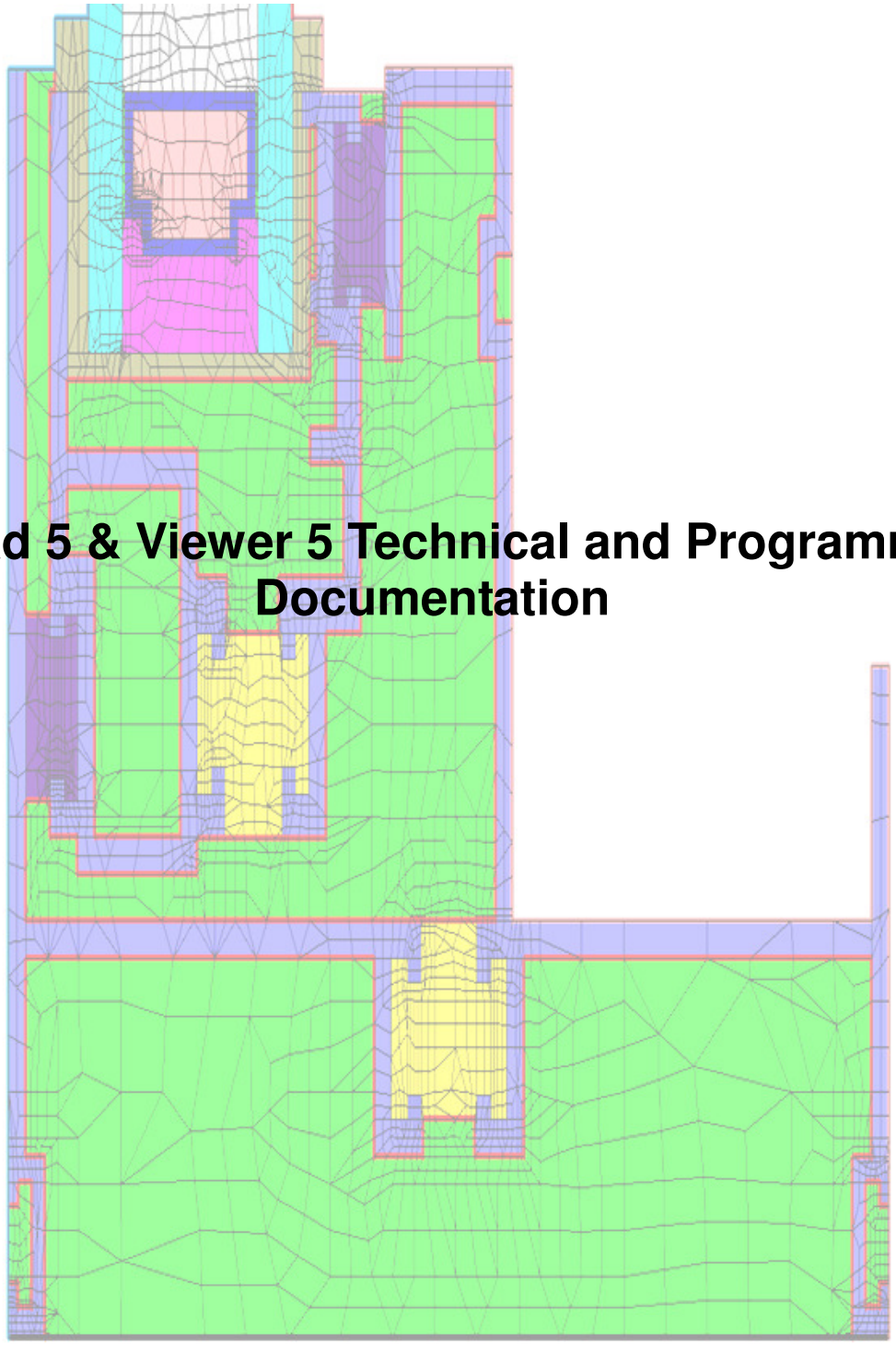
Carli, Inc.

18 Tanglewood Rd.
Amherst, MA 01002

TEL: (413) 256-4647
FAX: (413) 256-4823

home@fenestration.com
http://www.fenestration.com



Conrad 5 & Viewer 5 Technical and Programming Documentation

June 20, 2006

TABLE OF CONTENTS

1. Governing Equations

1.1. Introduction

Heat Transfer is a branch of engineering that deals with the transfer of thermal energy from one point to another within a medium or from one medium to another due to the occurrence of a temperature difference. Heat transfer may take place in one or more of its three basic forms: conduction, convection and radiation.

1.2. Governing Equation

The Governing Equation of two-dimensional heat conduction in a two-dimensional orthotropic medium Ω, under the assumption of constant physical properties, is derived from the general energy equation and is given by the following partial differential equation:

$$(k_{11}\frac{\partial^2 T}{\partial x^2}+k_{22}\frac{\partial^2 T}{\partial y^2})+Q(x,y)=0 \quad in \ \ \Omega \qquad \textbf{Eq. 1.2-1}$$

where k_{11} and k_{22} are conductivities in the x and y directions, respectively, and Q(x, y) is the known internal heat generation per unit volume. For a nonhomogeneous conducting medium, the conductivities k_{ij} are functions of position (x, y). For an isotropic medium, we set $k_{11}=k_{22}=k$ in equation (Eq. 1.2-1) and obtain the Poisson equation:

$$\frac{\partial}{\partial x}(k\frac{\partial T}{\partial x})+\frac{\partial}{\partial y}(k\frac{\partial T}{\partial y})+Q(x,y)=0 \qquad \textbf{Eq. 1.2-2}$$

$$k(\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial y^2})+Q(x,y)=0 \quad in \ \ \Omega \qquad \textbf{Eq. 1.2-3}$$

For medium without internal heat generation equation (Eq. 1.2-3) becomes:

$$k(\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial y^2})=0 \qquad \textbf{Eq. 1.2-4}$$

1.3. Boundary Conditions

To complete description of the general problem posed in the previous sections, suitable boundary and initial conditions are required. Boundary Conditions are most easily understood and described by considering the fluid mechanics separate from other transport processes. The magnitude of heat flux vector normal to the boundary is given by Fourier's law:

$$q_f+q_c+q_r=-k(\frac{\partial T}{\partial x}\overrightarrow{n_x}+\frac{\partial T}{\partial y}\overrightarrow{n_y}+\frac{\partial T}{\partial Z}\overrightarrow{n_z}) \qquad \textbf{Eq. 1.3-1}$$

There are three types of boundary conditions:

1.3.1. With Defined Temperature (Dirichlet or essential condition)

This is referring to boundary conditions with defined temperature on boundary surface as function of time and space:

$$T = f(t, x, y, z) \quad \textbf{on } \Gamma_T \qquad \textbf{Eq. 1.3-2}$$

where T is temperature on surface, t is time and x, y, z are surface coordinates, or special case with constant temperature on boundary surface:

$$T = T_{const} = const. \quad \textbf{on } \Gamma_T \qquad \textbf{Eq. 1.3-3}$$

1.3.2. With Defined Flux (Neumann or natural condition)

This is referring to boundary conditions with defined flux on boundary surface as function of time and space:

$$q_f = f(t, x, y, z) \quad \textbf{on } \Gamma_Q \qquad \textbf{Eq. 1.3-4}$$

where q_f is heat flux on surface, t is time and x, y, z are surface coordinates, or special case with Constant Flux on boundary surface:

$$q_f = q_{const} = const. \quad \textbf{on } \Gamma_Q \qquad \textbf{Eq. 1.3-5}$$

there is one special case of this boundary condition known as Adiabatic Boundary condition defined with following equation:

$$q_f = q_a = 0 \qquad \textbf{Eq. 1.3-6}$$

This means that there is no heat flux exchange between adiabatic surface and surrounding space.

1.3.3. Newton's Law

This is referring to boundary conditions with defined surrounding space temperature and defined law of heat flux exchange between surface of the body and that surrounding space. In most cases this is defined with Newton's law of convection heat transfer:

$$q_a = \pm h(T_a(t) - T_o(t)) \qquad \textbf{Eq. 1.3-7}$$

where q_a is heat flux on surface, h is heat transfer coefficient, $T_a(t)$ is temperature on body surface and $T_o(t)$ is temperature of surrounding space. There are several cases of boundary conditions which refer on Newton's Law:

1.3.3.1. Convection Boundary Condition

Convection boundary condition is defined by following equation:

$$q_c = h_c(T, t, x, y, z) * (T - T_c) \qquad \textbf{Eq. 1.3-8}$$

where q_c is convective heat flux, h_c is the convective heat transfer coefficient which, in general, depends on the location on the boundary (x,y,z), temperature (T) and time (t),

and T_c is a reference (or sink) temperature for convective transfer. In most of cases coefficient h_c is constant and in that case equation (Eq. 1.3-8) becomes:

$$q_c = h_c(T - T_c) \qquad \textbf{Eq. 1.3-9}$$

1.3.3.2. Radiation Boundary Condition (without enclosure)

Radiation boundary condition (without enclosure) is defined by following equation:

$$q_r = \varepsilon\sigma(T^4 - T_r^4) \qquad \textbf{Eq. 1.3-10}$$

where q_r is radiative part of heat flow, ε is boundary emissivity, σ is Stefan-Boltzmann constant and T_r is a reference temperature for radiative transfer. Equation (Eq. 1.3-10) also can be shown in following shape:

$$q_r = h_r(T,t,x,y,z) * (T - T_r) \qquad \textbf{Eq. 1.3-11}$$

where h_r is the linearized effective radiation heat transfer coefficient calculated by equation:

$$h_r(T,t,x,y,z) = \varepsilon\sigma(T + T_r)(T^2 + T_r^2) \qquad \textbf{Eq. 1.3-12}$$

Note that this boundary condition is appropriate when a body or surface radiates to a black body environment that can be characterized by a single temperature.

1.3.3.3. Material Interface

Another condition that is of concern when a material interface between two or more solid region is the problem of gap or contact resistance. The boundary or interface conditions in this situation are the usual continuity conditions on temperature and heat flux since the gap resistance is dictated by property variations. A more mathematical representation of contact resistance provides that the heat flux across the interface be described by an internal boundary condition:

$$q_{gap} = h_{gap}(T_{gap},t,x,y,z)(T_M - T_S) \qquad \textbf{Eq. 1.3-13}$$

where h_{gap} is an effective heat transfer coefficient for the contact surface, and T_{gap} is an average temperature between T_M and T_S. The subscripts M and S designate the "master" and "slave" sides of the contact surface, a distinction that is important in the numerical implementation of this condition.

1.3.3.4. Condensation Resistance Modeling

Heat flux on indoor surfaces of the fenestration systems can also be presented as convection:

$$q_{fenestation} = h_{fen}(T_m,t,x,y,z,gas\_properties) * (T_1 - T_2) \qquad \textbf{Eq. 1.3-14}$$

where $q_{fenestration}$ is heat flux through indoor side of fenestration system, h_{fen} is condensation resistance factor which depend of mean temperature (T_m), time (t), location on the inside boundary (x,y,z) and gas properties (density, thermal conductivity,

dynamic viscosity, specific heat), T_1 (hot side) and T_2(cold side) temperatures of indoor sides of the fenestration system (Figure 1-1).

The method is based on the use of the variable convective heat transfer coefficients on the vertical sides of IGU cavity and the simple radiative heat transfer exchange between the cavity sides.

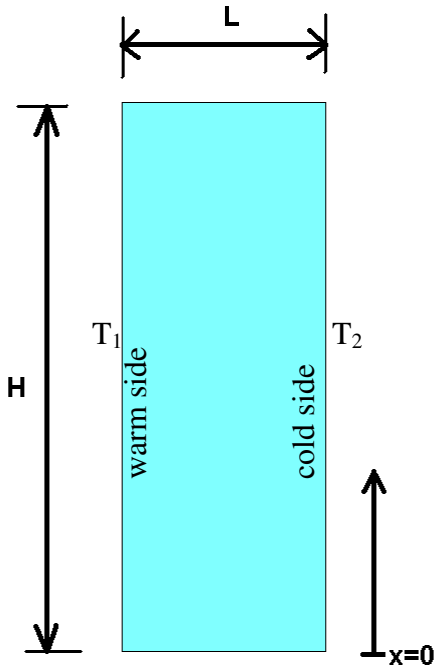


Figure 1-1: IGU of the Fenestration System

There are two criteria for condensation resistance factor calculations which depend of laminar or turbulent regime of heat flow:

Conduction Regime ($G_{rl}P_r \le 500\frac{H}{L}$)

Boundary Layer Regime ($G_{rl}P_r > 500\frac{H}{L}$)

where G_{rl} is Grashoff number and P_r is Prandtl number. Grashoff number is calculated by following equation:

$$G_{rl} = \frac{g\beta\Delta TL^3}{\upsilon^2} \qquad \textbf{Eq. 1.3-15}$$

where is g =9.807 $\frac{m}{s^2}$ (acceleration due to gravity), thermal expansion coefficient $\beta = \frac{1}{T_m}$, $\Delta T = |T_1 - T_2|$, L is cavity width (Figure 1-1) and υ is kinematics viscosity which is equal:

$$\upsilon = \frac{\mu}{\rho} \qquad \textit{Eq. 1.3-16}$$

where μ is dynamic viscosity and ρ is density (gas properties).

Prandtl number is equal:

$$P_{rl} = \frac{\mu}{\alpha} \qquad \textit{Eq. 1.3-17}$$

where μ is dynamic viscosity and temperature conductivity coefficient α is:

$$\alpha = \frac{k}{\rho C_p} \qquad \textit{Eq. 1.3-18}$$

where k is thermal conductivity, ρ is density and C_p is specific heat.

In order to calculate condensation resistance, starting and departing corners must be determined. To determine which corner is starting and which is departing see Figure 1-2 and Figure 1-3:

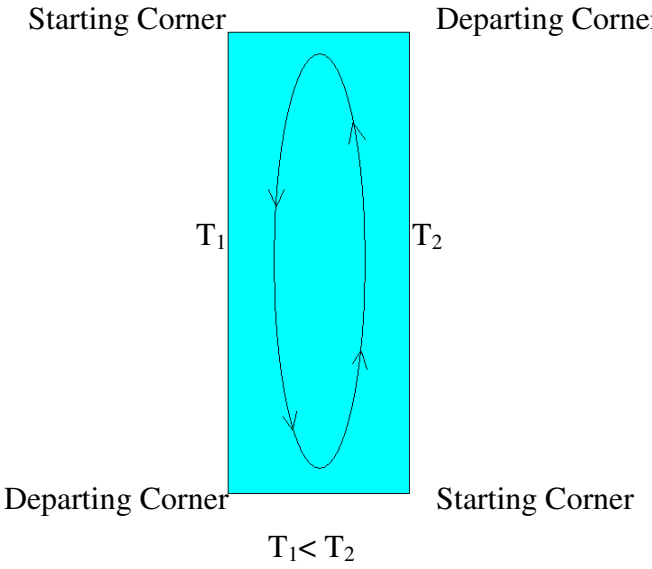


Figure 1-2: Gas (Gas Mixtures) Flow Direction in case $T_1<T_2$

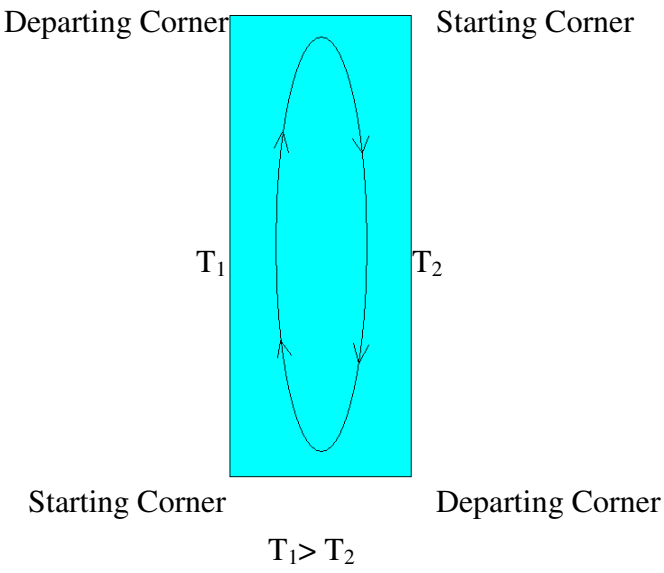


Figure 1-3: Gas (Gas Mixtures) Flow Direction in case $T_1>T_2$

Lengths of starting and departing corners are determined by following equations:

$$x_s = 0.0077 * L * (G_{rl})^{0.8571} \qquad \textbf{Eq. 1.3-19}$$

and

$$x_d = 0.00875 * L * (G_{rl})^{0.75} \qquad \textbf{Eq. 1.3-20}$$

where x_s and x_d are length of starting and departing corners (Figure 1-4).

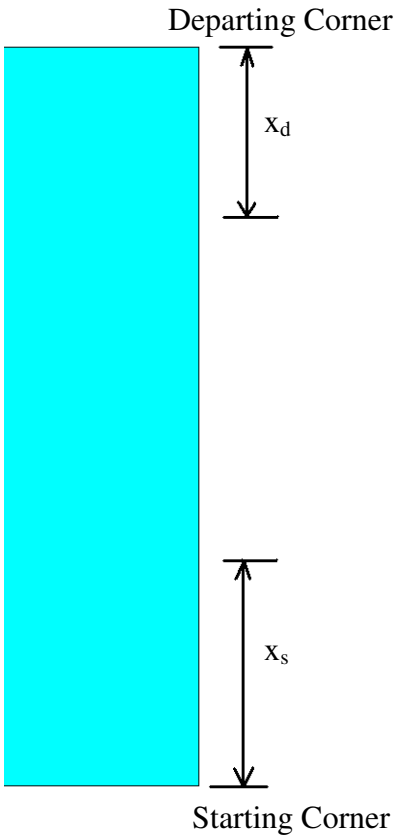


Figure 1-4: Starting and Departing Regions (Warm Side)

Therefore, in depending of regime, condensation resistance factor in fenestration system is:

a) Convective Part

i) Conduction Regime $G_{rl}P_r \le 500\frac{H}{L}$

- starting corner

$$h_c = h_s = 0.256\frac{k}{L^{0.72}}(G_{rl})^{0.24}x^{-0.28} \qquad 0 \le x \le x_s \qquad \textbf{Eq. 1.3-21}$$

$$h_c = \frac{k}{L} \qquad x_s \le x \le 0.1 \qquad \textbf{Eq. 1.3-22}$$

- departing corner

$$h_c = h_d = 2.58\frac{k}{L^{1.2}}(G_{rl})^{-0.15}x^{0.2} \qquad 0 \le x \le x_d \qquad \textbf{Eq. 1.3-23}$$

$$h_c = \frac{k}{L} \qquad x_d \le x \le 0.1 \qquad \textbf{Eq. 1.3-24}$$

ii) **Boundary Layer Regime** ($G_{rl}P_r > 500\frac{H}{L}$)

- starting corner

$$h_c = h_s = 0.231 * k * \frac{G_{rl}^{0.3}}{L^{0.9}} x^{-0.1}(0.83 - 0.6 * \frac{x}{A * L})^{1.3} \qquad \textit{Eq. 1.3-25}$$

- departing corner

$$h_c = h_d = 0.231 * k * \frac{G_{rl}^{0.3}}{L^{0.9}} (H - x)^{-0.1}(0.83 - 0.6\frac{H - x}{AL})^{1.3} \qquad \textit{Eq. 1.3-26}$$

where $A = \frac{H}{L}$

b) Radiative Part

$$h_r = 4\frac{\sigma}{(\frac{1}{\varepsilon_1} + \frac{1}{\varepsilon_2} - 1)} * T_m^{\ 3} \qquad \textit{Eq. 1.3-27}$$

and finally:

$$h_{fen} = h_c + h_r \qquad \textit{Eq. 1.3-28}$$

II) Gas (Mixture) Properties

Purpose of gas property calculations is to find coefficients of convective/conductive heat transfer in gas filled space between isothermal solid layers. Gas (mixture) properties are thermal conductivity, dynamic viscosity, density, specific heat and Prandtl number.

a) Gas Properties

i) Density

The density of fill gases in windows is calculated using the perfect gas law:

$$\rho = \frac{pM}{\Re T_m} \qquad \textit{Eq. 1.3-29}$$

where ρ is density, p is pressure, M is molecular mass, $\Re$ is universal gas constant (=8314,51 $\frac{J}{kmol * K}$) and T_m is mean gas fill temperature.

ii) ii) Specific heat capacity (C_p), dynamic viscosity (μ) and thermal conductivity (k)

The specific heat capacity, C_p, and the transport properties μ and k are evaluated using linear functions of temperature. For example, the viscosity can be expressed as:

$$\mu = a + bT_m \qquad \textit{Eq. 1.3-30}$$

Values of a and b coefficients appropriate for calculating C_p, μ and k for a variety of fill gases are listed in ISO15099 Standard (Annex B).

b) Gas Mixtures

The density, conductivity, viscosity and specific heat of gas mixtures can be calculated as a function of corresponding properties of individual constituents.

i) Molecular mass

$$M_{mix} = \sum_{i=1}^{n} x_i M_i \qquad \textit{Eq. 1.3-31}$$

where x_i is mole fraction of the i^{th} gas component in a mixture of n gases.

ii) Density

$$\rho = \frac{pM_{mix}}{\Re T_m} \qquad \textit{Eq. 1.3-32}$$

iii) Specific Heat

$$C_{pmix} = \frac{\overline{C_{pmix}}}{\overline{M_{mix}}} \qquad \textit{Eq. 1.3-33}$$

where:

$$\overline{C_{pmix}} = \sum_{i=1}^{n} x_i \overline{C_{p,i}} \qquad \textit{Eq. 1.3-34}$$

and molar specific heat of the i^{th} gas is:

$$\overline{C_{p,i}} = C_{p,i}\overline{M_i} \qquad \textit{Eq. 1.3-35}$$

iv) Viscosity

$$\mu_{mix} = \sum_{i=1}^{n} \frac{\mu_i}{\{1 + \sum_{\substack{j=1 \\ j \neq i}}^{n} \phi_{i,j}^{\mu} \frac{x_j}{x_i}\}} \qquad \textit{Eq. 1.3-36}$$

where

$$\phi_{i,j}^{\mu} = \frac{[1 + (\frac{\mu_i}{\mu_j})^{\frac{1}{2}} (\frac{\overline{M_j}}{\overline{M_i}})^{\frac{1}{4}}]^2}{2\sqrt{2} * [1 + \frac{\overline{M_i}}{\overline{M_j}}]^{\frac{1}{2}}} \qquad \textit{Eq. 1.3-37}$$

v) Thermal conductivity

$$k_{mix} = k'_{mix} + k''_{mix} \qquad \textit{Eq. 1.3-38}$$

where k' is the monatomic thermal conductivity and k'' is included to account for additional energy moved by the diffusional transport of internal energy in polyatomic gases.

$$k'_{mix} = \sum_{i=1}^{n} \frac{k'_i}{\{1 + \sum_{\substack{j=1 \\ j \neq i}}^{n} \psi_{i,j} \frac{x_j}{x_i}\}} \qquad \textit{Eq. 1.3-39}$$

and,

$$\psi_{i,j} = \frac{[1 + (\frac{k'_i}{k'_j})^{\frac{1}{2}} (\frac{\overline{M_i}}{\overline{M_j}})^{\frac{1}{4}}]^2}{2\sqrt{2} * [1 + (\frac{\overline{M_i}}{\overline{M_j}})]^{\frac{1}{2}}} * [1 + 2.41 \frac{(\overline{M_i} - \overline{M_j})(\overline{M_i} - 0.142\overline{M_j})}{(\overline{M_i} + \overline{M_j})^2}] \qquad \textit{Eq. 1.3-40}$$

and

$$k''_{mix} = \sum_{i=1}^{n} \frac{k''_i}{\{1 + \sum_{\substack{j=1 \\ j \neq i}}^{n} \phi^{\lambda}_{i,j} \frac{x_i}{x_j}\}} \qquad \textit{Eq. 1.3-41}$$

where, the previous expression for $\phi_{i,j}$ can also be written as

$$\phi^{\lambda}_{i,j} = \frac{[1 + (\frac{k_i}{k_j})^{\frac{1}{2}} (\frac{\overline{M_j}}{\overline{M_i}})^{\frac{1}{4}}]^2}{2\sqrt{2} * [1 + \frac{\overline{M_i}}{\overline{M_j}}]^{\frac{1}{2}}} \qquad \textit{Eq. 1.3-42}$$

1.3.3.5. Radiation Enclosure Boundary Condition

Radiant energy exchange between neighboring surfaces of region or between a region and its surroundings can produce large effects in the overall heat transfer problem. Thought the radiation effects generally enter the heat transfer problem only thought the boundary conditions, the coupling is especially strong due to the nonlinear dependence of the radiation on the surface temperature.

Enclosure or surface-to-surface radiation is limited to diffuse gray, opaque, surfaces. This assumption implies that all energy emitted or reflected from a surface is diffuse. Further, surface emissivity ε, absorbtivity α, and reflectivity ρ are independent of wavelength and direction so that:

$$\varepsilon(T) = \alpha(T) = 1 - \rho(T) \qquad \textit{Eq. 1.3-43}$$

Each individual area or surface that is considered in the radiation process must be at a uniform temperature; emitted and reflected energy are uniform over each such surface. Heat flux through i^{th} enclosure (q_{ri}) surface is given by following equation:

$$q_{ri} = \varepsilon_i \sigma T_i^4 - \alpha_i H_i \qquad \textbf{Eq. 1.3-44}$$

where ε_i emissivity of i^{th} surface, T_i is temperature of i^{th} surface, α_i is absorbtivity of i^{th} surface, σ is Stefan-Boltzmann constant and H_i is irradiation of the surface, and for i^{th} surface it is equal to:

$$H_i = \frac{1}{1-\varepsilon_i}(B_i - \varepsilon_i \sigma T_i^4) \qquad \textbf{Eq. 1.3-45}$$

where B_i is radiosity of the surface "i" and it is equal:

$$B_i = \varepsilon_i \sigma T_i^4 + (1-\varepsilon_i)\sum_{j=1}^{n} F_{ij} B_j \qquad \textbf{Eq. 1.3-46}$$

where F_{ij} designates view factor between i^{th} and j^{th} surface. Equation (Eq. 1.3-46) represents a system of n linear algebraic equations which is solved for B_i.

1.3.3.6. View Factors

The view factor is defined as the fraction of energy leaving a surface that arrives at a second surface. For surfaces with finite areas, the view factors are defined by

$$F_{k-j} = \frac{1}{A_k}\int_{A_k}\int_{A_j} \frac{\cos\theta_k \cos\theta_j}{\pi S^2} dA_k dA_j \qquad \textbf{Eq. 1.3-47}$$

where S is distance from a point on surface A_j to a point on surface A_k. The angles θ_j and θ_k are measured between the line S and the normal to the surface as shown in Figure 1-5.

