POINTS

NODE          X                       Y                       APP. POT.
  1       .2000000+001            .6000000+001             .2448166+004
  2       .2000000+001            .4000000+001             .2640574+004
  3       .2000000+001            .2000000+001             .2483208+004
  4       .4000000+001            .2000000+001             .2645709+004
  5       .6000000+001            .2000000+001             .2477092+004
  6       .8000000+001            .2000000+001             .2419300+004
```

DATA AND RESULTS FOR PROBLEM (4a)

```
M=1        LMICMI=2       NONL=1

NC(K):    20

N= 20   KODEI=1   KODEP=0

COORDINATES OF THE EXTREME POINTS OF THE  BOUNDARY ELEMENTS

  POINT             X                    Y
     1       .1000000-002          .0000000
     2       .2500000+000          .0000000
     3       .5000000+000          .0000000
     4       .7500000+000          .0000000
     5       .9990000+000          .0000000
     6       .1000000+001          .1000000-002
     7       .1000000+001          .2500000+000
     8       .1000000+001          .5000000+000
     9       .1000000+001          .7500000+000
    10       .1000000+001          .9990000+000
    11       .9990000+000          .1000000+001
    12       .7500000+000          .1000000+001
    13       .5000000+000          .1000000+001
    14       .2500000+000          .1000000+001
    15       .1000000-002          .1000000+001
    16       .0000000              .9990000+000
    17       .0000000              .7500000+000
    18       .0000000              .5000000+000
    19       .0000000              .2500000+000
    20       .0000000              .1000000-002

BOUNDARY CONDITIONS

   NODE(I)    KODE(I)      ALPHA(I)              BETA(I)
     1           2         .0000000             .0000000
     2           2         .0000000             .0000000
     3           2         .0000000             .0000000
     4           2         .0000000             .0000000
     5           2         .0000000             .0000000
     6           3         .0000000             .0000000
     7           3         .0000000             .0000000
     8           3         .0000000             .0000000
     9           3         .0000000             .0000000
    10           3         .0000000             .0000000
    11           2         .0000000             .0000000
    12           2         .0000000             .0000000
    13           2         .0000000             .0000000
    14           2         .0000000             .0000000
    15           2         .0000000             .0000000
    16           2         .0000000             .1000000+004
    17           2         .0000000             .1000000+004
    18           2         .0000000             .1000000+004
    19           2         .0000000             .1000000+004
    20           2         .0000000             .1000000+004

   EPSMAX= .100      TAMB=350.0      NST=  6      NLA= 10      THC= 1.00
CONDITION NO=       .30235+002
CONDITION NO=       .51157+002
CONDITION NO=       .44460+002
CONDITION NO=       .43857+002

                 ITER =  4                    EPS =    .2099609-001
```

```
RESULTS
 BOUNDARY NODES

 NODE          X                         Y                        APP. POT.           APP. POT. DERIV.
  1        .1000000-002            .0000000                   .1418607+004            .0000000
  2        .2500000+000            .0000000                   .1174095+004            .0000000
  3        .5000000+000            .0000000                   .9243633+003            .0000000
  4        .7500000+000            .0000000                   .6746285+003            .0000000
  5        .9990000+000            .0000000                   .4301250+003            .0000000
  6        .1000000+001            .1000000-002               .4250370+003           -.9995961+003
  7        .1000000+001            .2500000+000               .4251029+003           -.1000745+004
  8        .1000000+001            .5000000+000               .4249988+003           -.9989309+003
  9        .1000000+001            .7500000+000               .4251029+003           -.1000745+004
 10        .1000000+001            .9990000+000               .4250370+003           -.9995959+003
 11        .9990000+000            .1000000+001               .4301251+003            .0000000
 12        .7500000+000            .1000000+001               .6746285+003            .0000000
 13        .5000000+000            .1000000+001               .9243633+003            .0000000
 14        .2500000+000            .1000000+001               .1174095+004            .0000000
 15        .1000000-002            .1000000+001               .1418607+004            .0000000
 16        .0000000                .9990000+000               .1423698+004            .1000000+004
 17        .0000000                .7500000+000               .1423577+004            .1000000+004
 18        .0000000                .5000000+000               .1423827+004            .1000000+004
 19        .0000000                .2500000+000               .1423577+004            .1000000+004
 20        .0000000                .1000000-002               .1423698+004            .1000000+004