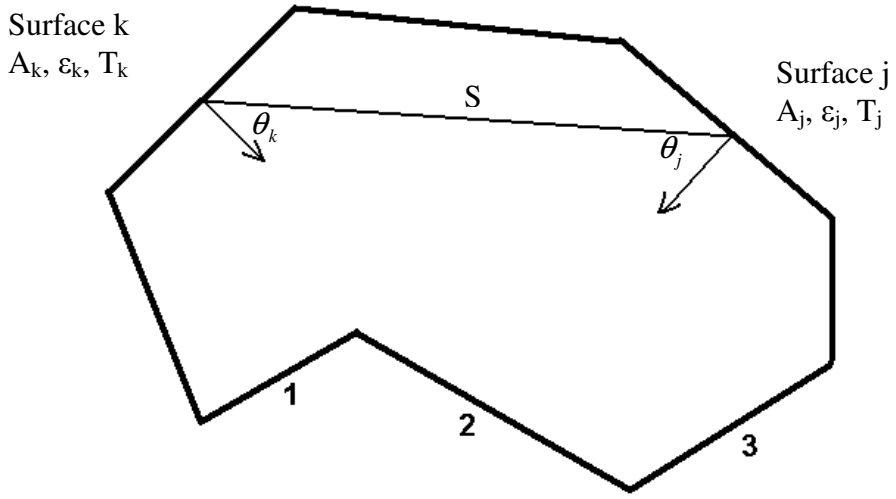


Figure 1-5: Nomenclature for enclosure radiation

From equation (Eq. 1.3-47), following equation is obtained:

$$A_k F_{k-j} = A_j F_{j-k} \qquad \textit{Eq. 1.3-48}$$

There are several ways to calculate view factors. One of them is "**cross-string"** rule which is illustrated in Figure 1-6

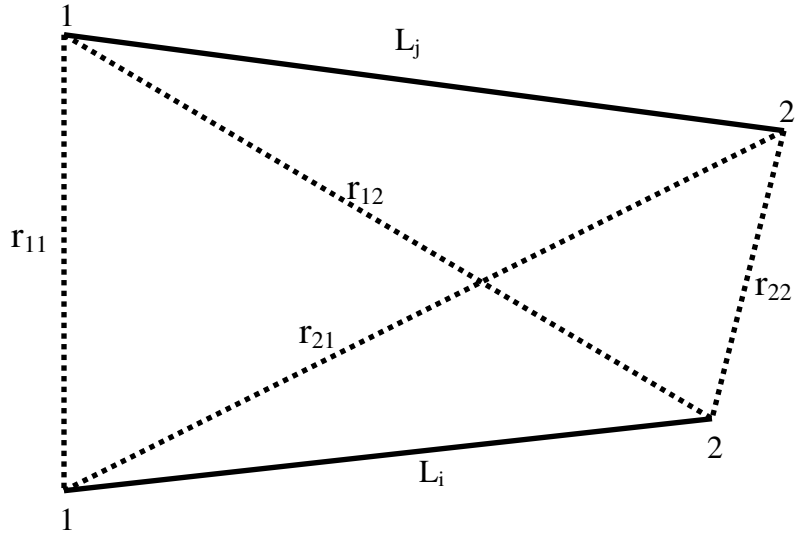


Figure 1-6: Cross-string rule

and given by following equation:

$$F_{ij} = \frac{r_{12} + r_{21} - (r_{11} + r_{22})}{2L_i} \qquad \textit{Eq. 1.3-49}$$

When partial, or third shadowing exist, the two radiating surfaces are subdivided into n finite subsurfaces and contribution to the summation in equation

$$F_{ij} = \sum_{k=1}^{n} \sum_{l=1}^{n} F_{kl} \qquad \textit{Eq. 1.3-50}$$

of those subsurfaces in which ray r_{kl} intersects a shadowing surface is excluded (Figure 1-7).

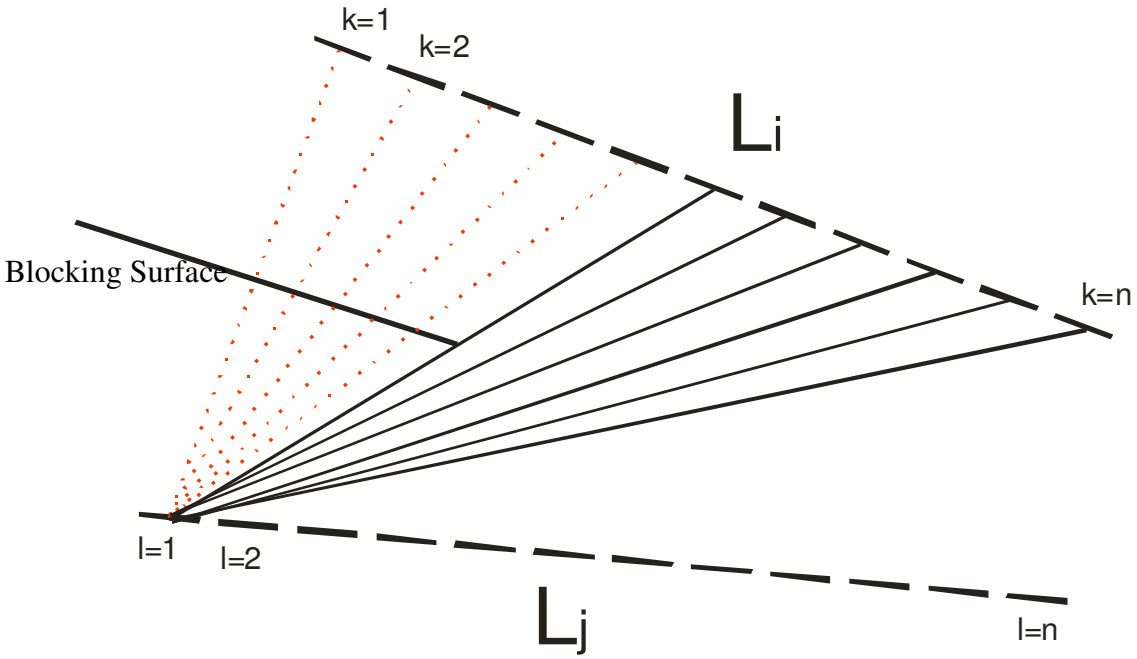


Figure 1-7: Third Surface Shadowing

2. Basic Concepts of the Finite-Element Method

Regardless of the physical nature of the problem, a standard finite-element method primarily involves the following steps:

1. Definition of the Problem and its Domain
2. Discretization of the Domain
3. Identification of State Variable(s)
4. Formulation of the Problem
5. Establishing Coordinate Systems
6. Constructing Approximate Functions for the Elements
7. Obtain Element Matrices and Equations

8. Coordinate Transformations
9. Assembly of Element Equations
10. Introduction of Boundary Conditions
11. Solution of the Final Set of Simultaneous Equations
12. Interpretation of the results

2.1. Definition of the Problem and its Domain

In finite element methods, there are primarily three sources of approximation. The first one is the definition of the domain (physically and geometrically); the other two are the discretization and solution algorithms. The approximation used in defining the physical characteristics of different regions. In case of heat transfer through material domain, governing equations are defined in previous chapter.

2.2. Discretization of the Domain

Since the problem is usually defined over a continuous domain, the governing equations, with the exception of essential boundary conditions, are valid for entirety of, as well as for any portion of, that domain. This allows idealization of the domain in the form of interconnected finite-sized domains (elements) of different size and shape.

In finite-element idealization of the domain, we shall, in general, make reference to the following elements: *finite element* Ω_e and *master element* $\tilde{\Omega}_m$ *.*

Finite elements are those which, when put together, result in discrete version of the actual continuous domain. Their geometric approximations are controlled by the number of nodes utilized at the exterior of the elements to define their shape. The physical approximations are controlled by the total number of nodes utilized in defining the *trial functions* (shape functions) for state variable.

For a moment let us assume that it is possible to systematically generate the approximation temperature field function for the element Ω_e :

$$T(x,y) \approx T^e(x,y) = \sum_{j=1}^{n} T_j^e \psi_j^e(x,y) \qquad \textbf{Eq. 2.2-1}$$

where $T^e(x,y)$ represents an approximation of $T(x,y)$ over the element Ω_e , T_j^e denote the values of function $T^e(x,y)$ at selected number of points, called *element nodes*, in the element Ω_e , and $\psi_j^e(x,y)$ are the approximation functions associated with the element.

Master elements are those which are used in place of finite elements in order to facilitate computations in the element domain. Figure 2-1 illustrates an actual finite element Ω_e and corresponding master element $\tilde{\Omega}_m$ with associated coordinate axes.

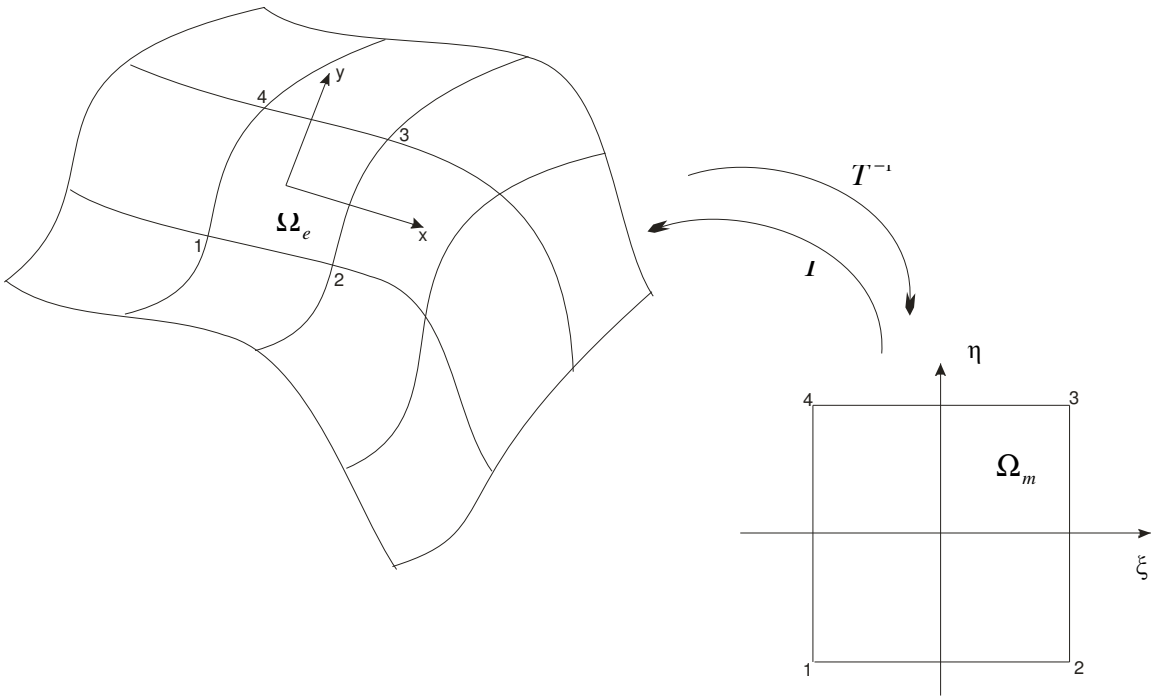


Figure 2-1: Demonstration of Coordinate for a Rectangular Finite Element

In general, the master elements are straight lines, right triangles or prisms, squares, and cubes. They are defined in reference to normalized coordinate axes (ξ, η, ζ). The actual elements can be of any shape and size.

2.3. Identification of State Variables

Until this step, no reference has been made to the physical nature of the problem. Whether it is a heat-transfer problem, fluid or solid-mechanics problem, etc., comes into the picture at this stage. The mathematical description of steady-state physical phenomena, for instance, leads to an elliptic boundary-value problem in which the formula contains the *state variable* and the *flux.* These variables are related to each other by a *constitutive equation* representing a mathematical expression of a particular physical law.

For heat transfer presented in previous chapter (and in Conrad) state variables are temperatures in element nodes (T_j^e) or temperature distribution T(x, y).

2.4. Formulation of the Problem

Vary often a physical problem is formulated either via a set of *differential equations:*

$$\overline{\overline{L}}\,\overline{u} = \overline{f} \qquad \textbf{Eq. 2.4-1}$$

with boundary conditions or by an *integral equation*:

$$\pi = \int_{\Omega} G(x, y, z, u)d\Omega + \int_{\Gamma} g(x, y, z, u)d\Gamma \qquad \textbf{Eq. 2.4-2}$$

where u present state variable(s).

For heat transfer governing equation (Eq. 1.2-1):

$$-[\frac{\partial}{\partial x}(k_{11}\frac{\partial T}{\partial x})+\frac{\partial}{\partial y}(k_{22}\frac{\partial T}{\partial y})]=Q \qquad \textbf{Eq. 2.4-3}$$

The weak form of differential equation is weighted-integral statement that is equivalent to the governing differential equation as well as associated boundary conditions:

$$0=\int_{\Omega_e}\omega[-\frac{\partial}{\partial x}(k_{11}\frac{\partial T}{\partial x})-\frac{\partial}{\partial y}(k_{22}\frac{\partial T}{\partial y})-Q(x,y)]dxdy \qquad \textbf{Eq. 2.4-4}$$

The expression in square brackets of the above equation represents a residual of the approximation of differential equation and it is called *weighted-residual statement* of equation (Eq. 1.2-1).

Note the following identities for any differentiable functions $\omega(x,y)$, $F_1(x,y)$, and $F_2(x,y)$:

$$\frac{\partial}{\partial x}(\omega F_1)=\frac{\partial \omega}{\partial x}F_1+\omega\frac{\partial F_1}{\partial x} \quad \textbf{or} \quad -\omega\frac{\partial F_1}{\partial x}=\frac{\partial \omega}{\partial x}F_1-\frac{\partial}{\partial x}(\omega F_1) \qquad \textbf{Eq. 2.4-5}$$

$$\frac{\partial}{\partial y}(\omega F_2)=\frac{\partial \omega}{\partial y}F_2+\omega\frac{\partial F_2}{\partial y} \quad \textbf{or} \quad -\omega\frac{\partial F_2}{\partial y}=\frac{\partial \omega}{\partial y}F_2-\frac{\partial}{\partial y}(\omega F_2) \qquad \textbf{Eq. 2.4-6}$$

and gradient (divergent) theorem:

$$\int_{\Omega_e}\frac{\partial}{\partial x}(\omega F_1)dxdy=\oint_{\Gamma_e}(\omega F_1)\vec{n}_x ds \qquad \textbf{Eq. 2.4-7}$$

$$\int_{\Omega_e}\frac{\partial}{\partial y}(\omega F_2)dxdy=\oint_{\Gamma_e}(\omega F_2)\vec{n}_y ds \qquad \textbf{Eq. 2.4-8}$$

where $\vec{n}_x$ and $\vec{n}_y$ are the components of unit normal vector. Using equations (Eq. 2.4-5),(Eq. 2.4-6), (Eq. 2.4-7), (Eq. 2.4-8) and (Eq. 2.4-4) with:

$$F_1=k_{11}\frac{\partial T}{\partial x} \quad \textbf{and} \quad F_2=k_{22}\frac{\partial T}{\partial y} \qquad \textbf{Eq. 2.4-9}$$

we obtain

$$0=\int_{\Omega_e}(\frac{\partial \omega}{\partial x}k_{11}\frac{\partial T}{\partial x}+\frac{\partial \omega}{\partial y}k_{22}\frac{\partial T}{\partial y}-\omega Q)dxdy-\oint_{\Gamma_e}\omega(k_{11}\frac{\partial T}{\partial x}\vec{n}_x+k_{22}\frac{\partial T}{\partial y}\vec{n}_y)ds \qquad \textbf{Eq. 2.4-10}$$

2.5. Establishing Coordinate Systems

There are primarily two reasons for choosing special coordinate axes for elements in addition to the global axes for entire system. The first is case of constructing the trial functions for the elements and the second is ease integration over the elements.

Once the coordinate axes are established, the element equations are ordinarily computed first in master element $\tilde{\Omega}_m$. They are then transformed into Ω_e and finally into the global system for assembly.

2.6. Constructing Approximate Functions for the Elements

Once the state variable(s) and the local coordinate system have been chosen, the functions can be approximated in numerous ways. The reader is reminded that there are two entities that need to be approximated. The first is *physical* (the state variable) and the second is geometrical (the shape of element). The analyst must decide whether to approximate *physics* and *geometry* equally or give preference to one or the other in various regions of the domain. This leads to the three different categories of elements with m and n representing the degree of approximation for element shape and state variable, respectively:

Subparametric (m<n)

Isoparametric (m=n)

Superparametric (m>n)

2.7. Linear Elements

2.7.1.1. Shape Functions for Master Line

Master line element is shown in Figure 2-2:

Figure 2-2: Master Linear Line Element

and shape functions are:

$$\begin{Bmatrix} \tilde{\psi}_1^e \\ \tilde{\psi}_2^e \end{Bmatrix} = \frac{1}{2} \begin{Bmatrix} 1-\xi \\ 1+\xi \end{Bmatrix} \qquad \textit{Eq. 2.7-1}$$

2.7.1.2. Shape functions for master rectangular element

Master rectangular element is shown in Figure 2-3:

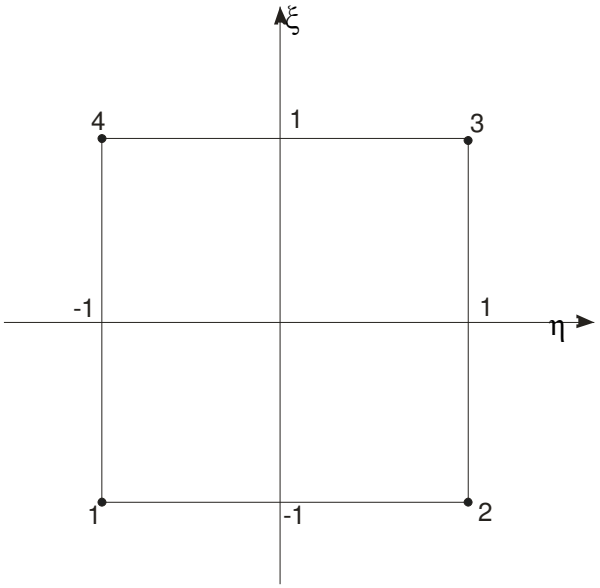


Figure 2-3: Master Linear Rectangular Element

and shape functions for this element are:

$$\begin{Bmatrix} \tilde{\psi}_1^e \\ \tilde{\psi}_2^e \\ \tilde{\psi}_3^e \\ \tilde{\psi}_4^e \end{Bmatrix} = \frac{1}{4} \begin{Bmatrix} (1-\xi)(1-\eta) \\ (1+\xi)(1-\eta) \\ (1+\xi)(1+\eta) \\ (1-\xi)(1+\eta) \end{Bmatrix} \qquad \textit{Eq. 2.7-2}$$

2.7.1.3. Shape Functions for Triangular element

The linear interpolation functions in global coordinate system for the three-node

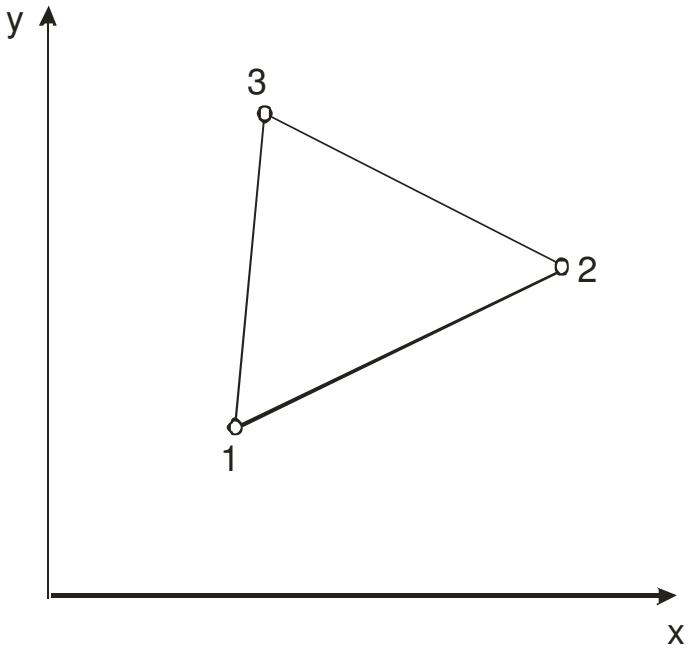


Figure 2-4 : The Linear Triangular Element in Global Coordinates

triangle (see Figure 2-4) are (see [3]):

$$\psi_i^e = \frac{1}{2A_e}(\alpha_i^e + \beta_i^e x + \gamma_i^e y) = L_i \quad i = 1,\ 2,\ 3 \qquad \textbf{Eq. 2.7-3}$$

where A_e is the area of the triangle, and α_i^e, β_i^e and γ_i^e are geometric constants known in terms of the nodal coordinates (x_i, y_i):

$$\begin{aligned} \alpha_i^e &= x_j y_k - x_k y_j \\ \beta_i^e &= y_j - y_k \\ \gamma_i^e &= -(x_j - x_k) \end{aligned} \qquad \textbf{Eq. 2.7-4}$$

Here the subscripts are such that $i \neq j \neq k$, and *i, j* and *k* permute in natural order. Eq. 2.7-3 is used to mapping from global to local coordinate system where L_i presents nodal function in local coordinate system. Inverse mapping is presented by following equations:

$$x = \sum_{i=1}^{3} x_i L_i;\ \ y = \sum_{i=1}^{3} y_i L_i \qquad \textbf{Eq. 2.7-5}$$

Note also equation for calculating triangle area:

$$2A_e = \begin{vmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{vmatrix} \qquad \textbf{Eq. 2.7-6}$$

2.7.1.4. Shape functions for master 3D rectangular element

Master 3D rectangular element is shown in Figure 2-5:

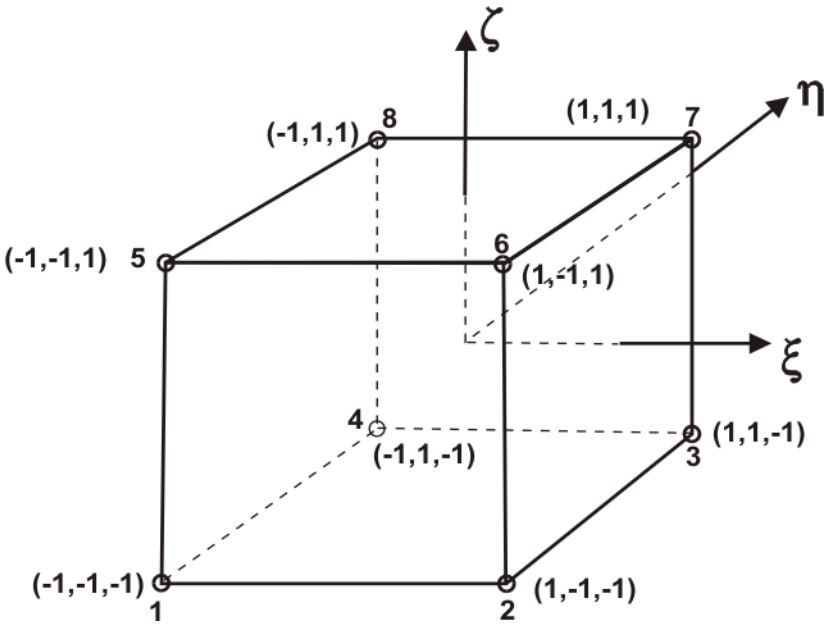


Figure 2-5: Master Linear 3D Rectangular Element

and shape functions are:

$$\begin{Bmatrix} \tilde{\psi}_1^e \\ \tilde{\psi}_2^e \\ \tilde{\psi}_3^e \\ \tilde{\psi}_4^e \\ \tilde{\psi}_5^e \\ \tilde{\psi}_6^e \\ \tilde{\psi}_7^e \\ \tilde{\psi}_8^e \end{Bmatrix} = \frac{1}{8} \begin{Bmatrix} (1-\xi)(1-\eta)(1-\zeta) \\ (1+\xi)(1-\eta)(1-\zeta) \\ (1+\xi)(1+\eta)(1-\zeta) \\ (1-\xi)(1+\eta)(1-\zeta) \\ (1-\xi)(1-\eta)(1+\zeta) \\ (1+\xi)(1-\eta)(1+\zeta) \\ (1+\xi)(1+\eta)(1+\zeta) \\ (1-\xi)(1+\eta)(1+\zeta) \end{Bmatrix} \qquad \textit{Eq. 2.7-7}$$

2.8. Quadratic Elements

2.8.1.1. Shape Functions for Master Line

Master Quadratic line element is shown in Eq. 2.8-1:

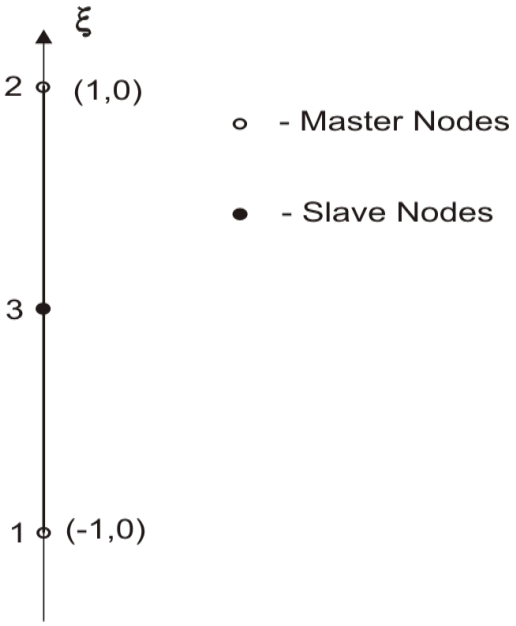


Figure 2-6: Master Quadratic Line Element

and shape functions for line element are:

$$\begin{Bmatrix} \tilde{\psi}_1^e \\ \tilde{\psi}_2^e \\ \tilde{\psi}_3^e \end{Bmatrix} = \frac{1}{2} \begin{Bmatrix} -\xi(1-\xi) \\ \xi(1+\xi) \\ 2(1-\xi^2) \end{Bmatrix} \qquad \textit{Eq. 2.8-1}$$

2.8.1.2. Shape functions for master rectangular element

Master rectangular element is shown in Figure 2-7:

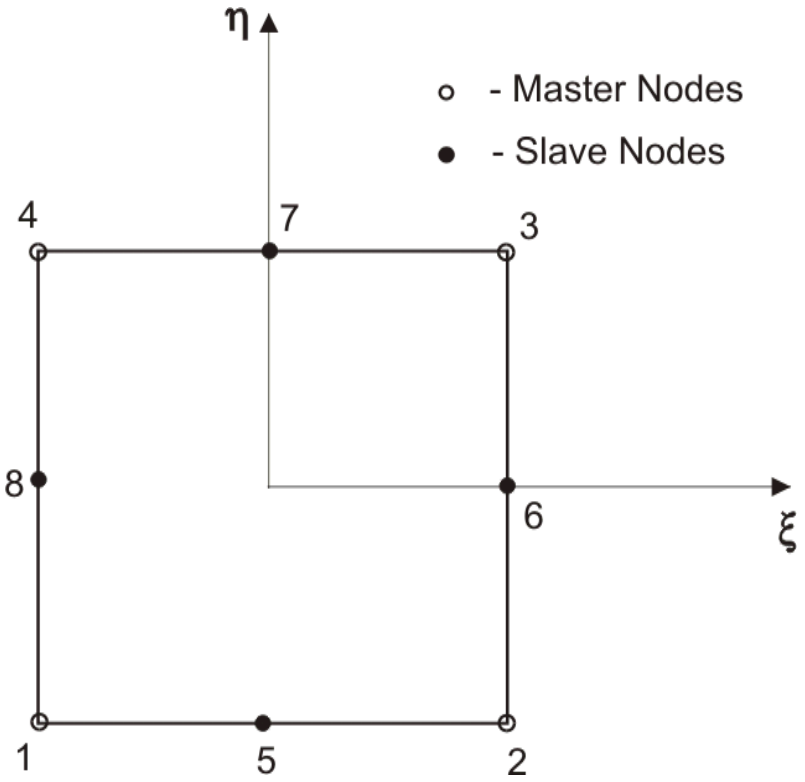


Figure 2-7: Master Rectangular Quadratic Element

and shape functions are:

$$\begin{Bmatrix} \tilde{\psi}_1^e \\ \tilde{\psi}_2^e \\ \tilde{\psi}_3^e \\ \tilde{\psi}_4^e \\ \tilde{\psi}_5^e \\ \tilde{\psi}_6^e \\ \tilde{\psi}_7^e \\ \tilde{\psi}_8^e \end{Bmatrix} = \frac{1}{4} \begin{Bmatrix} (1-\xi)(1-\eta)(-\xi-\eta-1) \\ (1+\xi)(1-\eta)(\xi-\eta-1) \\ (1+\xi)(1+\eta)(\xi+\eta-1) \\ (1-\xi)(1+\eta)(-\xi+\eta-1) \\ 2(1-\xi^2)(1-\eta) \\ 2(1+\xi)(1-\eta^2) \\ 2(1-\xi^2)(1+\eta) \\ 2(1-\xi)(1-\eta^2) \end{Bmatrix} \qquad \textit{Eq. 2.8-2}$$