 INTERNAL POINTS

NODE          X                         Y                        APP. POT.
  1      .2500000+000            .2500000+000               .1173917+004
  2      .7500000+000            .2500000+000               .6748170+003
  3      .7500000+000            .7500000+000               .6748170+003
  4      .2500000+000            .7500000+000               .1173917+004
  5      .5000000+000            .5000000+000               .9243668+003
```

APPENDIX D

FINITE DIFFERENCE METHOD

In the finite difference approach the partial differential equation of the heat conduction is approximated by a set of algebraic equations for temperature at a number of grid points over the region. Therefore, the first in the analysis is the finite difference representation or the transformation into a set of algebraic equations of the differential equation of heat conduction [19, p.128]. The second in the analysis is the solution of a system of simultaneous equations with the temperatures as the unknowns. The Gauss-Seidel iterative process is one method frequently used [17, p. 486]. To start, a temperature is assumed everywhere at the plate. The process of iteration through all grid point is repeated until further iterations would produce, it is hoped, very little change in the computed temperatures. The following programs stop if the sum EPS, over all grid points, of the absolute values of the deviations of the temperatures from their previously computed values, falls below a small quantity EPSMAX. Computations will also be discontinued if the number of complete iterations, ITER, exceeds an upper limit, ITMAX.

```
C
C
C      FINITE DIFFERENCE METHOD
C      STEADY-STATE HEAT CONDUCTION IN A PLATE
C      GAUSS-SEIDEL TYPE OF SOLUTION
C      FOR PROBLEM (1b)
C
      DIMENSION T(30,30)
C
C     N:NO. OF GRID POINTS IN X- AND Y-DIRECTIONS
C     ITMAX=MAXIMUM NO. OF ITERATIONS
C     EPSMAX=MAXIMUM DEVIATION OF THE TEMPERATURES
C
      N=9
      ITMAX=1000
      EPSMAX=0.001
C
C      INITIAL GUESSES FOR TEMP.
C
      DO 1 I=1,N
      DO 1 J=1,N
    1 T(I,J)=0.0
C
C
C      CALCULATE SUCCESSIVELY BETTER APPROXIMATIONS FOR TEMP.
C      AT ALL POINTS,ITERATING UNTIL SATISFACTORY
C      CONVERGENCE IS ACHIEVED
C
C
      ITER=0
    3 ITER=ITER+1
      EPS=0.
C
C     G=(VOL. HEAT GENERATION)*(MESH SIZE**2)/THERMAL COND.
C     H=2.*CONVECTIVE H.T.C.*MESH SIZE/THERMAL COND.
C     HA=H*AMBIENT TEMP.
C
      G=0.*(.125**2)/1.
      H=2.*20.*.125/1.
      HA=H*1.
C
C          ALONG THE X-AXIS
C
      DO 5 I=2,N-1
      HOLDT=T(I,1)
      T(I,1)=(T(I+1,1)+T(I-1,1)+2.0*T(I,2)+G)/4.
    5 EPS=EPS+ABS(T(I,1)-HOLDT)
C
C        ALONG THE Y-AXIS
C
      DO 6 J=2,N-1
      HOLDT=T(1,J)
      T(1,J)=(T(1,J+1)+T(1,J-1)+2.0*T(2,J)+G)/4.
    6 EPS=EPS+ABS(T(1,J)-HOLDT)
C
C        ALONG THE Y=1 LINE
C
      DO 7 I=2,N-1
      HOLDT=T(I,N)
      T(I,N)=(2.0*T(I,N-1)+T(I-1,N)+T(I+1,N)+G+HA)/(H+4.)
    7 EPS=EPS+ABS(T(I,N)-HOLDT)
C          FOR CORNER POINTS
C
      HOLDT=T(1,1)
      T(1,1)=(2.*T(2,1)+2.*T(1,2)+G)/4.
      EPS=EPS+ABS(T(1,1)-HOLDT)
      HOLDT=T(1,N)
      T(1,N)=(2.*T(2,N)+2.*T(1,N-1)+G+HA)/(H+4.)
      EPS=EPS+ABS(T(1,N)-HOLDT)
C
C          FOR INTERIOR POINTS
C
      DO 8 I=2,N-1
      DO 8 J=2,N-1
      HOLDT=T(I,J)
      T(I,J)=(T(I,J+1)+T(I,J-1)+T(I+1,J)+T(I-1,J)+G)/4.
    8 EPS=EPS+ABS(T(I,J)-HOLDT)
C
C      STOP ITERATIONS IF COMPUTED VALUES SHOW LITTLE FURTHER
C      CHANGE,OR IF THE NO. OF ITERATIONS IS TOO LARGE
C
      IF(ITER.GE.ITMAX)GO TO 14
      IF(EPS-EPSMAX)14,3,3
C
C      PRINT        NO. OF ITERATIONS(ITER),THE LAST DEVIATION(EPS)
C      AND THE TEMPERATURES
C
   14 WRITE(6,201)ITER,EPS
      DO 11 J=N,1,-1
   11 WRITE(6,202) (T(I,J),I=1,N)
  201 FORMAT(1H1,///,30X,'ITER =',I5,///,30X,'EPS =',E14.9)
  202 FORMAT(//,2X,9E13.7)
      STOP
```