2.8.1.3. Shape functions for master 3D rectangular quadratic element

Master 3D rectangular element is shown in Figure 2-8:

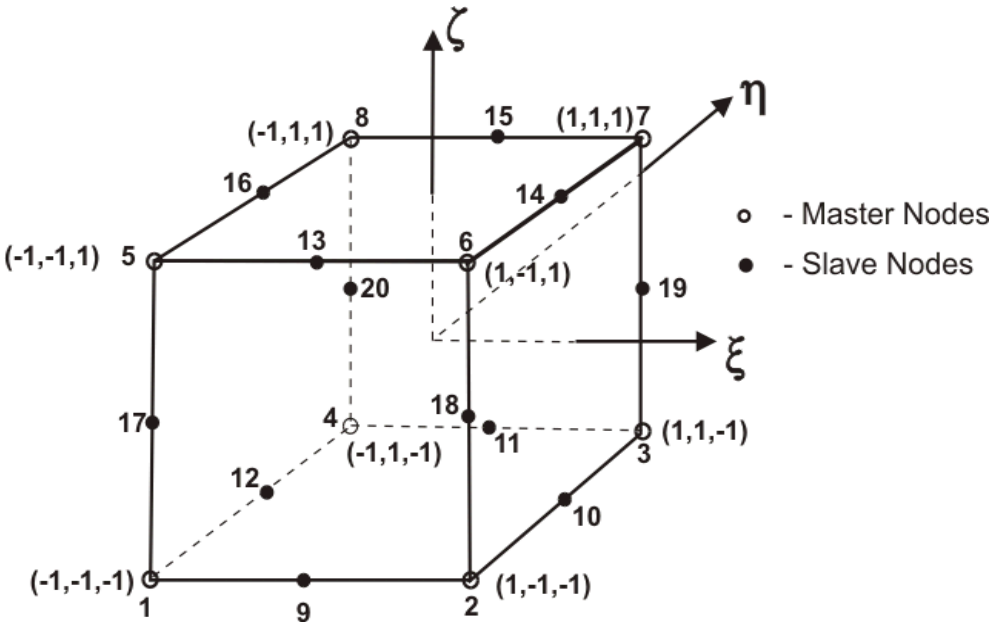


Figure 2-8: Master 3D Rectangular Quadratic Element

and shape functions are:

$$\begin{Bmatrix} \tilde{\psi}^e_1 \\ \tilde{\psi}^e_2 \\ \tilde{\psi}^e_3 \\ \tilde{\psi}^e_4 \\ \tilde{\psi}^e_5 \\ \tilde{\psi}^e_6 \\ \tilde{\psi}^e_7 \\ \tilde{\psi}^e_8 \\ \tilde{\psi}^e_9 \\ \tilde{\psi}^e_{10} \\ \tilde{\psi}^e_{11} \\ \tilde{\psi}^e_{12} \\ \tilde{\psi}^e_{13} \\ \tilde{\psi}^e_{14} \\ \tilde{\psi}^e_{15} \\ \tilde{\psi}^e_{16} \\ \tilde{\psi}^e_{17} \\ \tilde{\psi}^e_{18} \\ \tilde{\psi}^e_{19} \\ \tilde{\psi}^e_{20} \end{Bmatrix} = \frac{1}{8} \begin{Bmatrix} (1-\xi)(1-\eta)(1-\zeta)(-\xi-\eta-\zeta-2) \\ (1+\xi)(1-\eta)(1-\zeta)(\xi-\eta-\zeta-2) \\ (1+\xi)(1+\eta)(1-\zeta)(\xi+\eta-\zeta-2) \\ (1-\xi)(1+\eta)(1-\zeta)(-\xi+\eta-\zeta-2) \\ (1-\xi)(1-\eta)(1+\zeta)(-\xi-\eta+\zeta-2) \\ (1+\xi)(1-\eta)(1+\zeta)(\xi-\eta+\zeta-2) \\ (1+\xi)(1+\eta)(1+\zeta)(\xi+\eta+\zeta-2) \\ (1-\xi)(1+\eta)(1+\zeta)(-\xi+\eta+\zeta-2) \\ 2(1-\xi^2)(1-\eta)(1-\zeta) \\ 2(1-\eta^2)(1+\xi)(1-\zeta) \\ 2(1-\xi^2)(1+\eta)(1-\zeta) \\ 2(1-\eta^2)(1-\xi)(1-\zeta) \\ 2(1-\xi^2)(1-\eta)(1+\zeta) \\ 2(1-\eta^2)(1+\xi)(1+\zeta) \\ 2(1-\xi^2)(1+\eta)(1+\zeta) \\ 2(1-\eta^2)(1-\xi)(1+\zeta) \\ 2(1-\zeta^2)(1-\xi)(1-\eta) \\ 2(1-\zeta^2)(1+\xi)(1-\eta) \\ 2(1-\zeta^2)(1+\xi)(1+\eta) \\ 2(1-\zeta^2)(1-\xi)(1+\eta) \end{Bmatrix} \qquad \textit{Eq. 2.8-3}$$

2.9. Obtain Element Matrices and Equations

At this stage assume that the modeling of the problem has been completed. Let the approximate function for a steady-state problem be written as:

$$\tilde{u}_e(x,y,z) = \tilde{N}(x,y,z) * u_e \qquad \textit{Eq. 2.9-1}$$

where $\tilde{N}(x,y,z)$ is referred to as the *shape function*. It is called shape function because it contains not only the approximation made for state variables but also the coordinates of the element nodes which define the shape function of the element. The shape function can be written as:

$$\tilde{N}(x,y,z) = [N_1 \quad N_2 \quad ... \quad N_n] \qquad \textit{Eq. 2.9-2}$$

where n represents the number of nodes of the element and N_i is the shape function corresponding to node i. Substituting equation (Eq. 2.4-10) into the equation (Eq. 2.9-1) written for Ω_e, where the first term is often a quadratic form of u and its derivatives, yields:

$$\pi_e = \int_{\Omega_e} u_e^T B^T D B u_e d\Omega_e + \int_{\Gamma} u_e^T N^T p d\Gamma \qquad \textit{Eq. 2.9-3}$$

Matrix B contains the shape function and its derivatives as well as the constitutive relationships of the problem. Matrix D represents the physical parameters of the domain, and p represents disturbances at the boundaries. Carrying out the integrations (often numerically) results in the following matrix equation:

$$k_e u_e + p_e = 0 \qquad \textit{Eq. 2.9-4}$$

2.9.1. Linear Problem

For heat transfer described in previous chapter weak form of equation (Eq. 2.4-10) combined with (Eq. 2.2-1) and without boundary conditions gives:

$$0 = \int_{\Omega_e} [\frac{\partial \omega}{\partial x}(k_{11}\sum_{j=1}^{n} T_j^e \frac{\partial \psi_j^e}{\partial x}) + \frac{\partial \omega}{\partial y}(k_{22}\sum_{j=1}^{n} T_j^e \frac{\partial \psi_j^e}{\partial y}) - \omega Q] dxdy \qquad \textit{Eq. 2.9-5}$$

This equation must hold for any weight function ω. Since we need n independent equations to solve for the n unknowns, T_1^e, T_2^e, …, T_n^e, we choose n independent algebraic equations to solve for ω: $\omega = \psi_1^e, \psi_2^e, ..., \psi_n^e$. For each choice of ω we obtain an algebraic relation among $(T_1^e, T_2^e, ..., T_n^e)$. We label the algebraic equation resulting from substitution of ψ_1^e for ω into equation (Eq. 2.9-5) as the first algebraic equation. The i^{th} algebraic equation is obtained by substituting $\omega = \psi_i^e$ into equation (Eq. 2.9-5):

$$\sum_{j=1}^{n} K_{ij}^e T_j^e = Q_i^e \qquad \textit{Eq. 2.9-6}$$

where the coefficients K_{ij}^e and Q_i^e are defined by

$$K_{ij}^e = \int_{\Omega_e} (k_{11} \frac{\partial \psi_i^e}{\partial x} \frac{\partial \psi_j^e}{\partial x} + k_{22} \frac{\partial \psi_i^e}{\partial y} \frac{\partial \psi_j^e}{\partial y}) dxdy \qquad \textbf{Eq. 2.9-7}$$

$$Q_i^e = \int_{\Omega_e} Q \psi_i^e(x, y) dxdy \qquad \textbf{Eq. 2.9-8}$$

In matrix notation, equation (Eq. 2.9-6) takes the form

$$[K^e]\{T^e\} = \{Q^e\} \qquad \textbf{Eq. 2.9-9}$$

The matrix $[K^e]$ is called the *coefficient matrix*, or conductivity matrix. Equation (Eq. 2.9-9) is solved by $\{T^e\}$.

2.9.2. Nonlinear problem

For nonlinear problem following equation will be replaced in equation (Eq. 2.4-10):

$$T(x, y) + \Delta T(x, y) \approx T^e(x, y) + \Delta T^e(x, y) = \sum_{j=1}^{m} T_j^e \psi_j^e(x, y) + \sum_{j=1}^{m} \Delta T_j^e \psi_j^e(x, y) \qquad \textbf{Eq. 2.9-10}$$

substituting (Eq. 2.9-10) into the (Eq. 2.4-10) without boundary conditions:

$$0 = \int_{\Omega_e} [\frac{\partial \omega}{\partial x} \{k_{11} (\sum_{j=1}^{n} T_j^e \frac{\partial \psi_j^e}{\partial x} + \sum_{j=1}^{n} \Delta T_j^e \frac{\partial \psi_j^e}{\partial x})\} + \frac{\partial \omega}{\partial y} \{k_{22} \sum_{j=1}^{n} T_j^e \frac{\partial \psi_j^e}{\partial y}\} - \omega Q(T + \Delta T)] dxdy \qquad \textbf{Eq. 2.9-11}$$

where $Q(T + \Delta T)$ must be substituted with following equation:

$$\frac{\partial Q}{\partial T} = \frac{Q(T + \Delta T) - Q(T)}{\Delta T} \Rightarrow Q(T + \Delta T) = Q(T) + \frac{\partial Q}{\partial T} \Delta T \qquad \textbf{Eq. 2.9-12}$$

The i[th] algebraic equation is obtained by substituting $\omega = \psi_i^e$ into equation (Eq. 2.9-11):

$$\sum_{j=1}^{n} K_{ij}^e T_j^e + \sum_{j=1}^{n} K_{ij}^e \Delta T_j^e = Q_i^e + \sum_{j=1}^{n} Q_{ij}^{\cdot} \Delta T_j^e \qquad \textbf{Eq. 2.9-13}$$

where K_{ij}^e and Q_i^e are defined by equations (Eq. 2.9-7) and (Eq. 2.9-8) and:

$$Q_{ij}^{\cdot} = \int_{\Omega_e} Q^{\cdot} \psi_i^e(x, y) \psi_j^e(x, y) dxdy \qquad \textbf{Eq. 2.9-14}$$

In matrix notation, equation (Eq. 2.9-13) takes the form

$$[K^e - Q^{e,\cdot}]\{\Delta T^e\} = \{Q^e\} - [K^e]\{T^e\} \qquad \textbf{Eq. 2.9-15}$$

The matrix $[K^e]$ is called the *coefficient matrix*, or conductivity matrix. Equation (Eq. 2.9-15) is solved by $\{\Delta T^e\}$.

NOTE: To obtain $\psi_i^e(x,y)$ and $\psi_j^e(x,y)$ in equations (Eq. 2.9-8) and (Eq. 2.9-14) substitute equations (Eq. 2.10-1) and transformation to master element is obtained

For isotropic materials $k_{11} = k_{22}$ and conduction matrix becomes:

$$K_{ij}^e = \int_{\Omega_e} k\left(\frac{\partial \psi_i^e}{\partial x}\frac{\partial \psi_j^e}{\partial x} + \frac{\partial \psi_i^e}{\partial y}\frac{\partial \psi_j^e}{\partial y}\right)dxdy \qquad \textit{Eq. 2.9-16}$$

2.10. Coordinate Transformations

Coordinate transformations of physical entities such as vectors and matrices follow well defined rules. They are often done in the form of a Jacobian matrix.

2.10.1. Rectangular Element

For instance, let us assume that there are two different coordinate systems, for example x, y located in the element domain and ξ, η, located in the master element:

$$T: \begin{matrix} x = x(\xi,\eta) \\ y = y(\xi,\eta) \end{matrix} \qquad \textit{Eq. 2.10-1}$$

The transformation between actual element Ω_e and the master element $\tilde{\Omega}_m$ [or equivalently between (x, y) and (ξ,η)] is accomplished by a coordinate transformation of the form:

$$x = \sum_{j=1}^{m} x_j^e \tilde{\psi}_j^e(\xi,\eta), \qquad y = \sum_{j=1}^{m} y_j^e \tilde{\psi}_j^e(\xi,\eta) \qquad \textit{Eq. 2.10-2}$$

where $\tilde{\psi}_j^e$ denote the finite element interpolation functions of the master element $\tilde{\Omega}_m$.

An infinitesimal line segment (or area and volume) in one coordinate system can be transformed into another by following the usual rules of differentiation:

$$\begin{bmatrix} \frac{\partial \tilde{\psi}_i^e}{\partial \xi} \\ \frac{\partial \tilde{\psi}_i^e}{\partial \eta} \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} * \begin{bmatrix} \frac{\partial \psi_i^e}{\partial x} \\ \frac{\partial \psi_i^e}{\partial y} \end{bmatrix} \qquad \textit{Eq. 2.10-3}$$

The matrix on right-hand side of this equation is known as *Jacobian.* Equation (Eq. 2.10-3) transforms the line segments in $\tilde{\Omega}_m$ into line segments in Ω_e. The inverse transformation which defines mapping of element Ω_e back into the master element $\tilde{\Omega}_m$ follows a similar rule. This refer to as the *inverse transformation*

$$\begin{bmatrix} \frac{\partial \psi_i^e}{\partial x} \\ \frac{\partial \psi_i^e}{\partial y} \end{bmatrix} = J^{-1} * \begin{bmatrix} \frac{\partial \tilde{\psi}_i^e}{\partial \xi} \\ \frac{\partial \tilde{\psi}_i^e}{\partial \eta} \end{bmatrix}$$ ***Eq. 2.10-4***

where J^{-1} is the inverse matrix of the Jacobian:

$$J^{-1} = \frac{1}{\det} * \begin{bmatrix} \frac{\partial y}{\partial \eta} & -\frac{\partial y}{\partial \xi} \\ -\frac{\partial x}{\partial \eta} & \frac{\partial x}{\partial \xi} \end{bmatrix}$$ ***Eq. 2.10-5***

This implies that condition of $|J| > 0$ must be satisfied for every point in both domains.

For example, consider the element coefficients:

$$K_{ij}^e = \int_{\Omega_e} [k_{11}(x,y) \frac{\partial \psi_i^e}{\partial x} \frac{\partial \psi_j^e}{\partial x} + k_{22}(x,y) \frac{\partial \psi_i^e}{\partial y} \frac{\partial \psi_j^e}{\partial y}] dxdy$$ ***Eq. 2.10-6***

The integrand (i.e., expression in square brackets under the integral) is a function of global coordinates x and y. We must rewrite it in terms of ξ, η using the transformation (Eq. 2.10-4).

The functions $\psi_i^e(x,y)$ can be expressed in terms of the local coordinates ξ and η by means equation (Eq. 2.10-4). Hence, by the chain rule of partial differentiation, we have:

$$\frac{\partial \tilde{\psi}_i^e}{\partial \xi} = \frac{\partial \psi_i^e}{\partial x} \frac{\partial x}{\partial \xi} + \frac{\partial \psi_i^e}{\partial y} \frac{\partial y}{\partial \xi}$$ ***Eq. 2.10-7***

$$\frac{\partial \tilde{\psi}_i^e}{\partial \eta} = \frac{\partial \psi_i^e}{\partial x} \frac{\partial x}{\partial \eta} + \frac{\partial \psi_i^e}{\partial y} \frac{\partial y}{\partial \eta}$$ ***Eq. 2.10-8***

which gives the relation between the derivatives of ψ_i^e with respect to the global and local coordinates. Equations (Eq. 2.10-7) and (Eq. 2.10-8) can be expressed in following form:

$$\begin{Bmatrix} \frac{\partial \tilde{\psi}_i^e}{\partial \xi} \\ \frac{\partial \tilde{\psi}_i^e}{\partial \eta} \end{Bmatrix} = J * \begin{Bmatrix} \frac{\partial \psi_i^e}{\partial x} \\ \frac{\partial \psi_i^e}{\partial y} \end{Bmatrix}$$ ***Eq. 2.10-9***

where J is Jacobian matrix or inverse transformation:

$$\begin{Bmatrix} \frac{\partial \psi_i^e}{\partial x} \\ \frac{\partial \psi_i^e}{\partial y} \end{Bmatrix} = J^{-1} * \begin{Bmatrix} \frac{\partial \tilde{\psi}_i^e}{\partial \xi} \\ \frac{\partial \tilde{\psi}_i^e}{\partial \eta} \end{Bmatrix} \qquad \textbf{Eq. 2.10-10}$$

Equations (Eq. 2.10-2)-(Eq. 2.10-10) provide the necessary relations to transform integral expressions on any element Ω_e to an associated master element $\tilde{\Omega}_m$. Suppose that the finite element Ω_e can be generated by master element $\tilde{\Omega}_m$. Under the previous transformations we can write:

$$K_{ij}^e = \int_{\Omega_e} [k_{11}(x,y) \frac{\partial \psi_i^e}{\partial x} \frac{\partial \psi_j^e}{\partial x} + k_{22}(x,y) \frac{\partial \psi_i^e}{\partial y} \frac{\partial \psi_j^e}{\partial y}] dxdy = \int_{\tilde{\Omega}_m} F_{ij}(\xi,\eta) d\xi d\eta \qquad \textbf{Eq. 2.10-11}$$

and equation (Eq. 2.9-14) in local coordinate system becomes

$$Q_i^e = \int_{\Omega_e} Q \psi_i^e(x,y) dxdy = \int_{\tilde{\Omega}_m} Q * \tilde{\psi}_i^e(\xi,\eta) * \det * d\xi d\eta \qquad \textbf{Eq. 2.10-12}$$

2.10.2. Line Element

2.10.2.1. Linear Line Element

For line element consider following equation:

$$T : s = s(x,y) = \sum_{j=1}^{m} s_j^e \tilde{\psi}_j^e(\xi) \qquad \textbf{Eq. 2.10-13}$$

where s present line element in global coordinates, s_j^e are coordinates of line in global coordinates and $\tilde{\psi}_j^e(\xi)$ are functions of master line element (Eq. 2.7-1). Note also that for line elements m=2. Substituting equations (Eq. 2.7-1) in (Eq. 2.10-13):

$$s = \frac{1}{2}(s_1 + s_2) + \frac{1}{2}\xi(s_2 - s_1) \qquad \textbf{Eq. 2.10-14}$$

An infinitesimal line segment in one coordinate system can be transformed into another by following the usual rules of differentiation:

$$ds = \det * d\xi \qquad \textbf{Eq. 2.10-15}$$

and substituting equation (Eq. 2.10-13) into the (Eq. 2.10-15):

$$\det = \frac{ds}{d\xi} = \frac{1}{2}(s_2 - s_1) = \frac{1}{2}\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \qquad \textbf{Eq. 2.10-16}$$

2.10.2.2. Quadratic Line Element

Same approximation is used for quadratic line segment (see Eq. 2.10-13) just different shape functions Eq. 2.8-1 are used, which lead to:

$$s = -\frac{1}{2} * s_1 * \xi * (1-\xi) + \frac{1}{2} * s_2 * \xi * (1+\xi) + s_3 * (1-\xi^2) \qquad \textit{Eq. 2.10-17}$$

and corresponding determinant is:

$$\det = \frac{ds}{d\xi} = \frac{1}{2}(s_2 - s_1) + \xi(s_1 + s_2 - 2s_3) = \\ = \frac{1}{2}\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} + \xi\sqrt{(x_1 + x_2 - 2x_3)^2 + (y_1 + y_2 - 2y_3)^2} \qquad \textit{Eq. 2.10-18}$$

2.11. Assembly of Element Equations

The assembly of element matrix equations $(p_e = k_e u_e)$ is done according to the topological configuration of the elements after this equation is transformed into the global system. The assembly is done through the nodes as the interfaces which are common to the adjacent elements. At these nodes the continuities are established in respect to the state variable and possibly in respect to its derivatives. Sometimes this assembly is done through certain nodes only, referred to as the *primary nodes* (e.g. corner nodes), instead of to all the nodes at the interfaces. This reduces the overall size of the assembled matrix. The nodes that are not used in the assembly, the so-called *secondary nodes*, are used together with the primary nodes to increase the degree of approximation at the element level. Assume that the complete element matrix is partitioned as follows:

$$\begin{bmatrix} P_I \\ P_{II} \end{bmatrix} = \begin{bmatrix} K_{I,I} & K_{I,II} \\ K_{II,I} & K_{II,II} \end{bmatrix} \begin{bmatrix} U_I \\ U_{II} \end{bmatrix} \qquad \textit{Eq. 2.11-1}$$

in which subscripts I and II identify the portions of the equations corresponding to primary and secondary nodes, respectively. This equation can be brought to the following form:

$$P_I - K_{I,II} K_{II,II}^{-1} P_{II} = [K_{I,I} - K_{I,II} K_{II,II}^{-1} K_{II,I}] U_I \qquad \textit{Eq. 2.11-2}$$

which, in short, can be written as

$$P_e = K_e U_e \qquad \textit{Eq. 2.11-3}$$

this is the final equation to be assembled. It contains the unknown value of the function at the primary nodes only. To illustrate the assembly, let assume that domain Ω in 2D space consist of three elements (rectangular, triangular and line elements), as shown in Figure 2-9:

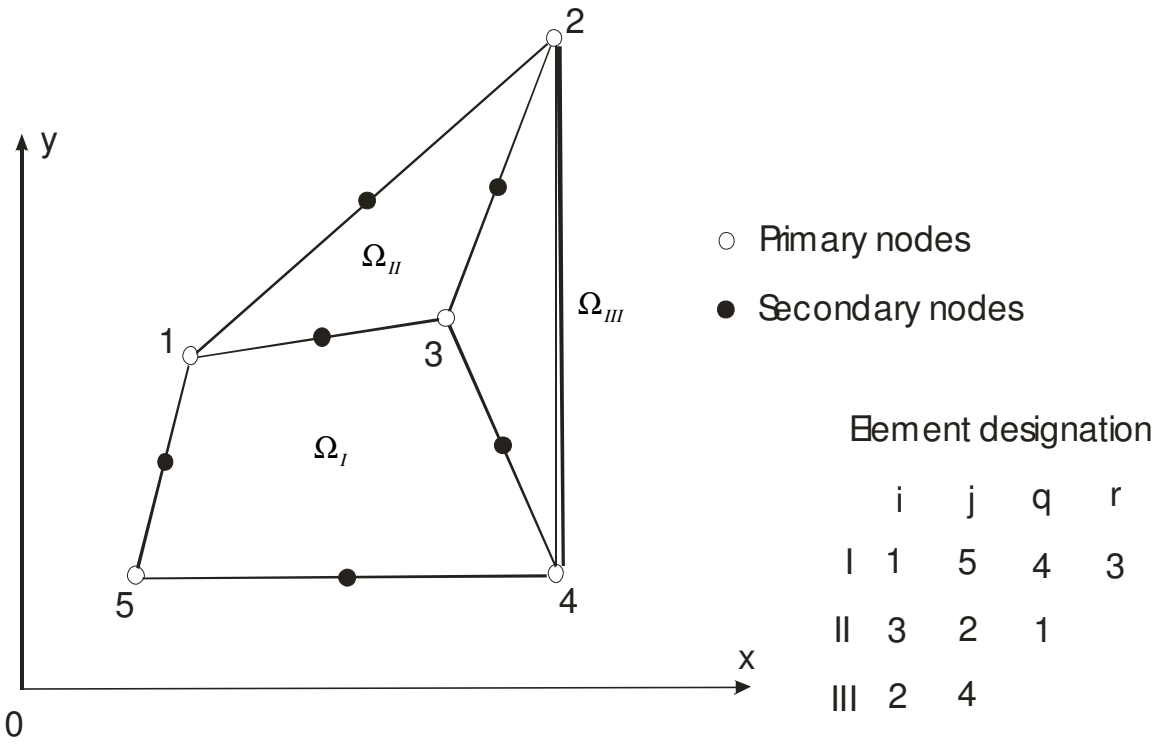


Figure 2-9: Assembly of three elements

The element submatrices are identified as the *dyadic* product of element designations using primary nodes (i, j, q, r are the numbers assigned to nodes)

$$\begin{bmatrix} i \\ j \\ q \\ r \end{bmatrix} \begin{bmatrix} i & j & q & r \end{bmatrix} = \begin{bmatrix} ii & ij & iq & ir \\ ji & jj & jq & jr \\ qi & qj & qq & qr \\ ri & rj & rq & rr \end{bmatrix} \qquad \textbf{Eq. 2.11-4}$$

which for example shown in Figure 2-9 leads to following element submatrices:

$$\begin{bmatrix} P_1 \\ P_5 \\ P_4 \\ P_3 \end{bmatrix} = \begin{bmatrix} K^I_{ii} & K^I_{ij} & K^I_{iq} & K^I_{ir} \\ K^I_{ji} & K^I_{jj} & K^I_{jq} & K^I_{jr} \\ K^I_{qi} & K^I_{qj} & K^I_{qq} & K^I_{qr} \\ K^I_{ri} & K^I_{rj} & K^I_{rq} & K^I_{rr} \end{bmatrix} * \begin{bmatrix} U_1 \\ U_5 \\ U_4 \\ U_3 \end{bmatrix} \qquad \textbf{Eq. 2.11-5}$$

$$\begin{bmatrix} P_3 \\ P_2 \\ P_1 \end{bmatrix} = \begin{bmatrix} K^{II}_{ii} & K^{II}_{ij} & K^{II}_{iq} \\ K^{II}_{ji} & K^{II}_{jj} & K^{II}_{jq} \\ K^{II}_{qi} & K^{II}_{qj} & K^{II}_{qq} \end{bmatrix} * \begin{bmatrix} U_3 \\ U_2 \\ U_1 \end{bmatrix} \qquad \textbf{Eq. 2.11-6}$$

$$\begin{bmatrix} P_2 \\ P_4 \end{bmatrix} = \begin{bmatrix} K^{III}_{ii} & K^{III}_{ij} \\ K^{III}_{ji} & K^{III}_{jj} \end{bmatrix} * \begin{bmatrix} U_2 \\ U_4 \end{bmatrix} \qquad \textbf{Eq. 2.11-7}$$

With this designation, the assembled version of the complete matrix of the configuration shown in Figure 2-9 will be:

$$\begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \end{bmatrix} = \begin{bmatrix} K_{ii}^{I} + K_{qq}^{II} & K_{qj}^{II} & K_{ir}^{I} + K_{qi}^{II} & K_{iq}^{I} & K_{ij}^{I} \\ & K_{jj}^{II} + K_{ii}^{III} & K_{ji}^{II} & K_{ij}^{III} & 0 \\ & & K_{rr}^{I} + K_{ii}^{II} & K_{rq}^{I} & K_{rj}^{I} \\ & & & K_{qq}^{I} + K_{jj}^{III} & K_{qj}^{I} \\ Symm. & & & & K_{jj}^{I} \end{bmatrix} * \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \\ U_5 \end{bmatrix} \qquad \textit{Eq. 2.11-8}$$

Final results of assembling all these elements are system of linear equation which is solved by unknown nodal values (temperatures in Conrad):

$$\|K\| * \{T\} = \{P\} \qquad \textit{Eq. 2.11-9}$$

which is solved by unknown nodal values $\{T\}$, or in equation form:

$$\begin{aligned} &K_{11} * T_1 + K_{12} * T_2 + ... + K_{1,i} * T_i + ... + K_{1,n} * T_n = P_1 \\ &K_{21} * T_1 + K_{22} * T_2 + ... + K_{2,i} * T_i + ... + K_{2,n} * T_n = P_2 \\ &. \\ &K_{i,1} * T_1 + K_{i,2} * T_2 + ... + K_{i,i} * T_i + ... + K_{i,n} * T_n = P_i \\ &. \\ &K_{n,1} * T_1 + K_{n,2} * T_2 + ... + K_{n,i} * T_i + ... + K_{n,n} * T_n = P_n \end{aligned} \qquad \textit{Eq. 2.11-10}$$

2.12. Introduction of Boundary Conditions

At this stage, the essential boundary conditions are introduced. As result of this, the complete set of equations will be reduced or condensed to its final form.

There are several boundary conditions presented in previous chapter. Using weighted-residual method for boundary conditions, following equation is obtained:

$$\int_{\Gamma_h} \omega h_c (T - T_\infty) d\Gamma_h + \int_{\Gamma_R} \omega \sigma \varepsilon (T^4 - T_\infty^4) d\Gamma_R - \int_{\Gamma_q} \omega q_f d\Gamma_q - \int_{\Gamma_r} \omega [\sigma \varepsilon (T^4 - T_\infty^4) - \alpha_i H_i] d\Gamma_r = 0 \qquad \textit{Eq. 2.12-1}$$

2.12.1. Convection Boundary Condition

Convection Boundary Condition is presented by first term of equation (Eq. 2.12-1). There is also linear and nonlinear type of problems in boundary condition presentation.

2.12.1.1. Linear Problem

Using first term of equation (Eq. 2.12-1) and equation (Eq. 2.2-1) for ith algebraic equation $(\omega = \psi_i^e)$ convection boundary condition is:

$$\int_{\Gamma_h} \omega h_c (T - T_\infty) d\Gamma_h = \int_{\Gamma_h} \psi_i^e(x, y) * h_c (\sum_{j=1}^{n} T_j^e \psi_j^e(x, y) - T_\infty) d\Gamma_h =$$

$$= \int_{\tilde{\Gamma}_h} \tilde{\psi}_i^e(\xi) * h_c * \det * \sum_{j=1}^{n} [\tilde{\psi}_j^e(\xi) * T_j^e] d\tilde{\Gamma}_h - \int_{\tilde{\Gamma}_h} \tilde{\psi}_i^e(\xi) * h_c * T_\infty * \det * d\tilde{\Gamma}_h \qquad \textit{Eq. 2.12-2}$$

where $\tilde{\psi}_i^e(\xi)$ and $\tilde{\psi}_j^e(\xi)$ are line functions for master line element (Eq. 2.7-1), det is determinant obtained from equation (Eq. 2.10-16) and $\bar{\Gamma}_h$ is segment in master element. Equation must hold for any weight function ω. Since we need n independent equations to solve for the n unknowns, T_1^e, T_2^e, …, T_n^e, we choose n independent algebraic equations to solve for ω: $\omega = \psi_1^e, \psi_2^e, ..., \psi_n^e$. For each choice of ω we obtain an algebraic relation among $(T_1^e, T_2^e, ..., T_n^e)$. We label the algebraic equation resulting from substitution of ψ_1^e for ω into equation (Eq. 2.12-2) as the first algebraic equation. The i[th] algebraic equation is obtained by substituting $\omega = \psi_i^e$ into equation (Eq. 2.12-2):

$$\sum_{j=1}^{n} A_{ij}^e T_j^e - B_i^e = 0 \qquad \textit{Eq. 2.12-3}$$

where:

$$A_{ij}^e = \int_{\tilde{\Gamma}_h} \tilde{\psi}_i^e(\xi) * \tilde{\psi}_j^e(\xi) * h_c * \det * d\tilde{\Gamma}_h \qquad \textit{Eq. 2.12-4}$$

$$B_i^e = \int_{\tilde{\Gamma}_h} \tilde{\psi}_i^e(\xi) * h_c * T_\infty * \det * d\tilde{\Gamma}_h \qquad \textit{Eq. 2.12-5}$$

these equations must be assembled into the equation (Eq. 2.11-9) to obtain system of linear equation with introduced convection boundary conditions. In matrix notation, equation (Eq. 2.12-3) becomes:

$$[A_{ij}]\{T_j\} = \{B_i\} \qquad \textit{Eq. 2.12-6}$$

Therefore, equation (Eq. 2.12-6) must be assembled into equation (Eq. 2.11-9).

2.12.1.2. Nonlinear Problem

Equation for convection boundary condition for nonlinear problems takes following form:

$$\int_{\Gamma_h} \omega * h_c (T + \Delta T) * [(T + \Delta T) - T_\infty] d\Gamma_h = 0 \qquad \textit{Eq. 2.12-7}$$

Convection boundary condition for nonlinear problems uses following approximations:

$$\frac{\partial h_c}{\partial t} = \frac{h_c(T + \Delta T) - h_c(T)}{\Delta T} \Rightarrow h_c(T + \Delta T) = h_c(T) + \Delta T \frac{\partial h_c}{\partial t} \qquad \textit{Eq. 2.12-8}$$

and

$$T(x,y)+\Delta T(x,y)\approx T^e(x,y)+\Delta T^e(x,y)=\sum_{j=1}^{m}T_j^e\psi_j^e(x,y)+\sum_{j=1}^{m}\Delta T_j^e\psi_j^e(x,y) \quad \textbf{Eq. 2.12-9}$$

Using these two approximations and $\omega=\psi_i^e$, equation (Eq. 2.12-7) becomes:

$$\int_{\Gamma_h}\psi_i^e(x,y)*[h_c(T)+(\sum_{j=1}^{m}\Delta T_j^e\psi_j^e(x,y))\frac{\partial h_c}{\partial t}]*[(T_{gn}+\Delta T_{gn})-T_\infty]d\Gamma_h \quad \textbf{Eq. 2.12-10}$$

where T_{gn} and ΔT_{gn} are temperatures from previous iteration.

Linear part of equation (Eq. 2.12-10) obtain same result as equations (Eq. 2.12-4), (Eq. 2.12-5) and (Eq. 2.12-6). Note also that for linear part of equation (Eq. 2.12-7) is not used $(T_{gn}+\Delta T_{gn})$ but equation (Eq. 2.12-9) .Nonlinear parts of equation (Eq. 2.12-7) are:

$$C_{ij}=\int_{\tilde{\Gamma}_h}\tilde{\psi}_i^e(\xi)*\tilde{\psi}_j^e(\xi)*\frac{\partial h_c}{\partial t}*\det*T_{gn}*d\tilde{\Gamma}_h \quad \textbf{Eq. 2.12-11}$$

$$D_{ij}=\int_{\tilde{\Gamma}_h}\tilde{\psi}_i^e(\xi)*\tilde{\psi}_j^e(\xi)*\frac{\partial h_c}{\partial t}*\det*\Delta T_{gn}*d\tilde{\Gamma}_h \quad \textbf{Eq. 2.12-12}$$

$$E_{ij}=\int_{\tilde{\Gamma}_h}\tilde{\psi}_i^e(\xi)*\tilde{\psi}_j^e(\xi)*\frac{\partial h_c}{\partial t}*\det*T_\infty*d\tilde{\Gamma}_h \quad \textbf{Eq. 2.12-13}$$

Convection boundary condition for nonlinear problem in matrix form:

$$[C_{ij}]\{\Delta T_j\}+[D_{ij}]\{\Delta T_j\}-[E_{ij}]\{\Delta T_j\}=\{B_i\}-[A_{ij}]\{T_j\}\ \Rightarrow$$
$$\Rightarrow\quad\{[C_{ij}]+[D_{ij}]-[E_{ij}]\}*\{\Delta T_j\}=\{B_i\}-[A_{ij}]\{T_j\} \quad \textbf{Eq. 2.12-14}$$

2.12.2. Flux Boundary Condition

Flux Boundary Condition is presented by third term of equation (Eq. 2.12-1). There is also linear and nonlinear presentation of this boundary condition.

2.12.2.1. Linear Problem

Using third term of equation (Eq. 2.12-1) and equation (Eq. 2.2-1) for i[th] algebraic equation $(\omega=\psi_i^e)$ flux boundary condition is:

$$\int_{\Gamma_q}\omega q_f d\Gamma_q=\int_{\Gamma_q}\psi_i^e(x,y)*q_f*d\Gamma_q=\int_{\tilde{\Gamma}_q}\tilde{\psi}_i^e(\xi)*q_f*d\tilde{\Gamma}_q \quad \textbf{Eq. 2.12-15}$$

$$Q_i=\int_{\tilde{\Gamma}_q}\tilde{\psi}_i^e(\xi)*q_f*\det*d\tilde{\Gamma}_q \quad \textbf{Eq. 2.12-16}$$

this matrix must be assembled into the (Eq. 2.11-9).

2.12.2.2. Nonlinear Problem

Nonlinear part uses following approximations:

$$q_f(T+\Delta T) = q_f(T) + \Delta T \frac{\partial q_f}{\partial t} \qquad \textit{Eq. 2.12-17}$$

and

$$\Delta T(x,y) \approx \Delta T^e(x,y) = \sum_{j=1}^{n} \Delta T_j^e \psi_j^e(x,y) \qquad \textit{Eq. 2.12-18}$$

Therefore, flux boundary condition for i-th equation is:

$$\int_{\Gamma_q} \omega * q_f(T+\Delta T) * d\Gamma_q = \int_{\Gamma_q} \psi_i^e(x,y) * [q_f(T) + \frac{\partial q_f}{\partial t} \sum_{j=1}^{n} \Delta T_j^e \psi_j^e(x,y)] * d\Gamma_q =$$
$$= \int_{\tilde{\Gamma}_q} \tilde{\psi}_i^e(\xi) * [q_f(T) + \frac{\partial q_f}{\partial t} \sum_{j=1}^{n} \Delta T_j^e \tilde{\psi}_j^e(\xi)] * \det * d\tilde{\Gamma}_q \qquad \textit{Eq. 2.12-19}$$

or

$$\sum_{j=1}^{n} F_{ij}^e \Delta T_j + Q_i = 0 \qquad \textit{Eq. 2.12-20}$$

or in matrix from

$$[F_{ij}^e]\{\Delta T_j\} - \{Q_i\} = 0 \qquad \textit{Eq. 2.12-21}$$

where

$$F_{ij}^e = \int_{\tilde{\Gamma}_q} \tilde{\psi}_i^e(\xi) * \tilde{\psi}_j^e(\xi) * \frac{\partial q_f}{\partial t} * \det * d\tilde{\Gamma}_q \qquad \textit{Eq. 2.12-22}$$

and Q_i is given by equation (Eq. 2.12-16).

2.12.3. Radiation Boundary Condition

Radiation (black body) boundary condition is given by second term of equation (Eq. 2.12-1):

$$\int_{\Gamma_R} \omega\sigma\varepsilon(T^4 - T_\infty^4) d\Gamma_R = 0 \qquad \textit{Eq. 2.12-23}$$

Because radiation makes problem nonlinear, this equation will be calculated only for nonlinear case. Before transformation equation (Eq. 2.12-23) must be linearized about temperature from previous iteration:

$$(T^4 - T_\infty^4) \cong (T_{PREV} + T_\infty)(T_{PREV}^2 + T_\infty^2)(T - T_\infty) \qquad \textit{Eq. 2.12-24}$$

where T_{PREV} is temperature from previous iteration. Substituting equation (Eq. 2.12-24) into the (Eq. 2.12-23):

$$\int_{\Gamma_R} \omega\sigma\varepsilon(T_{PREV} + T_\infty)(T_{PREV}^2 + T_\infty^2)(T - T_\infty) d\Gamma_R = \int_{\Gamma_R} \omega * h_r * (T - T_\infty) * d\Gamma_R \qquad \textit{Eq. 2.12-25}$$

where

$$h_r = \sigma\varepsilon(T_{PREV} + T_\infty)(T_{PREV}^2 + T_\infty^2) \qquad \textbf{Eq. 2.12-26}$$

is linearized about T_{PREV} .

Equation (Eq. 2.12-25) is same as equation (Eq. 2.12-2) but only for nonlinear case (because radiation makes problem nonlinear). Therefore, element matrices and equations are same as for convection boundary condition for nonlinear case:

$$\int_{\Gamma r} \omega h_r (T - T_\infty) d\Gamma_r = \int_{\Gamma_h} \psi_i^e(x,y) * h_r (\sum_{j=1}^{n} T_j^e \psi_j^e(x,y) - T_\infty) d\Gamma_r =$$
$$= \int_{\tilde{\Gamma}_r} \tilde{\psi}_i^e(\xi) * h_r * \det * \sum_{j=1}^{n} [\tilde{\psi}_j^e(\xi) * T_j^e] d\tilde{\Gamma}_r - \int_{\tilde{\Gamma}_r} \tilde{\psi}_i^e(\xi) * h_r * T_\infty * \det * d\tilde{\Gamma}_r \qquad \textbf{Eq. 2.12-27}$$

which leads to same matrix equations as convection boundary condition for nonlinear case:

$$A_{ij}^e = \int_{\tilde{\Gamma}_r} \tilde{\psi}_i^e(\xi) * \tilde{\psi}_j^e(\xi) * h_r * \det * d\tilde{\Gamma}_r \qquad \textbf{Eq. 2.12-28}$$

$$B_i^e = \int_{\tilde{\Gamma}_r} \tilde{\psi}_i^e(\xi) * h_r * T_\infty * \det * d\tilde{\Gamma}_r \qquad \textbf{Eq. 2.12-29}$$

$$C_{ij} = \int_{\tilde{\Gamma}_r} \tilde{\psi}_i^e(\xi) * \tilde{\psi}_j^e(\xi) * \frac{\partial h_r}{\partial t} * \det * T_{gn} * d\tilde{\Gamma}_r \qquad \textbf{Eq. 2.12-30}$$

$$D_{ij} = \int_{\tilde{\Gamma}_r} \tilde{\psi}_i^e(\xi) * \tilde{\psi}_j^e(\xi) * \frac{\partial h_r}{\partial t} * \det * \Delta T_{gn} * d\tilde{\Gamma}_r \qquad \textbf{Eq. 2.12-31}$$

$$E_{ij} = \int_{\tilde{\Gamma}_r} \tilde{\psi}_i^e(\xi) * \tilde{\psi}_j^e(\xi) * \frac{\partial h_r}{\partial t} * \det * T_\infty * d\tilde{\Gamma}_r \qquad \textbf{Eq. 2.12-32}$$

Or equation (Eq. 2.12-27) in matrix form:

$$[C_{ij}]\{\Delta T_j\} + [D_{ij}]\{\Delta T_j\} - [E_{ij}]\{\Delta T_j\} = \{B_i\} - [A_{ij}]\{T_j\} \quad \Rightarrow$$
$$\Rightarrow \qquad \{[C_{ij}] + [D_{ij}] - [E_{ij}]\} * \{\Delta T_j\} = \{B_i\} - [A_{ij}]\{T_j\} \qquad \textbf{Eq. 2.12-33}$$

2.12.4. Enclosure Radiation Boundary Condition

Enclosure radiation (gray body) boundary condition is given by fourth term of equation (Eq. 2.12-1):

$$\int_{\Gamma_r} \omega[\sigma\varepsilon_i T_i^4 - \alpha_i H_i] d\Gamma_r \cong \int_{\Gamma_r} \omega[\sigma\varepsilon_i (4 * T_{PREV}^3 * T_i - 3 * T_{PREV}^4) - \alpha_i H_i] d\Gamma_r \qquad \textbf{Eq. 2.12-34}$$

where index “i” mean i-th surface in enclosure model and H_i is calculated by equations (Eq. 1.3-45) and (Eq. 1.3-46). From equation (Eq. 1.3-46) is obtained following system of equations:

$$[(\frac{\varepsilon_1-1}{\varepsilon_1})F_{11}-\frac{1}{\varepsilon_1}]B_1+...+\frac{\varepsilon_1-1}{\varepsilon_1}F_{1i}B_i+...+\frac{\varepsilon_1-1}{\varepsilon_1}F_{1n}B_n=\sigma T_{1PREV}^4$$

.

$$\frac{\varepsilon_i-1}{\varepsilon_i}F_{i1}B_1+...+[(\frac{\varepsilon_i-1}{\varepsilon_i})F_{ii}-\frac{1}{\varepsilon_i}]B_i+...+\frac{\varepsilon_i-1}{\varepsilon_i}F_{in}B_n=\sigma T_{iPREV}^4 \qquad \textbf{Eq. 2.12-35}$$

.

$$\frac{\varepsilon_n-1}{\varepsilon_n}F_{n1}B_1+...+\frac{\varepsilon_n-1}{\varepsilon_n}F_{ni}B_i+...+[(\frac{\varepsilon_n-1}{\varepsilon_n})F_{nn}-\frac{1}{\varepsilon_i}]B_n=\sigma T_{nPREV}^4$$

where T_{iPREV} is temperature of i-th surface from previous iteration. Or in matrix notation

$$[AEF_{ij}]\{B_j\}=\sigma\{T_{iPREV}^4\} \;\Rightarrow\; \{B_j\}=[AEF_{ij}]^{-1}*\sigma*\{T_{iPREV}^4\} \qquad \textbf{Eq. 2.12-36}$$

equation (Eq. 2.12-36) gives solution of radiosity matrix $\{B_j\}$. For nonlinear iteration using for calculation of enclosure radiation boundary condition following matrix equation is used:

$$\{B_j\}=[AEF_{\substack{ij\\i\neq j}}]^{-1}*\sigma*\{T_{\substack{iPREV\\i\neq j}}^4\}+\{AEF^{-1}{}_{ii}\}*\sigma*\{T_j^4\}=$$
$$=\{K_j\}+\{AEF^{-1}{}_{ii}\}*\sigma*\{T_j^4\} \qquad \textbf{Eq. 2.12-37}$$

where $\{AEF^{-1}{}_{ii}\}$ represent vector of elements tacked from diagonal of inverse matrix $[AEF_{ij}]$ and $\{T_j^4\}$ is vector of unknown temperatures. Therefore, radiosity for i-th surface is:

$$B_i=\sum_{j=1}^{n}(AEF^{-1}{}_{ij}*\sigma*T_{jPREV}^4)+AEF^{-1}{}_{ii}*\sigma*T_i^4=$$
$$=\sum_{\substack{j=1\\j\neq i}}^{n}(psi_{ij}*\sigma*T_{jPREV}^4)+psi_{ii}*\sigma*T_i^4=K_i+psi_{ii}*\sigma*T_i^4 \qquad \textbf{Eq. 2.12-38}$$

replacing this equation into the equation (Eq. 1.3-45), following equation is obtained:

$$H_i=\frac{1}{1-\varepsilon_i}(K_i-(\varepsilon_i-psi_{ii})*\sigma*T_i^4) \qquad \textbf{Eq. 2.12-39}$$

and substituting this into (Eq. 2.12-34):

$$\int_{\Gamma_{er}}\omega[\sigma\varepsilon_iT_i^4-\alpha_iH_i]d\Gamma_{er}=$$
$$=\int_{\Gamma_{er}}\omega[\frac{\varepsilon_i}{1-\varepsilon_i}(1-psi_{ii})T_i^4-\frac{\varepsilon_i}{1-\varepsilon_i}K_i)]d\Gamma_{er} \qquad \textbf{Eq. 2.12-40}$$

where $\alpha_i=\varepsilon_i$ for gray and isothermal radiating surfaces. After Taylor approximation about temperature T_{iPREV} equation (Eq. 2.12-40) becomes:

$$\int_{\Gamma_{er}} \omega[\sigma\varepsilon_i T_i^4 - \alpha_i H_i]d\Gamma_{er} =$$

$$= \int_{\Gamma_{er}} \omega[\frac{\varepsilon_i}{1-\varepsilon_i}(1-psi_{ii})\sigma * 4T_{iPREV}^3 * T - \frac{\varepsilon_i}{1-\varepsilon_i}((1-psi_{ii})\sigma * 3T_{iPREV}^4 + K_i)]d\Gamma_{er} = \quad \textbf{Eq. 2.12-41}$$

$$= \int_{\Gamma_{er}} \omega[h_{er} * T - r_h]d\Gamma_{er}$$

where

$$h_{er} = \frac{\varepsilon_i}{1-\varepsilon_i}(1-psi_{ii})\sigma * 4T_{iPREV}^3 \quad \textbf{Eq. 2.12-42}$$

and

$$r_h = \frac{\varepsilon_i}{1-\varepsilon_i}((1-psi_{ii})\sigma * 3T_{iPREV}^4 + K_i) \quad \textbf{Eq. 2.12-43}$$

In order to obtain element equations and matrices, equation (Eq. 2.12-41) is used with following approximations:

$$h_{er}(T+\Delta T) = h_{er}(T) + \Delta T \frac{\partial h_{er}}{\partial T} \quad \textbf{Eq. 2.12-44}$$

and equations (Eq. 2.12-9) and (Eq. 2.7-1). Consider first term of equation (Eq. 2.12-41):

$$\int_{\Gamma_{er}} \omega * h_{er} * T * d\Gamma_{er} \quad \textbf{Eq. 2.12-45}$$

to obtain element equation, equation (Eq. 2.12-45) is considered in two different forms:

$$\int_{\tilde{\Gamma}_{er}} \tilde{\psi}_i^e(\xi) * (h_{er}(T) + \Delta T \frac{\partial h_{er}}{\partial T}) * (T+\Delta T) * \det * d\tilde{\Gamma}_{er} =$$

$$= \int_{\tilde{\Gamma}_{er}} \tilde{\psi}_i^e(\xi) * (h_{er}(T) + \frac{\partial h_{er}}{\partial T}\sum_{j=1}^{n} \Delta T_j^e \tilde{\psi}_j^e(\xi)) * (\sum_{j=1}^{n} T_j^e \tilde{\psi}_j^e(\xi) + \sum_{j=1}^{n} \Delta T_j^e \tilde{\psi}_j^e(\xi)) * \det * d\tilde{\Gamma}_{er} \quad \textbf{Eq. 2.12-46}$$

and

$$\int_{\tilde{\Gamma}_{er}} \tilde{\psi}_i^e(\xi) * (h_{er}(T) + \Delta T \frac{\partial h_{er}}{\partial T}) * (T+\Delta T) * \det * d\tilde{\Gamma}_{er} =$$

$$= \int_{\tilde{\Gamma}_{er}} \tilde{\psi}_i^e(\xi) * (h_{er}(T) + \frac{\partial h_{er}}{\partial T}\sum_{j=1}^{n} \Delta T_j^e \tilde{\psi}_j^e(\xi)) * (T_{ign} + \Delta T_{ign}) * \det * d\Gamma_{er} \quad \textbf{Eq. 2.12-47}$$

or

$$\sum_{j=1}^{n} (G_{ij} + H_{ij} + I_{ij})\Delta T_j^e + L_i = 0 \quad \textbf{Eq. 2.12-48}$$

or in matrix form:

$$[G_{ij} + H_{ij} + I_{ij}]\{\Delta T\} + \{L_i\} = 0 \qquad \textit{Eq. 2.12-49}$$

where

$$G_{ij} = \int_{\tilde{\Gamma}_{er}} \tilde{\psi}_i^e(\xi) * h_{er} * \tilde{\psi}_j^e(\xi) * \det * d\tilde{\Gamma}_{er} \qquad \textit{Eq. 2.12-50}$$

$$H_{ij} = \int_{\tilde{\Gamma}_{er}} \tilde{\psi}_j^e(\xi) * \frac{\partial h_{er}}{\partial T} * \tilde{\psi}_j^e(\xi) * T_{ign} * \det * d\tilde{\Gamma}_{er} \qquad \textit{Eq. 2.12-51}$$

$$I_{ij} = \int_{\tilde{\Gamma}_{er}} \tilde{\psi}_j^e(\xi) * \frac{\partial h_{er}}{\partial T} * \tilde{\psi}_j^e(\xi) * \Delta T_{ign} * \det * d\tilde{\Gamma}_{er} \qquad \textit{Eq. 2.12-52}$$

where T_{ign} and ΔT_{ign} are mean temperature (and temperature difference) on i-th segment from previous iteration, and:

$$L_i = [G_{ij}] * \{T_j^e\} \qquad \textit{Eq. 2.12-53}$$

where $\{T_j^e\}$ are matrix of node temperatures for i-th segment from previous iteration. Now consider second term of equation (Eq. 2.12-41):

$$\int_{\Gamma_{er}} \omega * r_h * d\Gamma_{er} = \int_{\tilde{\Gamma}_{er}} \tilde{\psi}_i^e(\xi) * (r_h(T) + \Delta T \frac{\partial r_h}{\partial T}) * \det * d\tilde{\Gamma}_{er} =$$
$$= \int_{\tilde{\Gamma}_{er}} \tilde{\psi}_i^e(\xi) * (r_h(T) + \frac{\partial r_h}{\partial T} \sum_{j=1}^{n} \tilde{\psi}_j^e(\xi) \Delta T_j^e) * \det * d\tilde{\Gamma}_{er} \qquad \textit{Eq. 2.12-54}$$

or in matrix form

$$[M_{ij}]\{\Delta T_j^e\} + \{N_i\} = 0 \qquad \textit{Eq. 2.12-55}$$

where

$$M_{ij} = \int_{\tilde{\Gamma}_{er}} \tilde{\psi}_i^e(\xi) * \tilde{\psi}_j^e(\xi) * \frac{\partial r_h}{\partial T} * \det * d\tilde{\Gamma}_{er} \qquad \textit{Eq. 2.12-56}$$

$$N_i = \int_{\tilde{\Gamma}_{er}} \tilde{\psi}_i^e(\xi) * r_h * \det * d\tilde{\Gamma}_{er} \qquad \textit{Eq. 2.12-57}$$

2.13. Solution of the Final Set of Simultaneous Equations

Until this step, we have made no reference to weather the problem is linear or nonlinear, or weather it is an eigenvalue problem or not. Regardless of the nature of the problem, the finite-element methods eventually yield the solution of a set of simultaneous differential equations. The solution procedure for simultaneous equations can in general, is categorized into the three parts: (1) *direct*, (2) *iterative*, and (3) *stochastic*.