```
RESULTS:

                        ITER =  188

                        EPS =.991735607-003

.9574618+000 .9563752+000 .9529376+000 .9465047+000 .9355642+000 .9162731+000 .8753595+000 .7422733+000 .0000000

.8522030+000 .8484980+000 .8367870+000 .8150270+000 .7786557+000 .7177713+000 .6098476+000 .4025513+000 .0000000

.7543712+000 .7486338+000 .7306913+000 .6981658+000 .6462642+000 .5663114+000 .4437095+000 .2580844+000 .0000000

.6680469+000 .6609936+000 .6391956+000 .6006954+000 .5419356+000 .4575090+000 .3405995+000 .1860780+000 .0000000

.5958773+000 .5881268+000 .5644280+000 .5235068+000 .4632914+000 .3812023+000 .2751092+000 .1456304+000 .0000000

.5392708+000 .5312463+000 .5069168+000 .4656418+000 .4065446+000 .3289167+000 .2330146+000 .1213375+000 .0000000

.4987879+000 .4907166+000 .4663925+000 .4256348+000 .3683569+000 .2949261+000 .2067075+000 .1067088+000 .0000000

.4745323+000 .4664925+000 .4423495+000 .4021888+000 .3463549+000 .2757473+000 .1921949+000 .9879492-001 .0000000

.4664809+000 .4584294+000 .4343763+000 .3944609+000 .3391627+000 .2695397+000 .1875457+000 .9628111-001 .0000000
```