2.13.1. Linear Method

Conrad improves two methods of solution which depends of problem type. For *linear* type of problem Conrad uses *direct* method which means that solution is obtained after solution of global matrices.

2.13.2. Nonlinear Method

For *nonlinear* type of problem Conrad uses *iterative* method to obtain final solution (this mean that solution is obtained after couple of iterations). To obtain achieved convergence Conrad uses following equations:

$$enorm1 = \sqrt{T_1^2 + T_2^2 + T_3^2 + ... + T_n^2} \qquad \textbf{Eq. 2.13-1}$$

where $T_1, T_2, T_3, ..., T_n$ are temperatures at nodes in current iteration which are obtained as:

$$T_i^{(n)} = T_i^{(n-1)} + relax * \Delta T_i^{(n)} \qquad 0 < relax \le 1 \qquad \textbf{Eq. 2.13-2}$$

where $T_i^{(n-1)}$ is temperature at i[th] node form previous iteration and $\Delta T_i^{(n)}$ is solution from current iteration.

I) Convergence Criteria

If signed *enorm2* as Eq. 2.13-1 from previous iteration, then solution is achieved when:

$$edif1 = \frac{|enorm1 - enorm2|}{enorm1} < tolerance \qquad \textbf{Eq. 2.13-3}$$

Eq. 2.13-3 is known as *convergence criteria* and *edif1* is achieved convergence.

II) Divergence Criteria

Sign *edif2* as achieved convergence in previous iteration. Solution diverged (for fixed *relax*) if following condition is satisfied **ten times** for fixed value of relax parameter:

$$edif2 < edif1 \qquad \textbf{Eq. 2.13-4}$$

2.14. Interpretation of the Results

The previous step resulted in the approximate values of the state variable at discrete points (nodes) of the domain. Normally these values are interpreted and used for calculations of other physical entities, such as flux, either thought the domain or in certain regions of it.

This is decision-making step and is probably the most important step in the entire process. Two important questions must be answered at this point: *How good are the results?* and *What should be done with them?* The first requires the estimation of *error* bounds, and the second involves the physical nature of the problem.

2.15. Numerical Integration

Numerical integration plays an important role in finite element methods. For instance, evaluation of element matrices requires integration of certain functions over the element domains. In order to facilitate these integrations and special coordinate systems are normally chosen.

Integrals defined over a rectangular master element $\tilde{\Omega}_m$ can be numerically evaluated using the Gauss-Legendre quadrature formulas:

$$\int_{\tilde{\Omega}_m} F(\xi,\eta)d\xi d\eta = \int_{-1}^{1}\int_{-1}^{1} F(\xi,\eta)d\xi d\eta \approx \sum_{I=1}^{M}\sum_{J=1}^{N} F(\xi_I,\eta_J)W_I W_J \qquad \textbf{Eq. 2.15-1}$$

where M and N denote the number of Gauss quadrature points, (ξ_I,η_J) denote the Gauss points coordinates, and W_I and W_J denote the corresponding Gauss weights as shown in Table 2.1.

For two-point formula gauss points are shown in Figure 2-10:

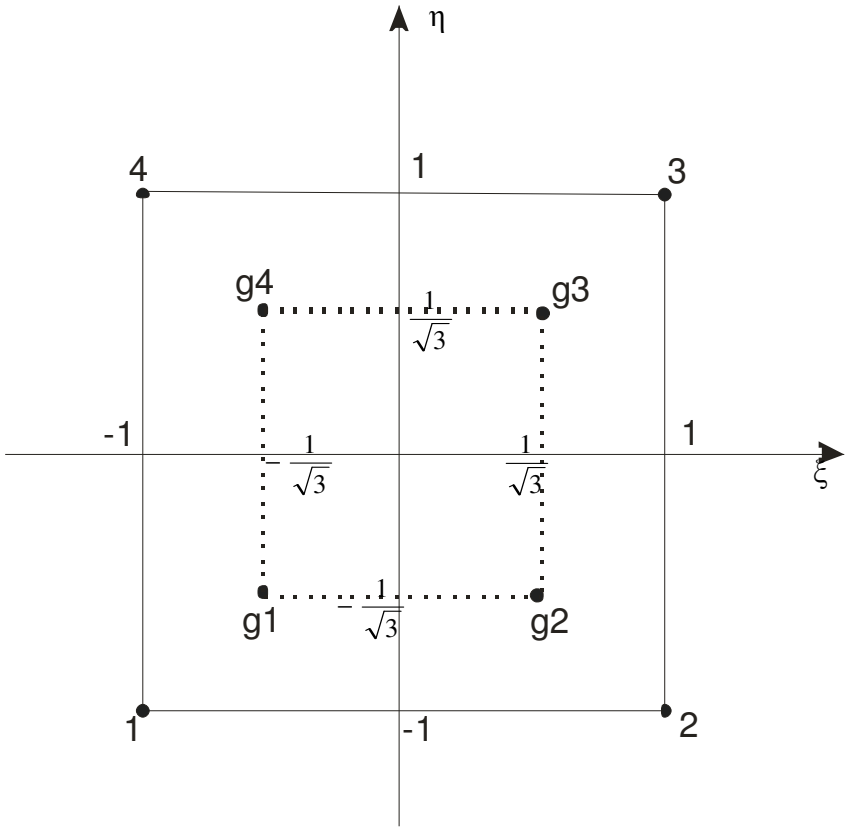


Figure 2-10: Gauss Points for Two-Point Numerical Integration

Table 2.15-1: Quadrature Weights and Points for Rectangular Elements

Points ξ_i	r	**Weights W_i**

0.0000000000	**One-point formula**	2.0000000000
±0.5773502692	**Two-point formula**	1.0000000000
0.0000000000	**Three-point formula**	0.8888888889
±0.7745966692		0.5555555555
±0.3399810435	**Four-point formula**	0.6521451548
±0.8611363116		0.3478548451
0.0000000000	**Five-point formula**	0.5688888889
±0.5384693101		0.4786286705
±0.9061798459		0.2369268850
±0.2386191861	**Six-point formula**	0.4679139346
±0.6612093865		0.3607615730
±0.9324695142		0.1713244924

3. Additional Algorithms and Descriptions

3.1. Bandwidth Minimization

A key numerical problem which arises throughout finite-element analysis (whether linear or nonlinear, static or dynamic) is that of the solution of large sets of linear algebraic equations such as, in matrix form,

$$\|A\| * \{x\} = \{b\} \qquad \textit{Eq. 3.1-1}$$

where the vector $\{b\}$ and the square matrix $\|A\|$ are known, and the unknown vector $\{x\}$ is sought.

In finite element applications, $\|A\|$ contains mostly zeros and efficiency in equation solving is obtained by avoiding arithmetic operations (multiplications and additions) on matrix terms that are known in advance to be zero. The computer execution time for most equation solvers and triangular factorization routines is proportional to the order *N* of the matrix. It is possible to choose an ordering for sparse matrices so that nonzeros are located to allow subsequent matrix operations such as equation solving or eigenvalue extraction. In general, a banded matrix has all its nonzero entries clustered about the main diagonal (Figure 3-1).

Conrad uses the Gibbs-Poole-Stockmeyer algorithm (see [2]) to determine a nodal numbering scheme which results in minimal bandwidth/profile.

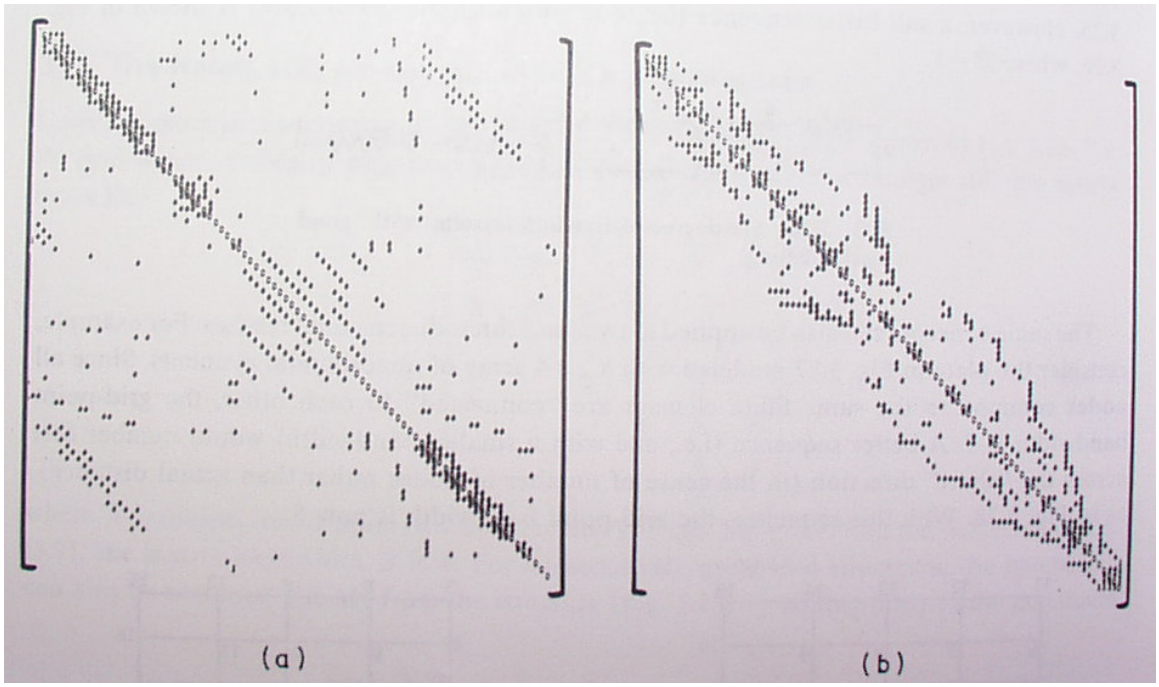


Figure 3-1: Location of nonzero terms in a stiffness matrix: (a) before and (b) after reordering (see [2]).

Sign transformation from original to reorder numbering system [Figure 3-1 from *(a)* to *(b)*] as BWM , and inverse transformation or from reorder to original numbering system as BWM^{-1}.

3.2. Gravity Arrow Algorithm and Frame Cavity Transformations for ISO15099 Calculations

3.2.1. Introduction

Gravity Arrow Algorithm is used to determine heat flow direction in frame cavities according to gravity arrow. It is needed to perform calculations shown in [1]. Note that in [1] frame cavity calculations are in 2D space and in THERM5 frame cavity are presented by 3 dimensions (or in 3D space). Therefore, purpose of this algorithm is to transform frame cavity from 3D presentation into the 2D presentation according to heat flow direction and Gravity Arrow direction.

3.2.2. Equivalent Gravity Arrow

Gravity Arrow is vector can point in any direction in 3D space and according to algorithm described bellow it will be transformed to *Equivalent Gravity Arrow* which can point only in the ”x”, “y” or “z”-axes direction.

Gravity Arrow is presented in 3D (Figure 3-2) space by coordinates g_x, g_y and g_z respectively.

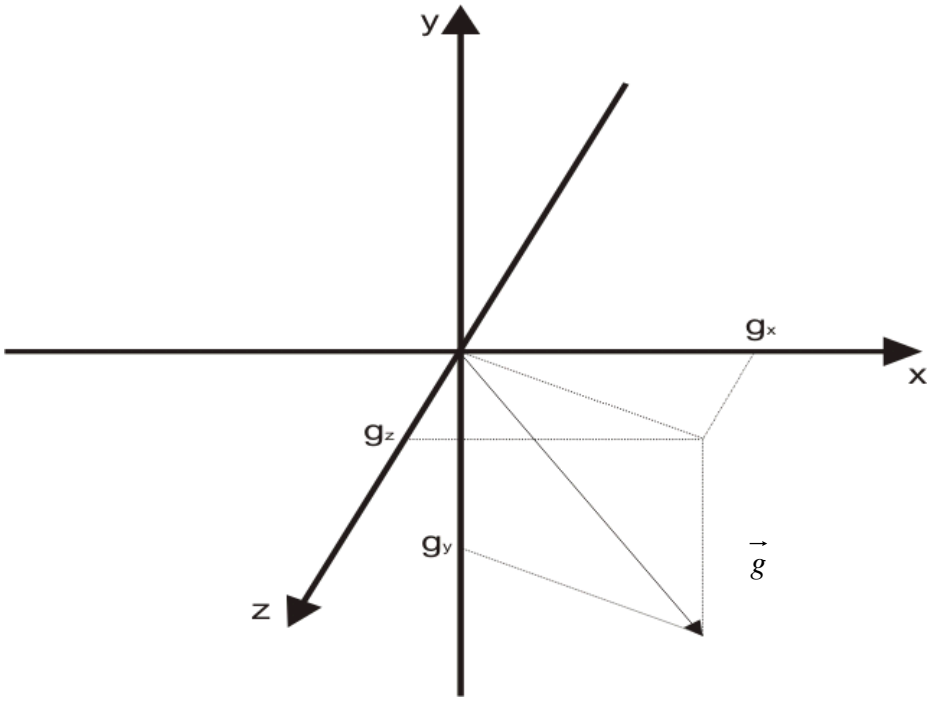


Figure 3-2: Gravity Arrow in 3D Coordinate System

Therefore, Gravity Arrow can be expressed by following equation:

$$\vec{g} = g_x \vec{x} + g_y \vec{y} + g_z \vec{z} \qquad \textit{Eq. 3.2-1}$$

where coordinates g_x, g_y, g_z must satisfy following equation:

$$\sqrt{(g_x)^2 + (g_y)^2 + (g_z)^2} = 1 \qquad \textit{Eq. 3.2-2}$$

In order to determine *equivalent Gravity Arrow* (this is gravity arrow which pointing in one of the axes direction), 3D space is divided into the six equivalent spaces using six surfaces which are shown in Figure 3-3, Figure 3-4 and Figure 3-5.

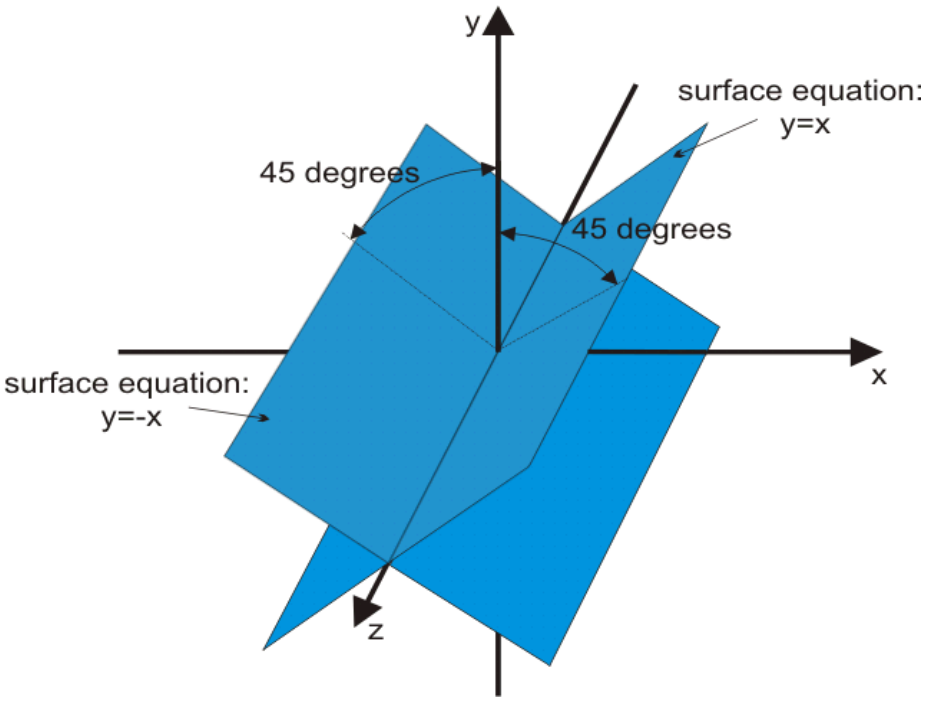


Figure 3-3: Surfaces Parallel With z-axis

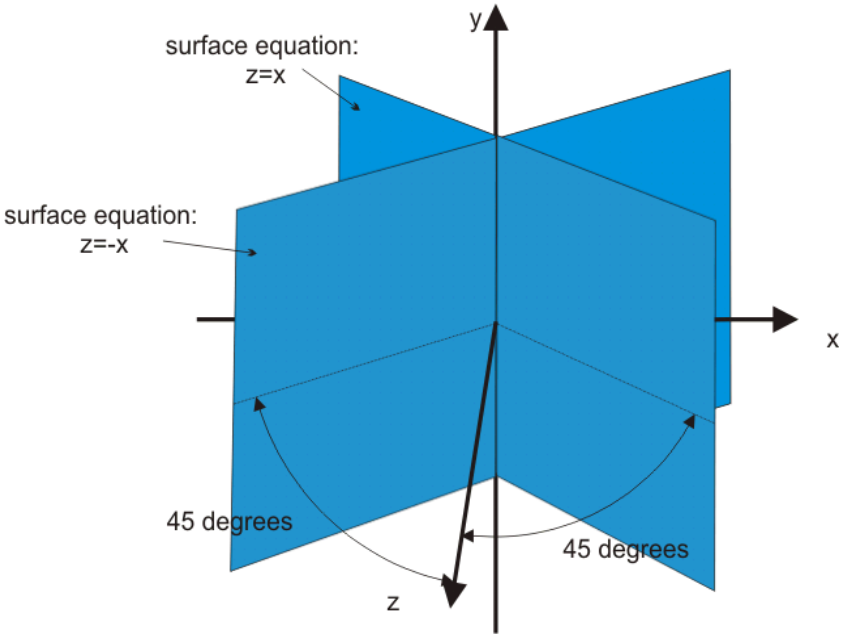


Figure 3-4: Surfaces Parallel With y-axis

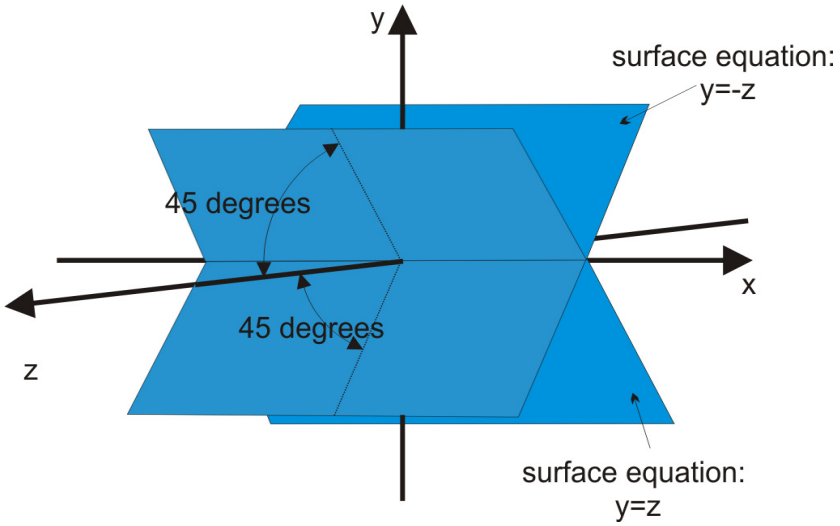


Figure 3-5: Surfaces Parallel With x-axe

Surfaces on Figure 2 and Figure 3 make pyramids in direction to the x-axis (Figure 3-6), one in positive direction of x-axes (yellow) and the other in negative (blue).

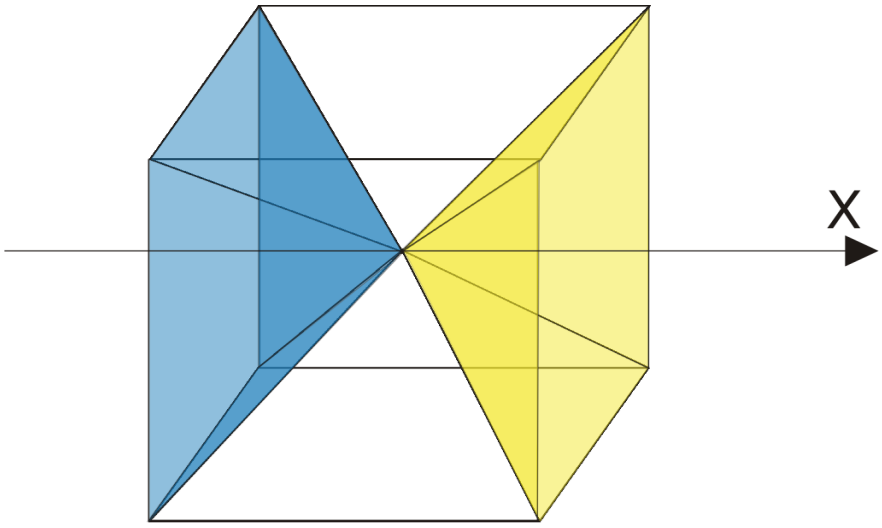


Figure 3-6: x-axe Pyramids

Note that these pyramids have top angle equal with 90° (Figure 3-7 and Figure 3-8).

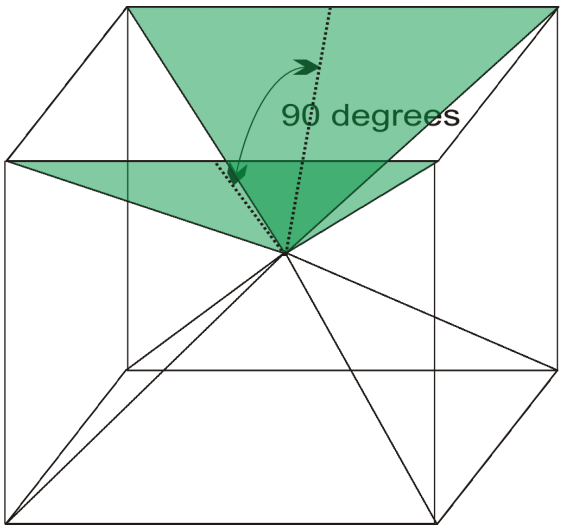


Figure 3-7: Pyramid Angle

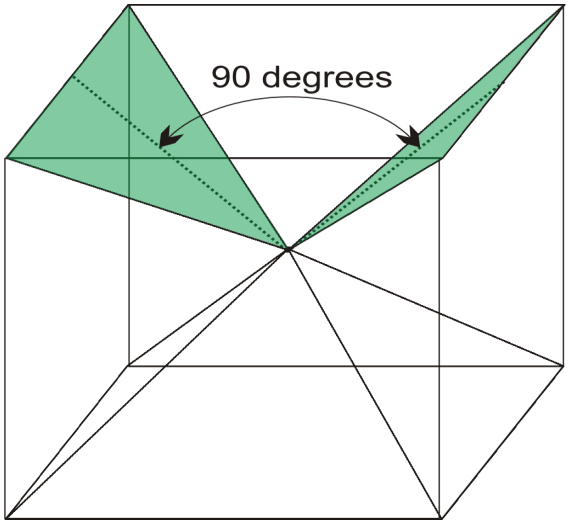


Figure 3-8: Pyramid Angle

Therefore, these four surfaces make two pyramids in x-axes direction in order to determine any gravity vector which belongs to pyramids space. According to surfaces equation (see Figure 3-3, Figure 3-4 and Figure 3-5) gravity arrow belongs to positive x-axes pyramid (yellow color in Figure 3-6) when following equations are satisfied:

$$y < x;\ \ y > -x;\ \ z < x\ \ and\ \ z > -x \qquad \textbf{Eq. 3.2-3}$$

when replacing gravity arrow orts

$$g_y < g_x;\ g_y > -g_x;\ g_z < g_x\ \ and\ \ g_z > -g_x \qquad \textbf{Eq. 3.2-4}$$

If condition (Eq. 3.2-4) is satisfied, gravity arrow is replaced with it equivalent which pointing in positive direction of x-axis and which orts are:

$$g_x = 1;\ \ g_y = 0;\ \ g_z = 0 \qquad \textbf{Eq. 3.2-5}$$

Same explanation is for negative x-axis pyramid (blue color in Figure 3-6) but for different equations.

To recover all directions in 3D space, there are also pyramids which belongs to “y” and “z” axis (Figure 3-9 and Figure 3-10)

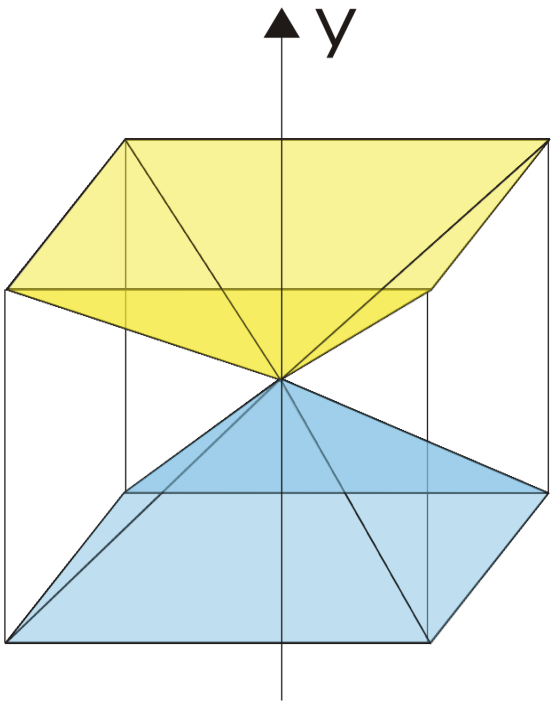


Figure 3-9: y-axis Pyramids

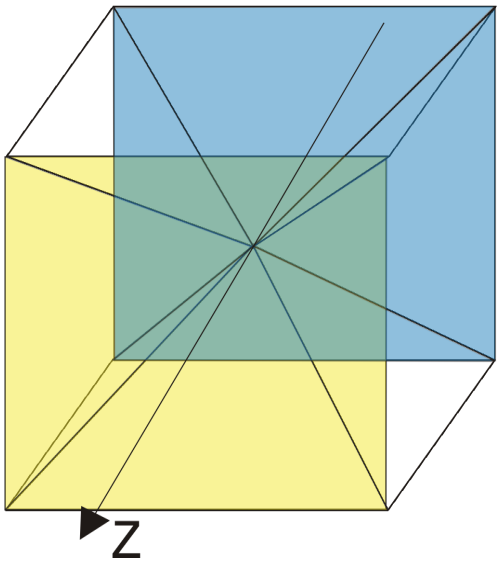


Figure 3-10: z-axis Pyramids

3.2.3. Summary of Pyramid Equations

3.2.3.1. X-Axis

I) Positive Direction

$$g_x > g_y;\ g_x > -g_y;\ g_x > g_z;\ g_x > -g_z \qquad \textit{Eq. 3.2-6}$$

II) Negative Direction

$$g_x < g_y;\ g_x < -g_y;\ g_x < g_z;\ g_x < -g_z \qquad \textit{Eq. 3.2-7}$$

3.2.3.2. Y-Axis

I) Positive Direction

$$g_y > g_x;\ g_y > -g_x;\ g_y > g_z;\ g_y > -g_z \qquad \textit{Eq. 3.2-8}$$

II) Negative Direction

$$g_y < g_x;\ g_y < -g_x;\ g_y < g_z;\ g_y < -g_z \qquad \textit{Eq. 3.2-9}$$

3.2.3.3. Z-Axis

I) Positive Direction

$$g_z > g_x;\ g_z > -g_x;\ g_z > g_y;\ g_z > -g_y \qquad \textit{Eq. 3.2-10}$$

II) Negative Direction

$$g_z < g_x;\ g_z < -g_x;\ g_z < g_y;\ g_z < -g_y \qquad \textit{Eq. 3.2-11}$$

3.2.4. Frame Cavity Presentation and Heat Flow Direction

Frame Cavity in THERM5 can be drawn only in 2D and third dimension is given by variable "jambheight" (see

Figure 3-16). Irregularly shaped Frame Cavities are rectangularized according to procedure given in [1]. Rectangularized Frame Cavity in THERM5 is presented by Figure 3-11:

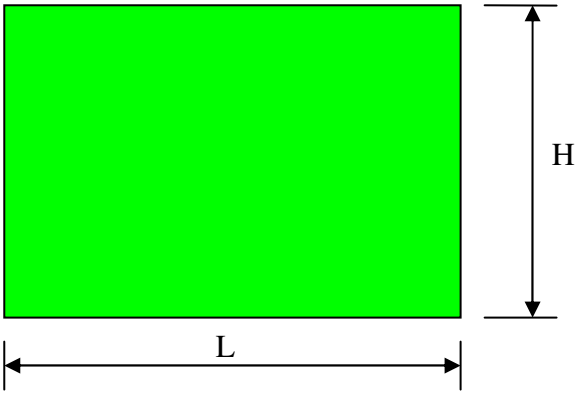


Figure 3-11: Rectangularized Frame Cavity (needed for calculation)

Heat Flow direction is calculated in Conrad and according to screen, heat flow direction can be “RIGHT” (Figure 3-12), “LEFT” (Figure 3-13), “VERTICAL DOWN” (Figure 3-14) and “VERTICAL UP” (Figure 3-15) which depends of temperatures on rectangularized frame cavity sides.

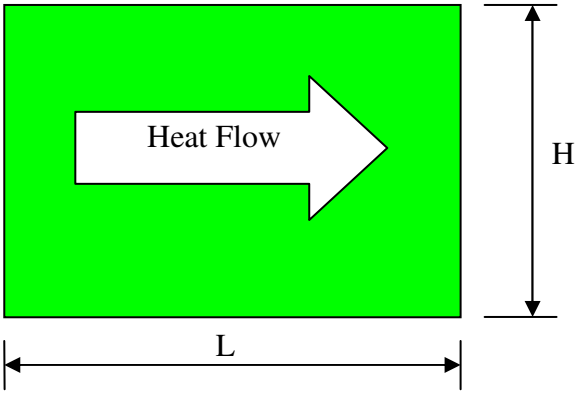


Figure 3-12: RIGHT Heat Flow Direction (According to Screen)

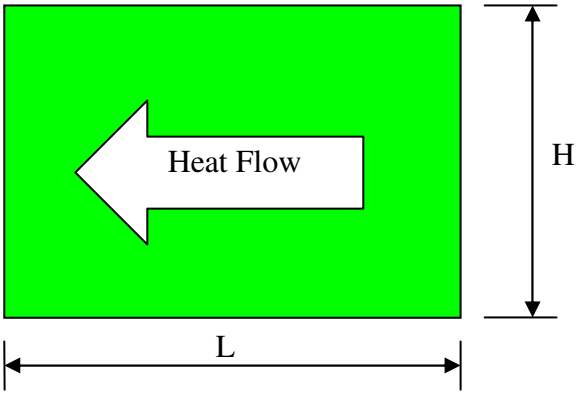


Figure 3-13: LEFT Heat Flow Direction (According to Screen)

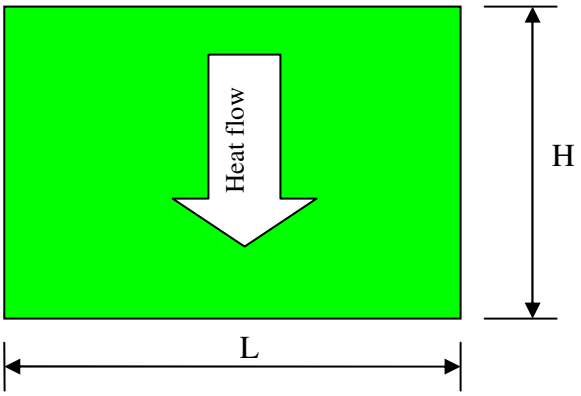


Figure 3-14: VERTICAL DOWN Heat Flow Direction (According to screen)

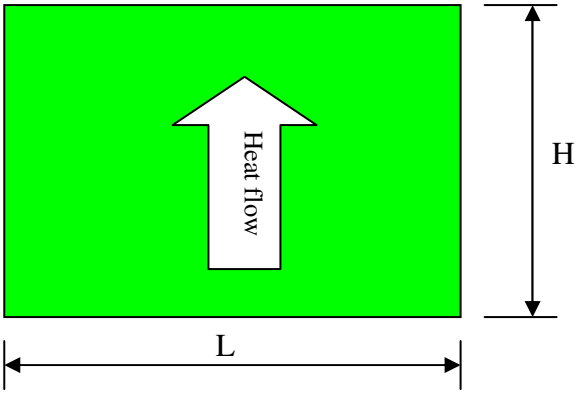


Figure 3-15: VERTICAL UP Heat Flow Direction (According to Screen)

Frame cavity in THERM5 is presented by three dimensions (

Figure 3-16) but only two dimensions can be seen. Third dimension is presented by value “jambheight”.

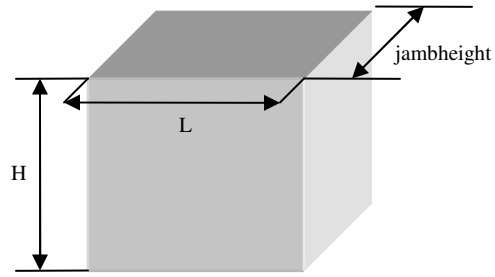


Figure 3-16: Frame Cavity Presentation in THERM

According to gravity arrow direction (in 3D space) and screen heat flow direction, heat flow direction in 3D (or according to gravity arrow) can be: “HORIZONTAL”, “VERTICAL UP”, VERTICAL DOWN”, “JAMB HORIZONTAL” and “JAMB VERTICAL”.

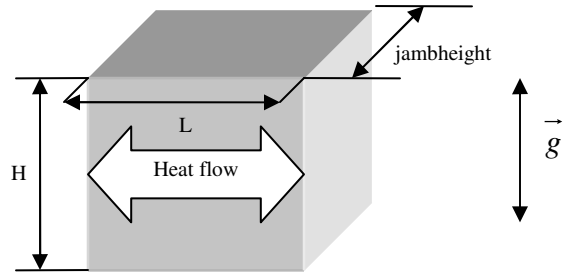


Figure 3-17: HORIZONTAL Heat Flow Direction (According to Gravity Arrow)

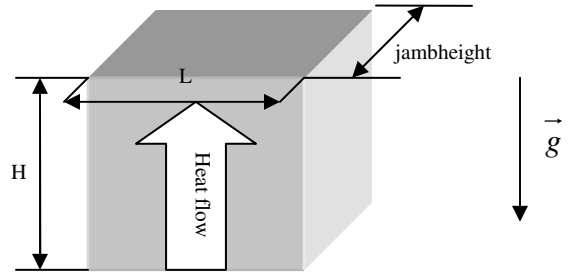


Figure 3-18: VERTICAL UP Heat Flow Direction (According to Gravity Arrow)

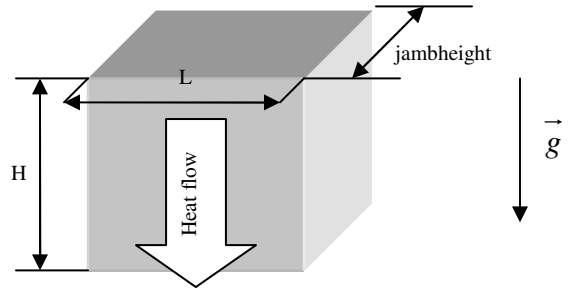


Figure 3-19: VERTICAL DOWN Heat Flow Direction (According to Gravity Arrow)

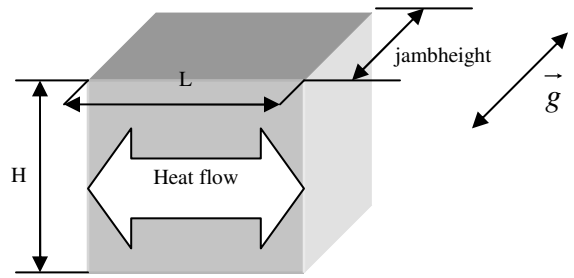


Figure 3-20: JAMB HORIZONTAL Heat Flow Direction (According to Gravity Arrow)

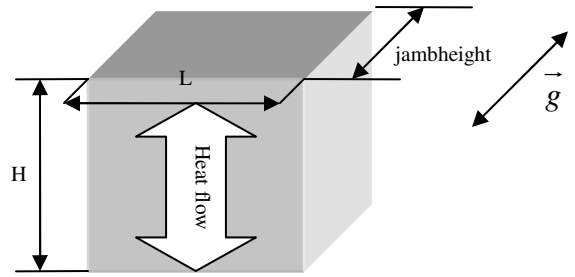


Figure 3-21: JAMB VERTICAL Heat Flow Direction (According to Gravity Arrow)

Important part of this algorithm is Frame Cavity transformation form 3D (or therm) presentation to 2D (needed for calculation) presentation. This transformation is presented by Figure 3-11 and

Figure 3-16.

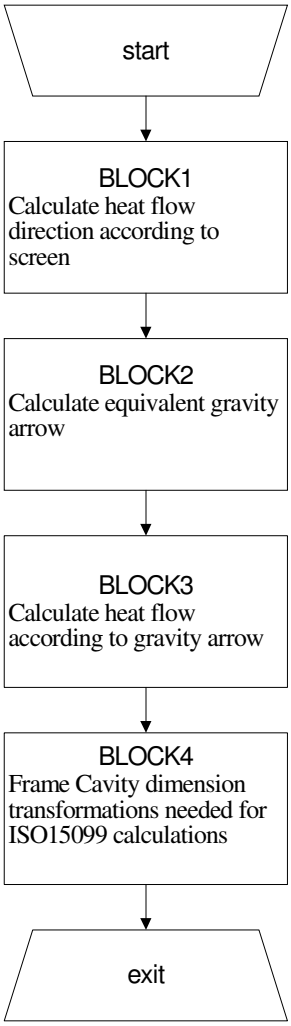


Figure 3-22: Gravity Arrow Algorithm

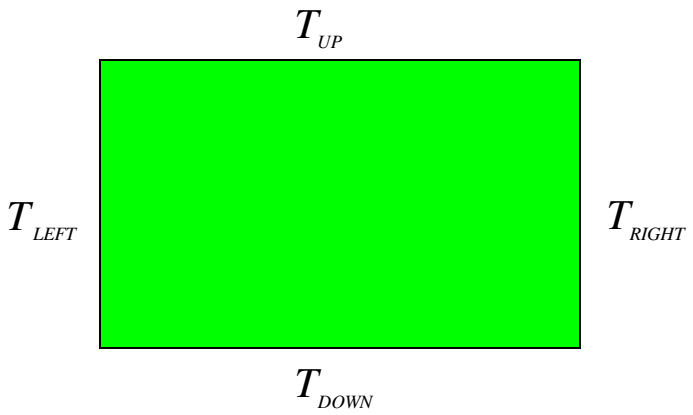


Figure 3-23: Temperatures on Rectangularized Frame Cavity

Depends of temperatures on equivalent frame cavity sides (Figure 2-1), screen heat flow direction is determined according to according to algorithm shown on