```
      FINITE DIFFERENCE METHOD FOR PROBLEM (3)
      DIMENSION T(90,90)
      N=11
      M=9
      ITMAX=1000
      EPSMAX=0.001
C
      DO 1 I=1,N
      DO 1 J=1,M
    1 T(I,J)=300.
C
      ITER=0
    3 ITER=ITER+1
      EPS=0.
      G= 2000.*1.*1./2.
      H=2.*40.*1./2.
      HA=500.*H
C
      DO 6  J=2,4
      HOLDT=T(N,J)
      T(N,J)=(T(N,J+1)+T(N,J-1)+2.*T(N-1,J)+G)/4.
    6 EPS=EPS+ABS(T(N,J)-HOLDT)
C
      DO 16 I=2,4
      HOLDT=T(I,M)
      T(I,M)=(T(I+1,M)+T(I-1,M)+2.*T(I,M-1)+G)/4.
   16 EPS=EPS+ABS(T(I,M)-HOLDT)
C
      DO 7 I=5,N-1
      HOLDT=T(I,5)
      T(I,5)=(T(I-1,5)+T(I+1,5)+2.*T(I,4)+HA+G)/(4.+H)
    7 EPS=EPS+ABS(T(I,5)-HOLDT)
C
      DO 17 J=6,M-1
      HOLDT=T(5,J)
      T(5,J)=(T(5,J-1)+T(5,J+1)+2.*T(4,J)+HA+G)/(4.+H)
   17 EPS=EPS+ABS(T(5,J)-HOLDT)
C
      HOLDT=T(5,M)
      T(5,M)=(2.*T(4,M)+2.*T(5,M-1)+G+HA)/(4.+H)
      EPS=EPS+ABS(T(5,M)-HOLDT)
C
      HOLDT=T(N,5)
      T(N,5)=(2.*T(N-1,5)+2.*T(N,4)+G+HA)/(4.+H)
      EPS=EPS+ABS(T(N,5)-HOLDT)
C
      DO 8 I=2,N-1
      DO 8 J=2,4
      HOLDT=T(I,J)
      T(I,J)=(T(I,J+1)+T(I,J-1)+T(I+1,J)+T(I-1,J)+G)/4.
    8 EPS=EPS+ABS(T(I,J)-HOLDT)
C
      DO 18 I=2,4
      DO 18 J=5,M-1
      HOLDT=T(I,J)
      T(I,J)=(T(I,J+1)+T(I,J-1)+T(I+1,J)+T(I-1,J)+G)/4.
   18 EPS=EPS+ABS(T(I,J)-HOLDT)
C
      IF(ITER.GE.ITMAX)GO TO 14
      IF(EPS-EPSMAX)14,3,3
   14 WRITE(6,201)ITER,EPS
      DO 11 J=M,6,-1
   11 WRITE(6,202)(T(I,J) ,I=1,5)


  201 FORMAT(1H1,///,20X,,ITER=,,I5,///,20X,,EPS=,,E14.9)
  202 FORMAT(//,2X,11E10.4)
      DO 111 J=5,1,-1
  111 WRITE(6,202)(T(I,J),I=1,N)
      STOP
      END
```

```
RESULTS:

                    ITER=   60

                    EPS=.862121582-003

.3000+003 .1887+004 .2467+004 .2037+004 .5970+003

.3000+003 .1889+004 .2472+004 .2042+004 .5973+003

.3000+003 .1899+004 .2491+004 .2062+004 .5982+003

.3000+003 .1915+004 .2531+004 .2116+004 .6017+003

.3000+003 .1931+004 .2600+004 .2271+004 .6457+003 .6017+003 .5982+003 .5971+003 .5967+003 .5965+003 .5965+003

.3000+003 .1908+004 .2670+004 .2720+004 .2270+004 .2116+004 .2060+004 .2039+004 .2030+004 .2026+004 .2025+004

.3000+003 .1731+004 .2450+004 .2669+004 .2600+004 .2530+004 .2489+004 .2468+004 .2458+004 .2454+004 .2452+004

.3000+003 .1265+004 .1731+004 .1908+004 .1931+004 .1914+004 .1897+004 .1887+004 .1881+004 .1878+004 .1877+004

.3000+003 .3000+003 .3000+003 .3000+003 .3000+003 .3000+003 .3000+003 .3000+003 .3000+003 .3000+003 .3000+003
```