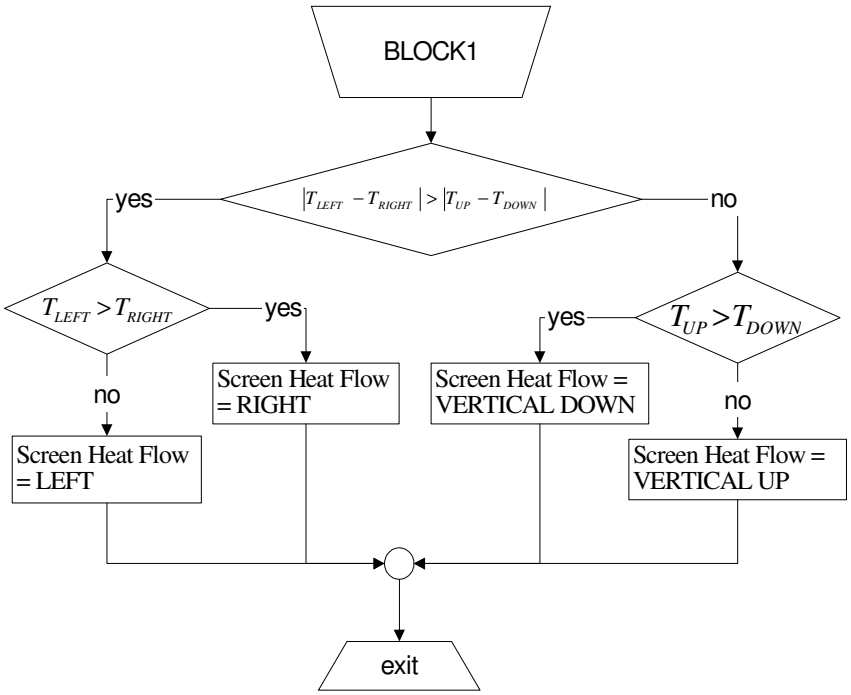


Figure 3-24: Screen Heat Flow Calculation Algorithm

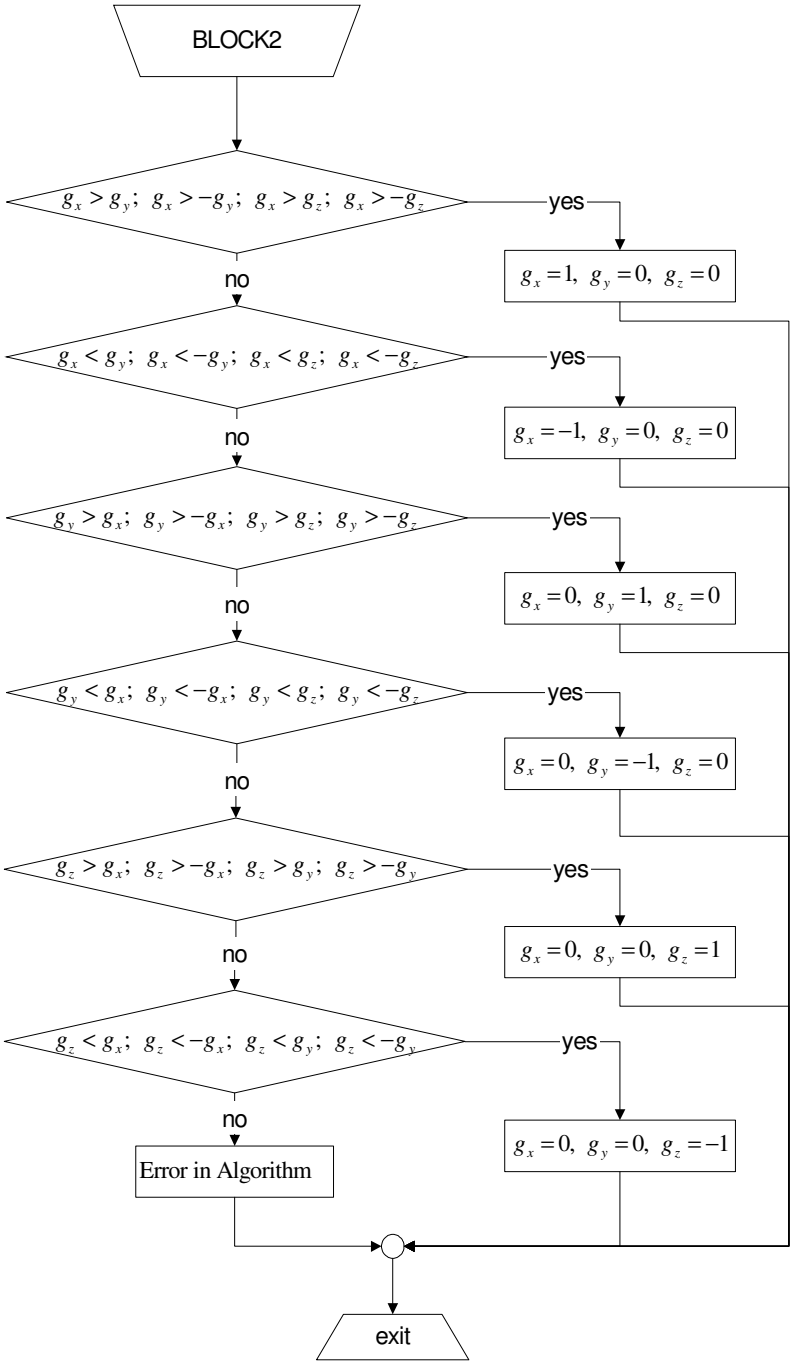


Figure 3-25: Equivalent Gravity Arrow Calculation Algorithm

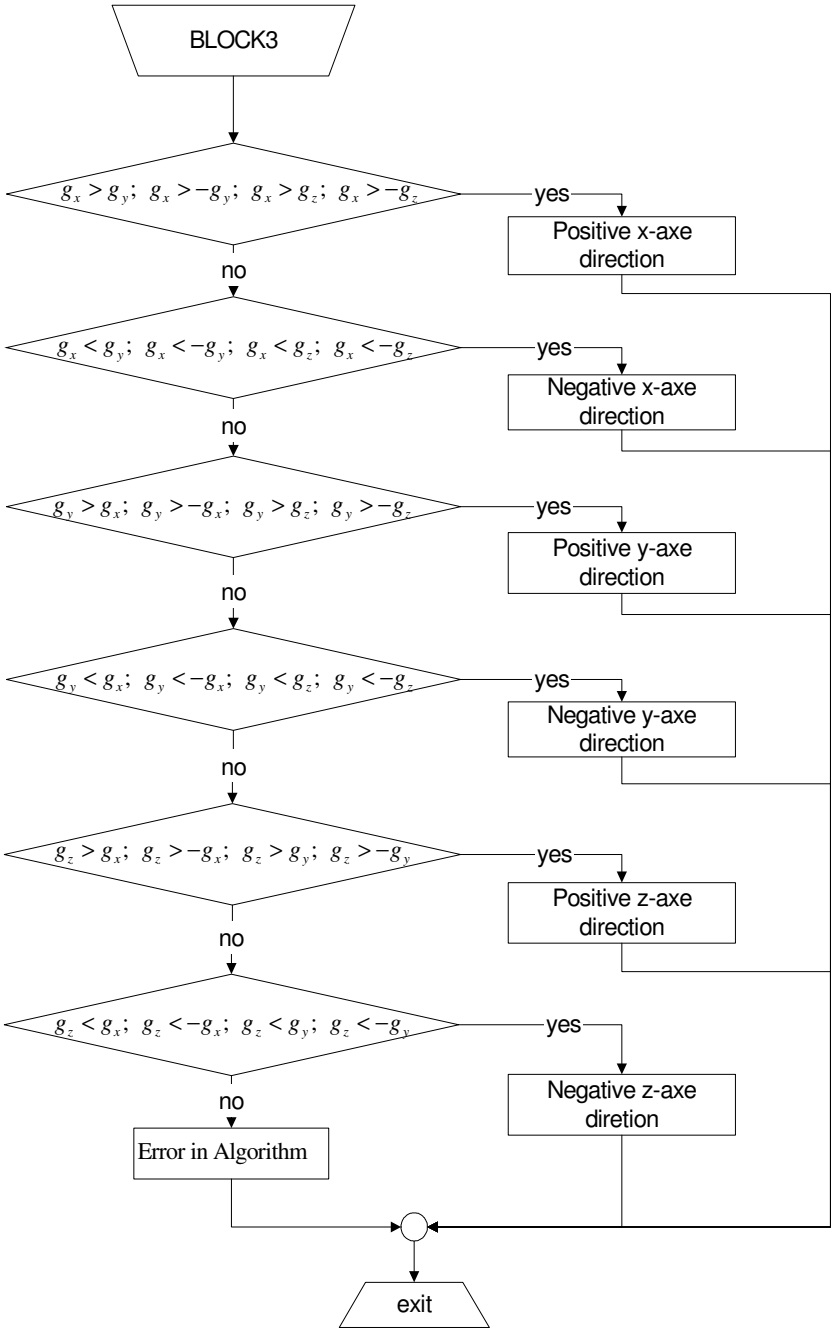


Figure 3-26: Gravity Heat Flow Algorithm

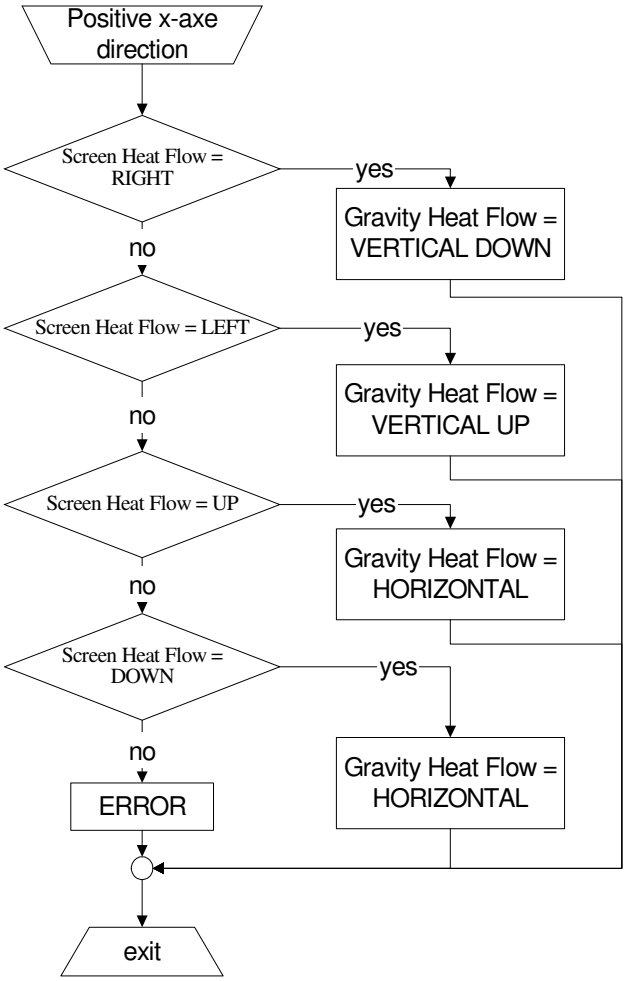


Figure 3-27: Positive x-axe Direction

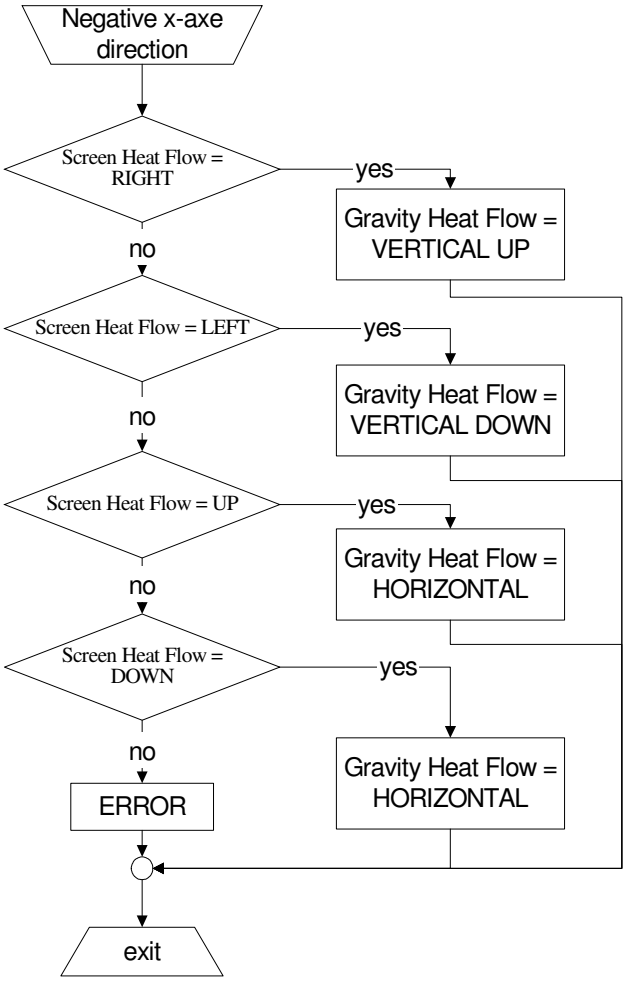


Figure 3-28: Negative x-axe Direction

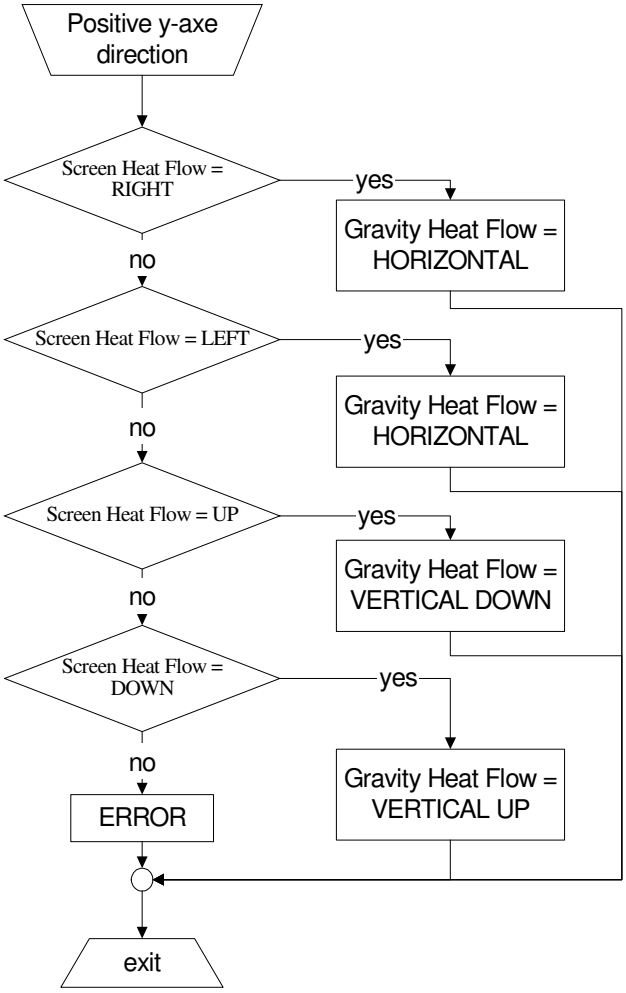


Figure 3-29: Positive y-axe Direction

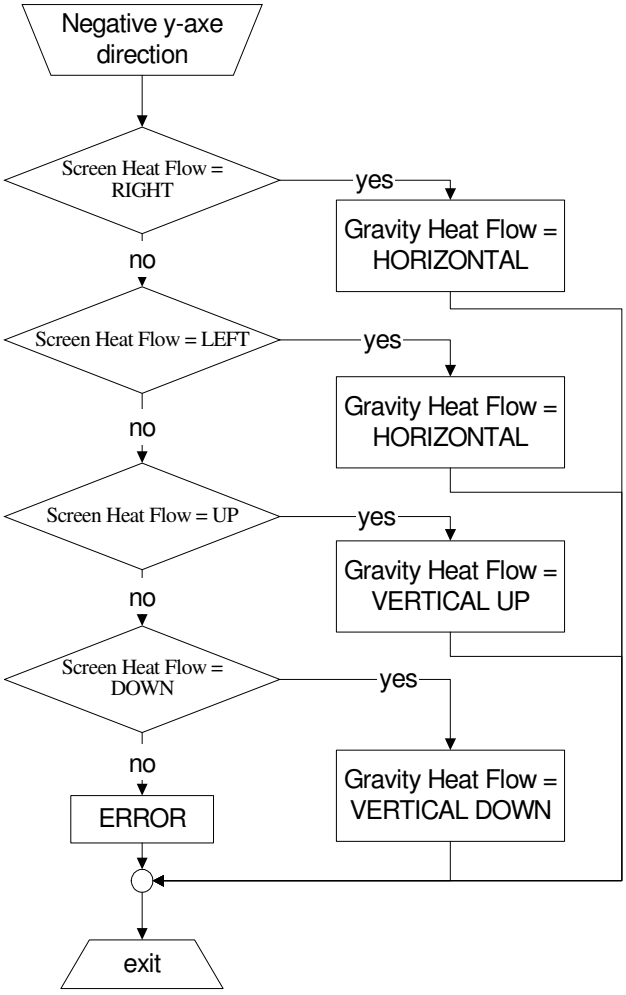


Figure 3-30: Negative y-axe Direction

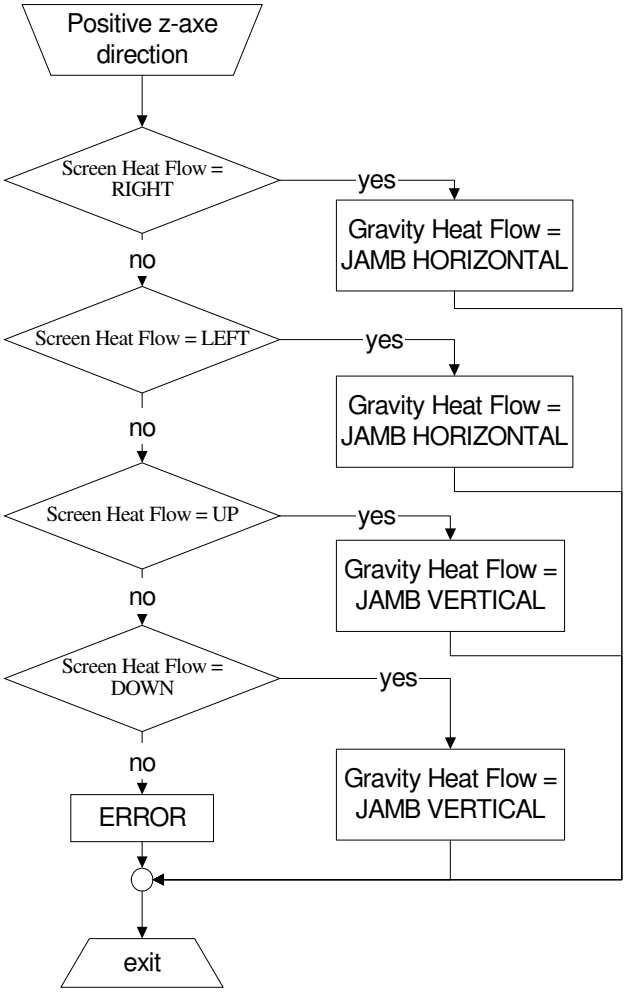


Figure 3-31: Positive z-axe Direction

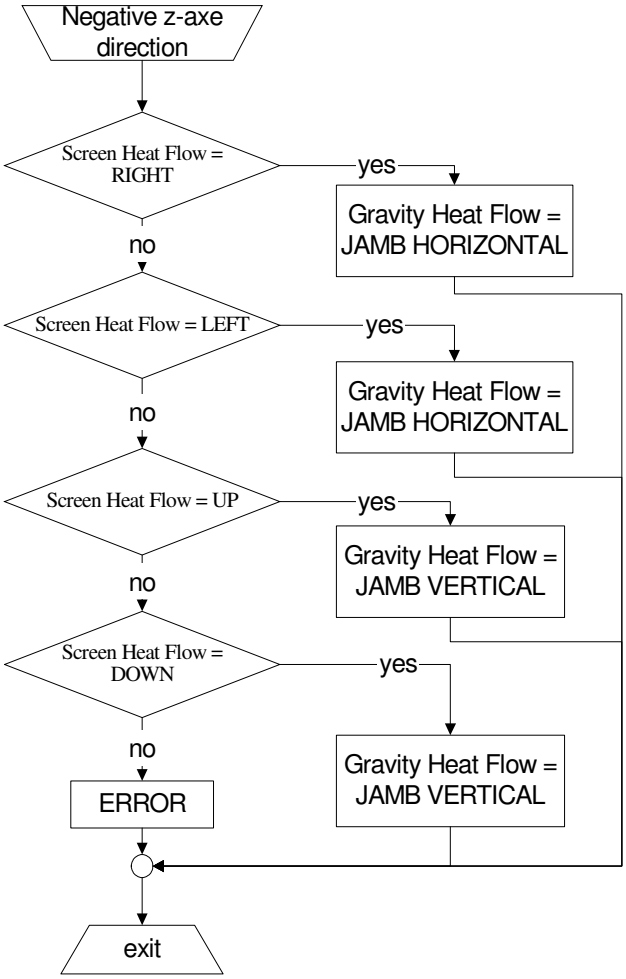


Figure 3-32: Negative z-axe Direction

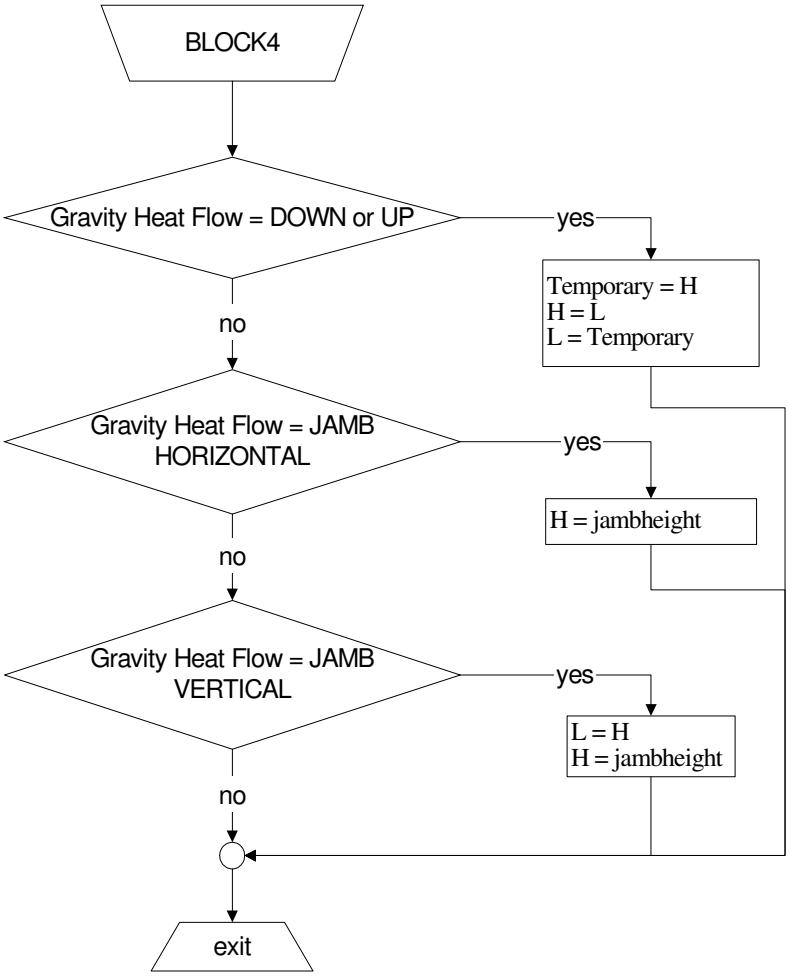


Figure 3-33: Transformation from 3D (THERM presentation) to 2D Frame Cavity

3.3. "Grid" Algorithm – Used for speed up Viewer

"Grid" algorithm is used to speed up view factor calculation. Main factor which has influence on program speed are calculation if blocking surfaces between two segments exist. Example of enclosure radiation segment which are segments used in view factor calculation are shown in Figure 3-34. View factor matrix can be very large and this depends of number of radiation enclosure segments. If you note that number of radiation enclosure segments is "n" than number of view factors are "n x n" and this can be large number. To remained that view factor is calculated by Eq. 1.3-49 and if ray intersection by third surface exist than by Eq. 1.3-50 (see Figure 1-7 also).

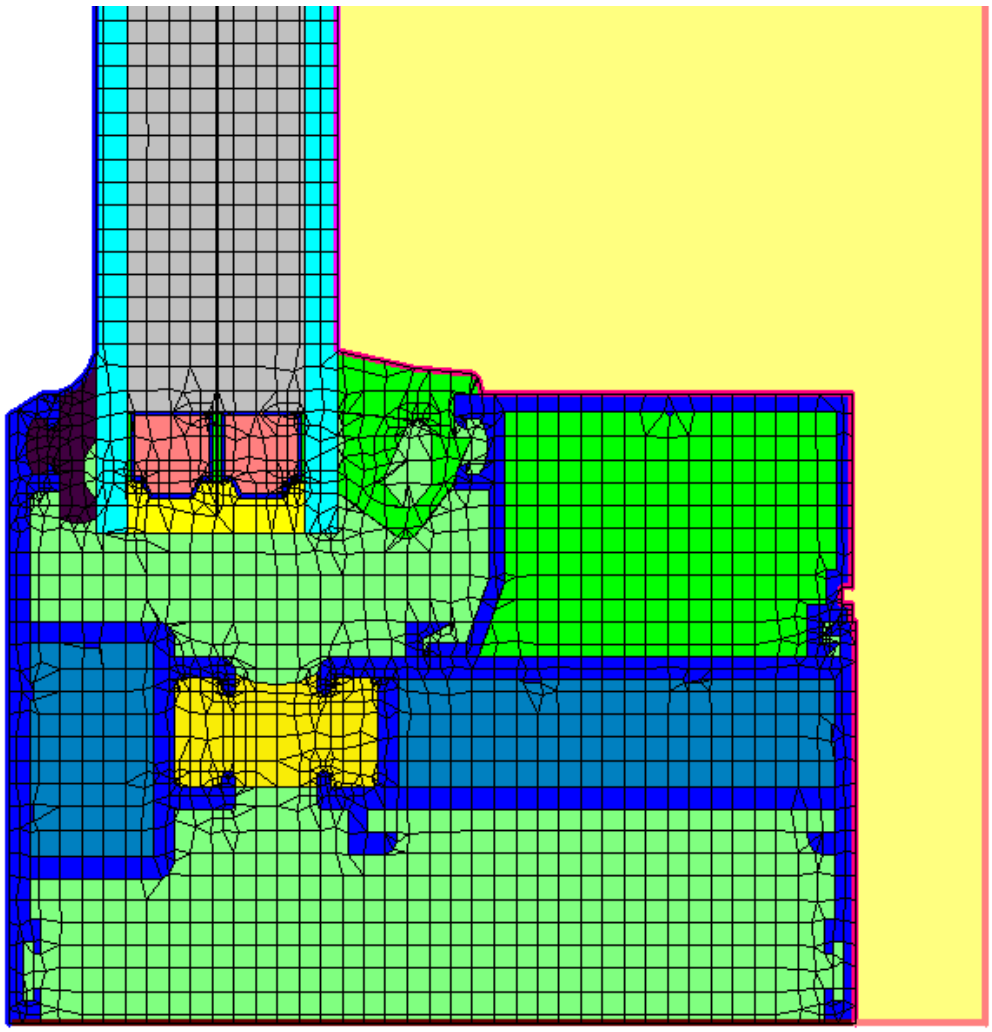


Figure 3-34: Example of Radiation Enclosure Segments (used in View Factor Calculations)

Therefore, in view factor calculation third surface intersection must be considered and for this purpose blocking surface are used. Any surface which can blocked ray of any other two segments must be signed as *blocking surface* and it is not always easy to determine in advance which surfaces are blocking. Sign number of blocking surfaces in one problem as “m”, and note that in most of cases m≈n. In calculation of view factor of any two surface program must pass through all surfaces which are signed as blocking to check if intersection exist (or simple – check if surface blocking ray between surfaces for which is currently calculate view factor). This leads that number of operations in view factor calculation is approximately equal n^3 (“n x n” view factors and “n” to determine number of blocking surfaces).

“Grid” Algorithm steps:

To speed up calculations algorithm set grid net (see Figure 3-35). Purpose of this net is that algorithm grouping blocking surfaces into the grid cells. Each cell contains blocking surfaces numbers that belong to this surface.

In view factor calculation between two segments algorithm determine grid cells that ray between two surfaces passes through.

Examine if there is surface that intercept ray. In this calculations algorithm uses only blocking surfaces which belong to cells that ray passes through (this rapidly decrease number of blocking surfaces which program will check).

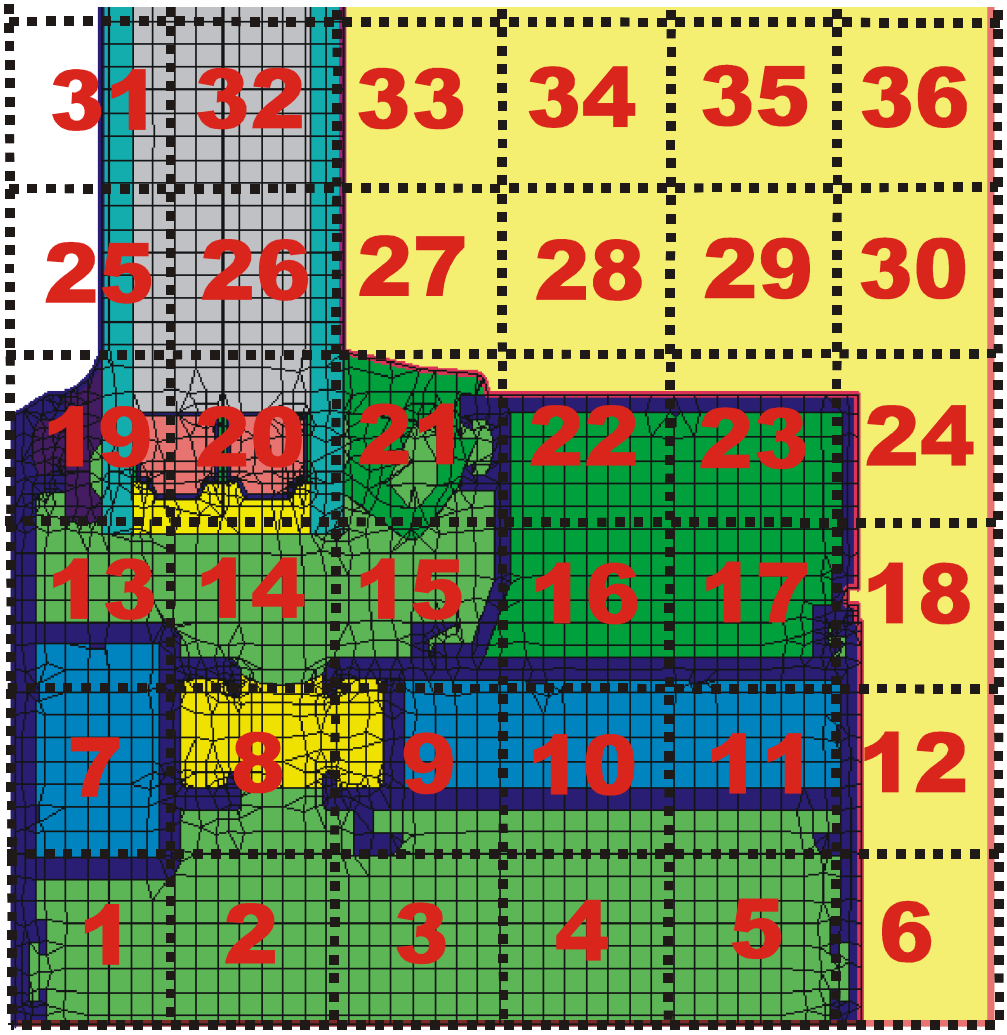


Figure 3-35: Grid Net

For example in Figure 3-36 is shown example where is no interception between two surfaces, but without using algorithm program will pass through all blocking surfaces to check if interception exists.

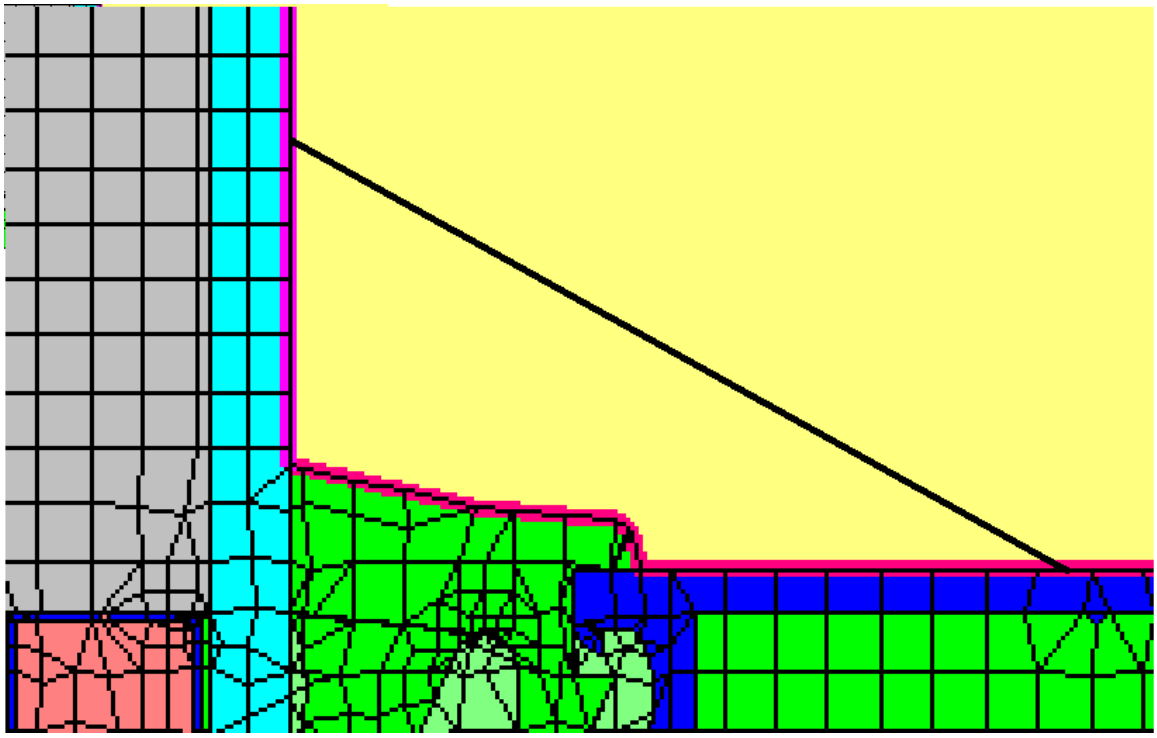


Figure 3-36: Ray Between Two Surfaces

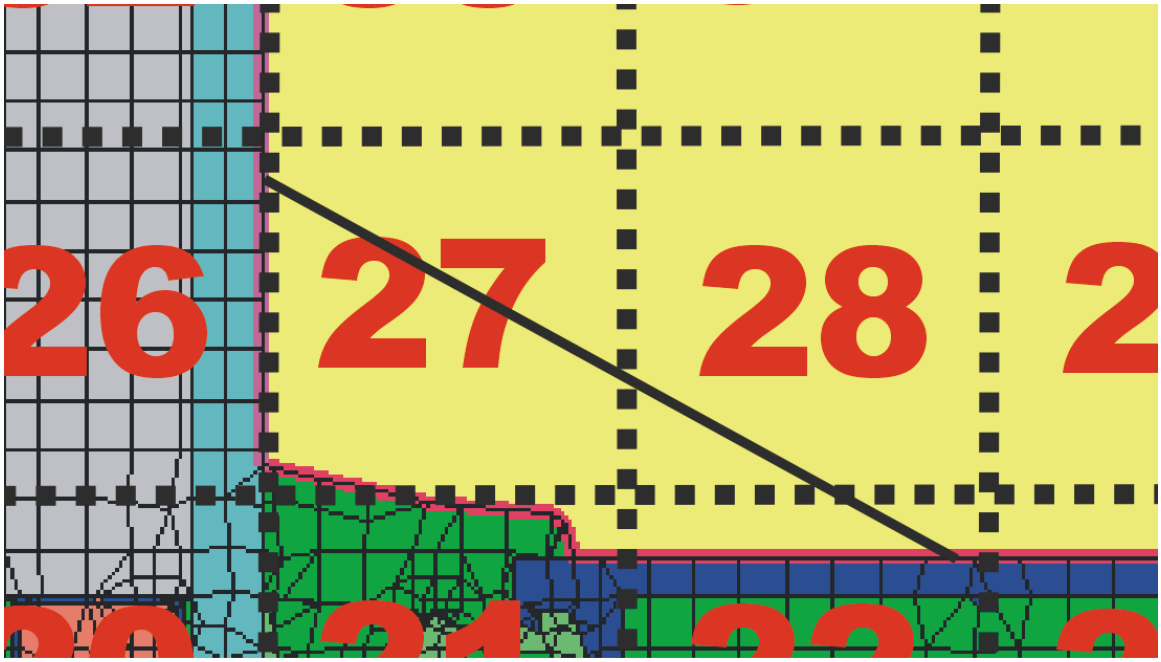


Figure 3-37: Ray in Grid Net

Same ray in grid net (see Figure 3-37) passes only through cells numbered 22, 27 and 28 which cause that algorithm will examine blocking surfaces only in these three cells

and this mean much less number of blocking surfaces (cell 28 is empty) to consideration.

There is no equation from which can be obtained grid net density (number of grid cells) because this also depends of problem geometry, number of segments and number of blocking surfaces. Note also that too much cells can decrease program speed and have very bad effects on program speed. For details see chapter 6.3.

3.4. Shadowing

Self shadowing and third surface shadowing are important rules in view factor calculations.

3.4.1. Calculating Surface Normal

Surface is presented by line between two points which are presented by coordinates (x_1, y_1) - first point and (x_2, y_2) - second point. Surface normal is calculated by following equations:

$$l = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \qquad \textit{Eq. 3.4-1}$$

$$x_n = \frac{y_2 - y_1}{l} \qquad \textit{Eq. 3.4-2}$$

$$y_n = -\frac{x_2 - x_1}{l} \qquad \textit{Eq. 3.4-3}$$

where x_n and y_n presents surface normal coordinates.

3.4.2. Self Shadowing

To check if segment can “see” any point (in 2D geometry) simple equations are applied. Note that line segment is presented by normal which point into the view surface direction (Figure 3-38).

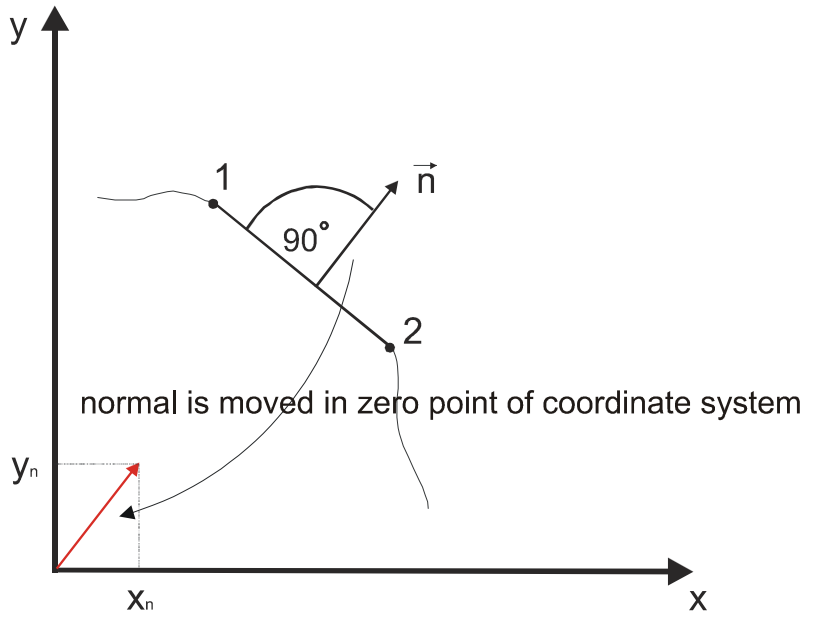


Figure 3-38: Surface Normal

Surface normal is moved in zero point of coordinate system and coordinate of normal must satisfy following equation:

$$\sqrt{x_n^2 + y_n^2} = 1 \qquad \textit{Eq. 3.4-4}$$

To calculate if surface can “see” point in 2D space, point also must be presented by vector (Figure 3-39).

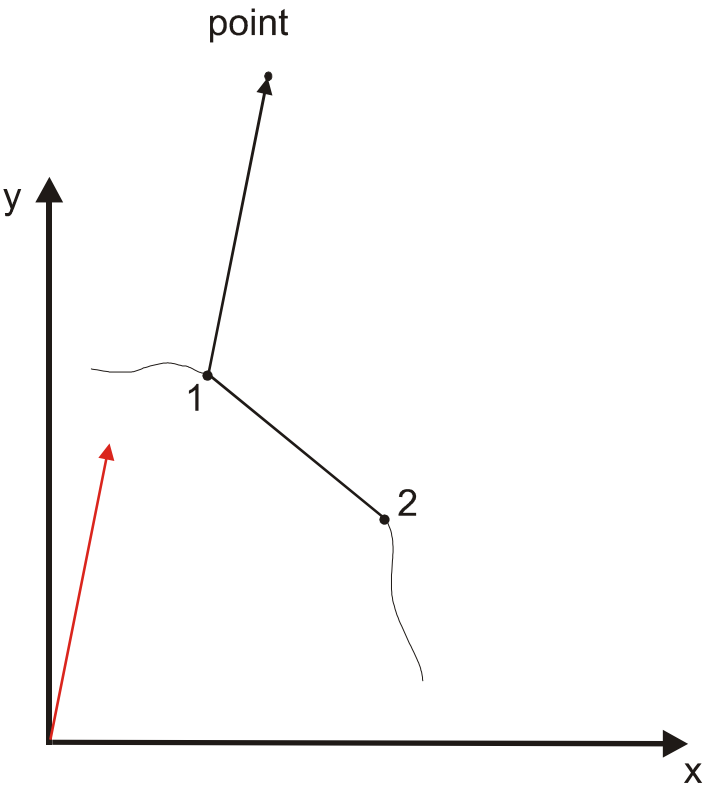


Figure 3-39: Point vector

To check if point belongs to view area (or can surface "see" the point) of surface following equations must be satisfied:

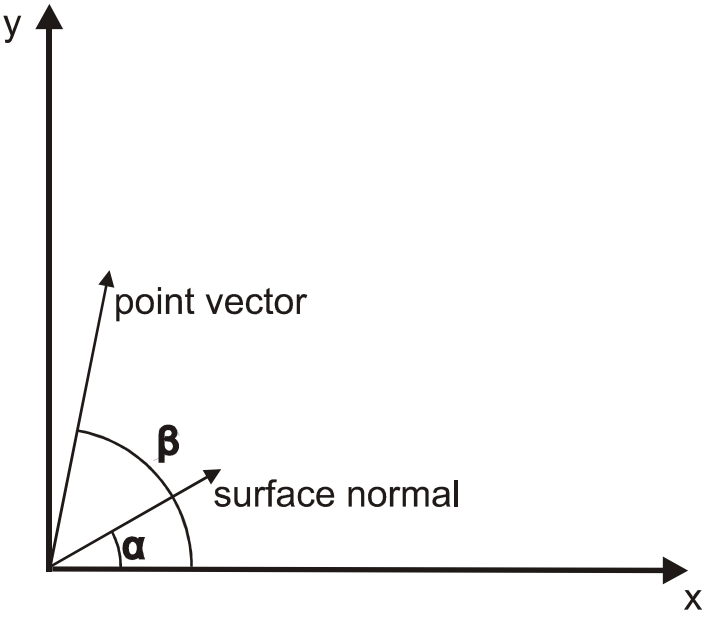


Figure 3-40: Angles Between Vectors

Sign coordinates of surface normal as $x_n \ and \ y_n$ and coordinates of point vector as $x_p \ and \ y_p$.

Check if point belongs to view area:

$$-90^o < \beta - \alpha < 90^o \qquad \textit{Eq. 3.4-5}$$

which is equal with

$$\cos(\beta - \alpha) > 0 \qquad \textit{Eq. 3.4-6}$$

after trigonometric transformation

Eq. 3.4-6

$$\cos(\alpha) * \cos(\beta) + \sin(\alpha) * \sin(\beta) > 0$$

Eq. 3.4-7

from Figure 3-40 following equations are obtained:

$$\cos(\alpha) = \frac{x_n}{\sqrt{(x_n^2 + y_n^2)}}$$

$$\sin(\alpha) = \frac{y_n}{\sqrt{(x_n^2 + y_n^2)}}$$

$$\cos(\beta) = \frac{x_p}{\sqrt{(x_p^2 + y_p^2)}}$$

$$\sin(\beta) = \frac{y_p}{\sqrt{(x_p^2 + y_p^2)}}$$

Eq. 3.4-8

note that equation $\sqrt{(x^2 + y^2)} > 0$ is always satisfied. After substituting Eq. 3.4-8 into the Eq. 3.4-7:

$$x_n * x_p + y_n * y_p > 0$$

Eq. 3.4-9

Self shadowing algorithm is shown on Figure 3-41:

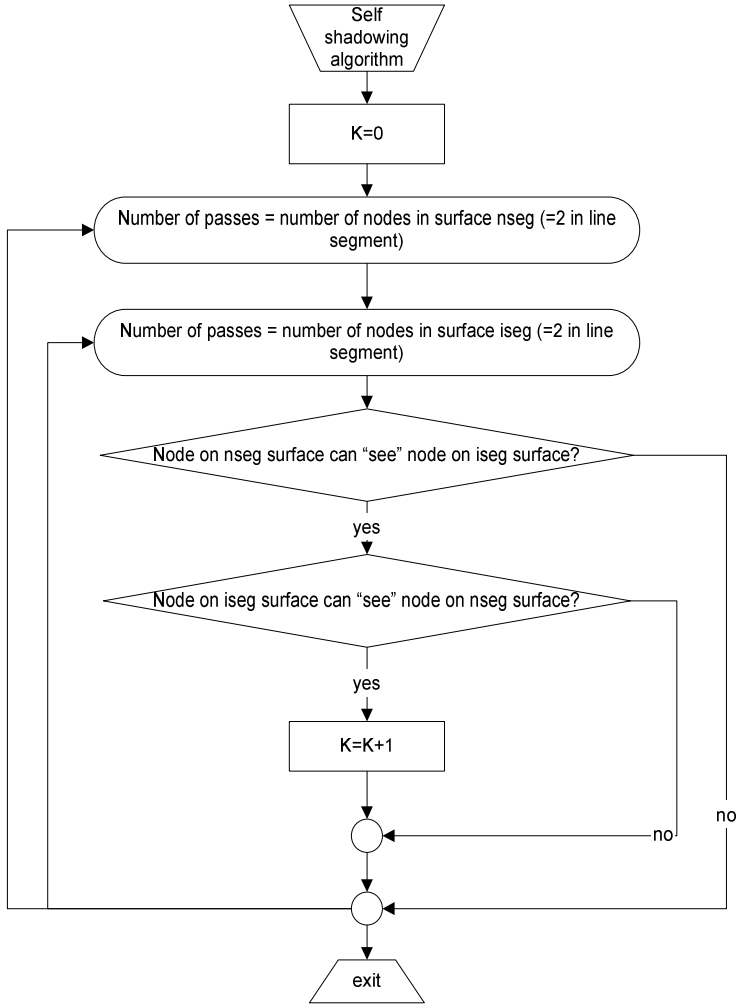


Figure 3-41: Self Shadowing Algorithm

There is three case of self shadowing which can be obtained:

- No self shadowing (k=4)
- Partial self shadowing (0<k<4)
- Total self shadowing (k=0)

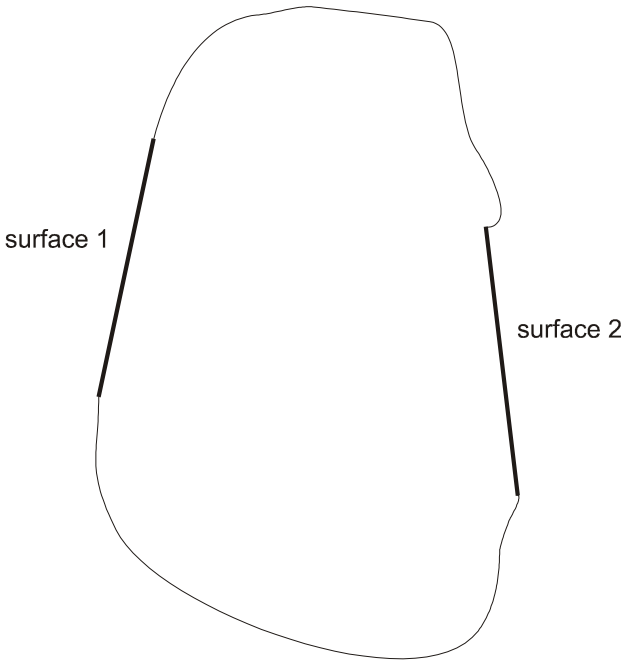


Figure 3-42: No Shadowing

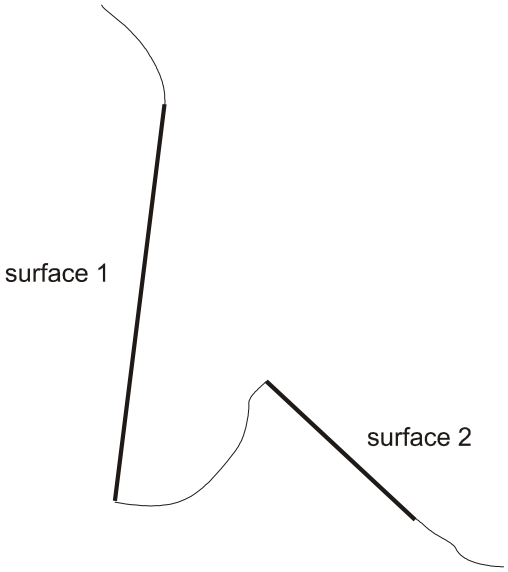


Figure 3-43: Partial Self Shadowing

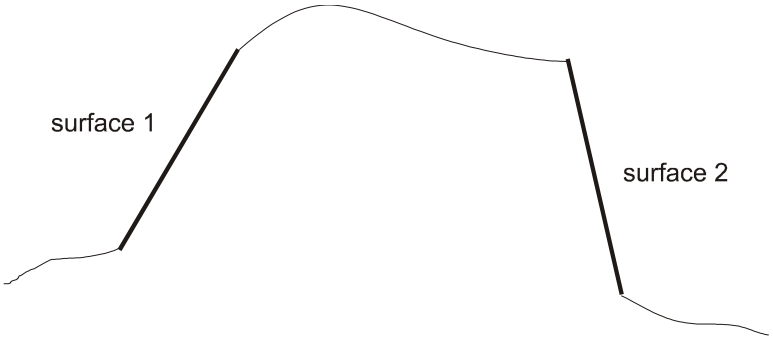


Figure 3-44: Total Self Shadowing

3.4.3. Third Surface Shadowing

Third surface shadowing is occurring when there is any third surface which blockade ray between surfaces. As in previous chapter, there are three possibilities for third surface shadowing:

- No third surface shadowing (Figure 3-42)
- Partial third surface shadowing
- Total third surface shadowing

To examine if intersection exist in determined area, algorithm uses line equations through two points (Figure 3-45).

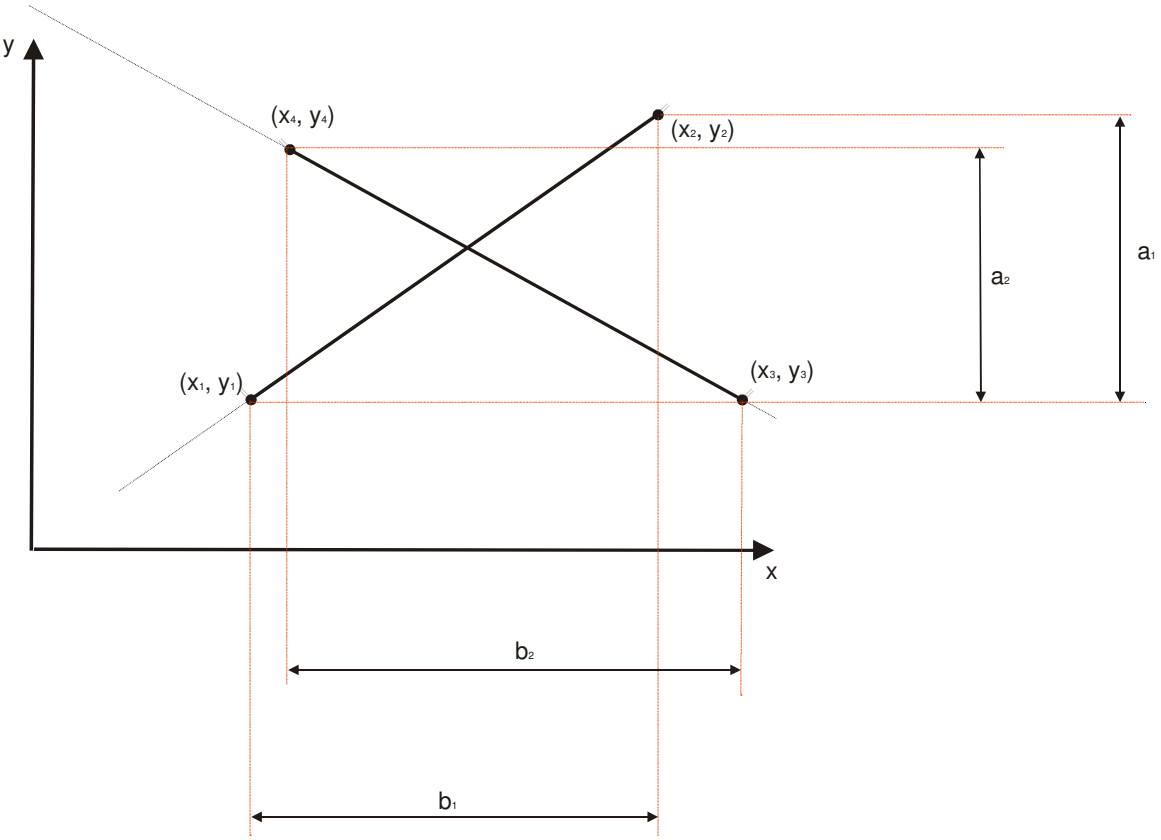


Figure 3-45: Intersection Between Two Points

Line equations between two points for these two lines are:

$$y - y_1 = \frac{(y_2 - y_1)}{(x_1 - x_2)} * (x - x_1)$$
$$y - y_3 = \frac{(y_4 - y_3)}{(x_3 - x_4)} * (x - x_3) \qquad \textit{Eq. 3.4-10}$$

Calculating intersection point from Eq. 3.4-10, following result is obtained:

$$x_i = \frac{c_1 * b_2 - c_2 * b_1}{a_1 * b_2 - a_2 * b_1}$$
$$y_i = \frac{c_2 * a_1 - c_1 * a_2}{a_1 * b_2 - a_2 * b} \qquad \textit{Eq. 3.4-11}$$

where x_i and y_i denotes coordinates of intersection point, and

$$c_1 = x_1 * y_2 - x_2 * y_1$$
$$c_2 = x_3 * y_4 - x_4 * y_3$$
$$a_1 = y_2 - y_1$$
$$a_2 = y_4 - y_3 \qquad \textit{Eq. 3.4-12}$$
$$b_1 = x_1 - x_2$$
$$b_2 = x_3 - x_4$$

To determine if intersection point is on line between end points (intersection exist) or intersection point is out of line (no exist) see Figure 3-46 and Figure 3-47.

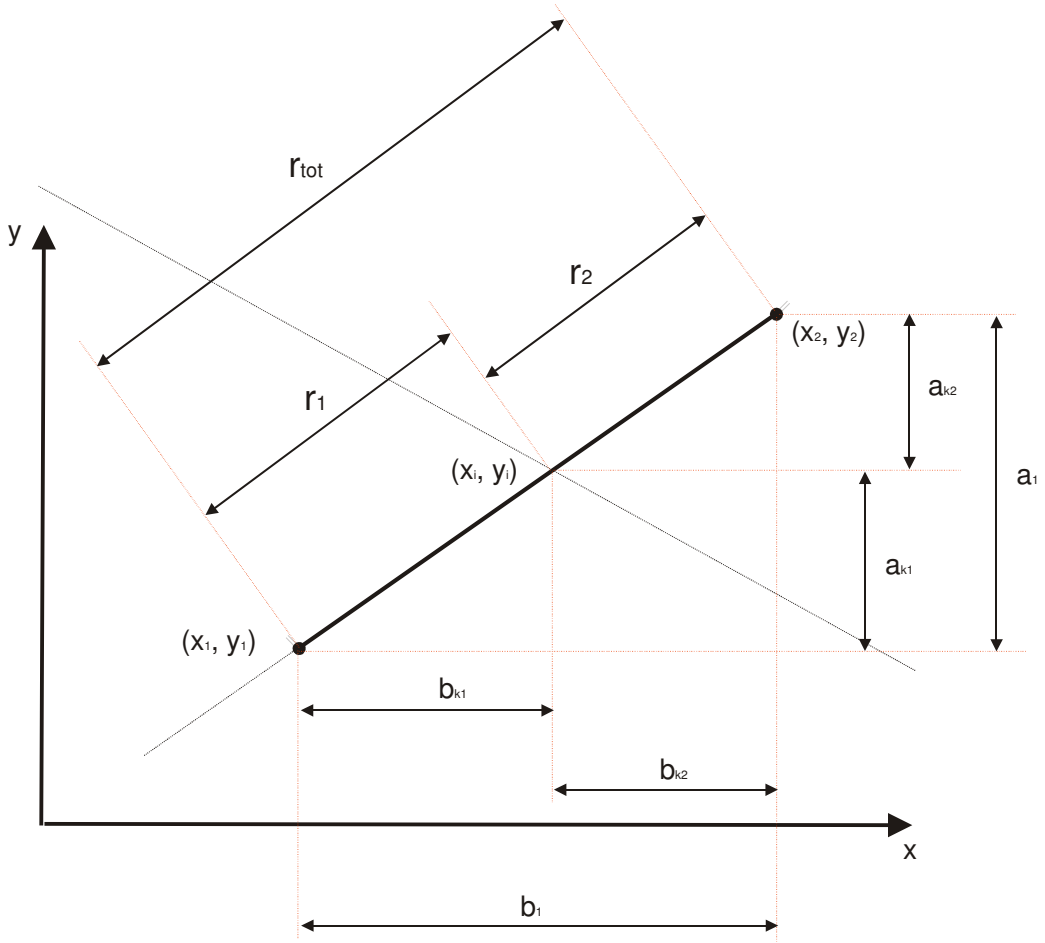


Figure 3-46: Intersection Exist

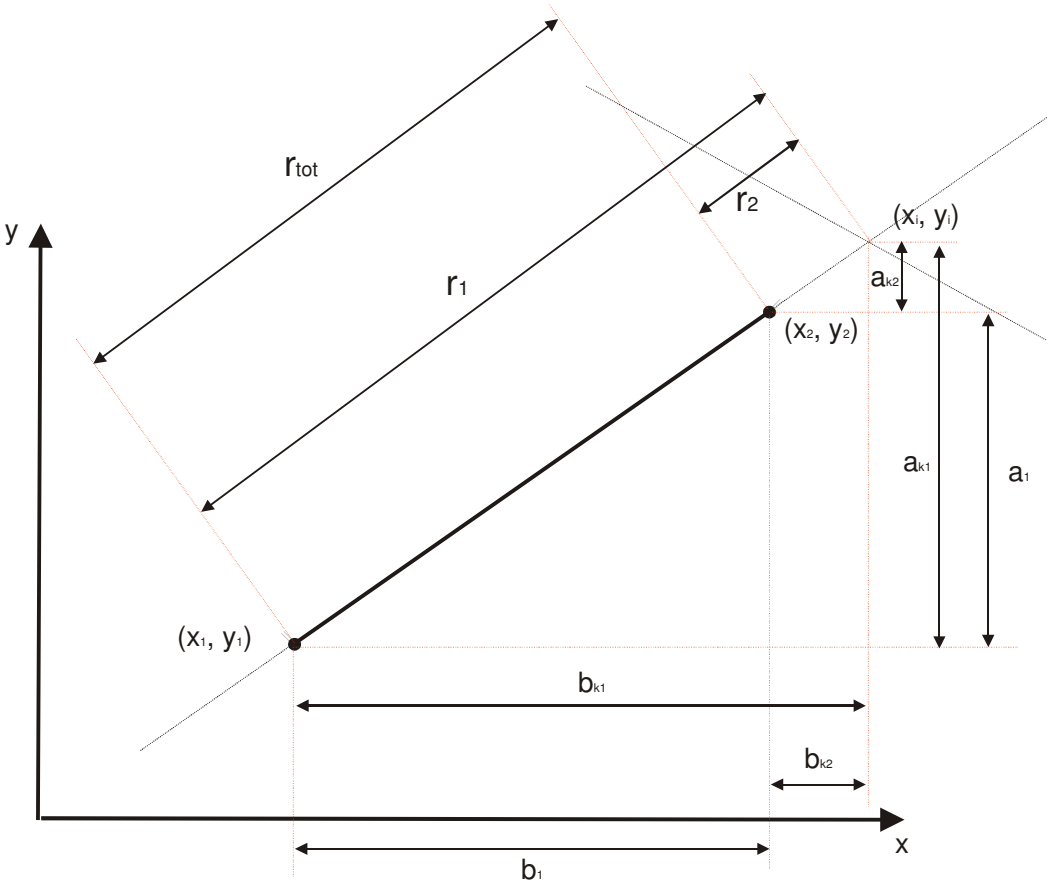


Figure 3-47: No Intersection

Whether intersection exist or not following equation are always satisfied:

$$r_{tot} = \sqrt{a_1^2 + b_1^2}$$
$$r_1 = \sqrt{a_{k1}^2 + b_{k1}^2} \qquad \textit{Eq. 3.4-13}$$
$$r_2 = \sqrt{a_{k2}^2 + b_{k2}^2}$$

in case when intersection exist

$$r_{tot} = r_1 + r_2 \qquad \textit{Eq. 3.4-14}$$

and if intersection not exist

$$r_{tot} < r_1 + r_2 \qquad \textit{Eq. 3.4-15}$$

Therefore, intersection between two lines exists only if Eq. 3.4-14 is satisfied for both line segments.

3.5. Frame Cavity Rectangularization

3.5.1. Rectangularization Algorithm

For unventilated irregularly shaped frame cavity, the geometry shall be converted into equivalent rectangular cavity according to ISO/DIS 10077-2. For these cavities, the following procedure shall be used to determine which surfaces belong to vertical and horizontal surfaces of equivalent rectangular cavity:

- any surface whose normal is between 315 and 45 degrees is left vertical surface
- any surface whose normal is between 45 and 135 degrees is bottom horizontal surface
- any surface whose normal is between 135 and 225 degrees is right vertical surface
- any surface whose normal is between 225 and 315 degrees is top horizontal surface

Assume that frame cavity is divided into the finite elements which number of edge sides whose normal is between 315 and 45 degrees is equal to “n”. Rectangularization of left vertical surface is calculated using following equation:

- Temperature

$$LeftTemp = \frac{\sum_{i=1}^{n} l_i * temp_i}{TotalLength} \qquad \textbf{Eq. 3.5-1}$$

where l_i is line segment length, $temp_i$ is mean segment temperature (mean temperature is calculate using mean temperature of segment nodes), $TotalLength$ is sum of all segment length which surface normal is between 315 and 45 degrees and “n” is number of segments which surface normal is between 315 and 45 degrees.

- Emissivity

$$LeftEmis = \frac{\sum_{i=1}^{n} l_i * emis_i}{TotalLength} \qquad \textbf{Eq. 3.5-2}$$

where l_i is line segment length, $emis_i$ is segment emissivity, $TotalLength$ is sum of all segment length which surface normal is between 315 and 45 degrees and “n” is number of segments which surface normal is between 315 and 45 degrees.

Calculation of other three (top, bottom and right) rectangularized sides are calculated using same equations (Eq. 3.5-1 and Eq. 3.5-2).

3.5.2. Rectangularization of Non Existing Sides Algorithm

“Non Existing Side” occurs when number of segment which belongs to one of sides (left, right, top and bottom) is equal to zero. It means that equations Eq. 3.5-1 and Eq. 3.5-2 can’t be applied because $TotalLength$ is equal to zero.

Suppose that left side is “Non Existing” in rectangularized frame cavity (see Figure 3-48).

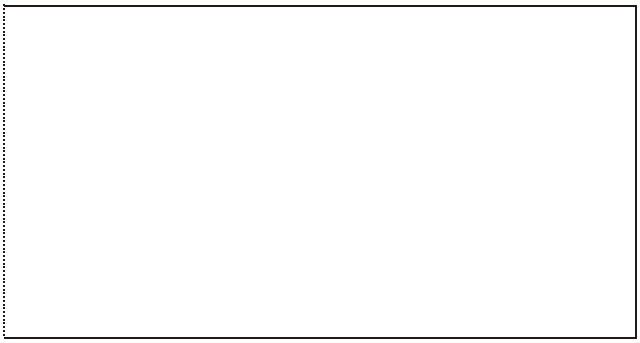


Figure 3-48: "Non Existing" Left Side of Rectangularized Frame Cavity

$$LeftSideTemperature = theta(nodewith \min x\,coordinate) \qquad \textbf{Eq. 3.5-3}$$

$$LeftSideEmissivity = (TopSideEmissivity + BottomSideEmissivity)/2 \qquad \textbf{Eq. 3.5-4}$$

Note that LeftSideTemperature is equal with temperature at node with minimal x-coordinate. Similar equations are applied on other three sides:

- Top

$$TopSideTemperature = theta(nodewith \max y\,coordinate) \qquad \textbf{Eq. 3.5-5}$$

$$TopSideEmissivity = (LeftSideEmissivity + RightSideEmissivity)/2 \qquad \textbf{Eq. 3.5-6}$$

- Bottom

$$BottomSideTemperature = theta(nodewith \min y\,coordinate) \qquad \textbf{Eq. 3.5-7}$$

$$BottomSideEmissivity = (LeftSideEmissivity + RightSideEmissivity)/2 \qquad \textbf{Eq. 3.5-8}$$

- Right

$$RightSideTemperature = theta(nodewith \max x\,coordinate) \qquad \textbf{Eq. 3.5-9}$$

$$RightSideEmissivity = (TopSideEmissivity + BottomSideEmissivity)/2 \qquad \textbf{Eq. 3.5-10}$$

4. Description of Conrad Subroutines

This chapter describes Conrad routines which use theoretical background described in previous chapters.

4.1. Routine CONRAD

Routine CONRAD is main routine which contains all calculations described in precious chapters. Routine CONRAD is used by THERM5 and it is implemented as *dynamic link library* (or dll) routine.

List of arguments:

in – *(input file)* Input file name (*.con file)

out – *(output file)* Output file name (*.o file)

gaus – *(output file)* Output file name for flux results (*.sol file)

view – *(output file)* Output file from Viewer – view factors (*.view file)

nerr – *(output value)* Error flag

List of commons:

common /blk03/

nummat - Number of materials

numnp - Number of node points

numel - Number of elements

igeom - type of geometry
eq.1: axisymetric
eq.2: 2D planar

iband - bandwidth minimization
eq.0: no minimization
eq.1: minimization
eq.2: minimization – nodal destination

nsl - number of slide lines (future implement)

nslvt - total number of slave nodes (future implement)

nmsrt - total number of master nodes (future implement)

numels - number of slide line elements??? (future implement)

nprof - matrix profile for actol solver

sigma - Stefan-Boltzmann constant [$5.6697e-8\ \frac{W}{m^2K^4}$]

irtyp - equal 4 (always)

itmax - Maximal number of Radiosity iterations – nonlinear iterations (100)

tolb - Radiosity convergence tolerance (def=1e-4)

numelt - ?

igenm - Thermal generation rate multiplier flag
eq.0: no thermal generation
eq.1: thermal generation

igene - ?

isotr - flag for material type
eq.0: isotropic
eq.1: orthotropic

mcut - ?

isoln - equation solver for [A]{x}={b} (always 0)
eq.0: fissle direct solver
eq.1: actol direct solver
eq.2: nonsymmetric solver

common /blk04/

nit - Number of nodes with temperature initial condition???

ntbc - Number of nodes with temperature boundary condition

nfbc - Number of flux boundary condition segments

ncbc - Number of convection boundary condition segments

nrbc - Number of radiation (Black Body) boundary condition segments

nebc - Number of enclosure radiation segments

necurv - Number of emissivity vs wavelength curves

nfelm - Number of fluid flow elements

common /blk06/

nonl - Type of problem [NOTE: Radiation makes problem nonlinear]
eq.0: linear
eq.1: nonlinear

maxrf - Maximum number of conductivity matrix reformations

maxit - Maximum number of equilibrium iterations per reformation

tol - Convergence tolerance (def = 1e-6)

relax - Divergence control parameter $0 < p \leq 1$ (def = 1)

step - Number of steps to decrement divergence control parameter

nsteps - Step value for which is Divergence control parameter is decremented

common /blk08/

title(1) - Project name

head - name, date and version information

longo - debug information flag
eq.0: no debug information
eq.1: debug information

iconv - type of temperature scale
eq.1: Celsius (tscale='c' or 'C')

eq.2: Fahrenheit (tscale='f' or 'F')

eq.-1: Kelvin (tscale='k' or 'K')

eq.-2: Rankin (tscale='r' or 'R')

common /blk11/

h - Values of master element functions in gauss points for numerical integration

dhdz - Values of first derivative of master element functions by x coordinate in Gauss points for numerical integration

dhde - Values of first derivative of master element functions by y coordinate in Gauss points for numerical integration

common /blk12/

mpcurv - Number of data points per curve

common /blk18/

pi - Value of pi number (eq. 3.14159265358979323846)

twopi - Value of two pi (eq. 2*pi)

common /coniob/

iobuf - Buffer to store information

common /iofilx/

fnames - Array to store names of input-output files

common /errchk/

nperr - Error counter

common /iter/

iter - Flag to show if new iteration is needed because frame cavity conditions are not satisfied

eq.0: new iteration is not needed

eq.1: need new iteration

common /blk11/

gcon - Array of constants for gas conductivity calculation

gvis - Array of constants for gas dynamic viscosity calculation

gcp - Array of constants for gas specific heat calculation

wght - Vector of Molecular weights for gasses

Program flow for Conrad routine is shown in Figure 4-1.

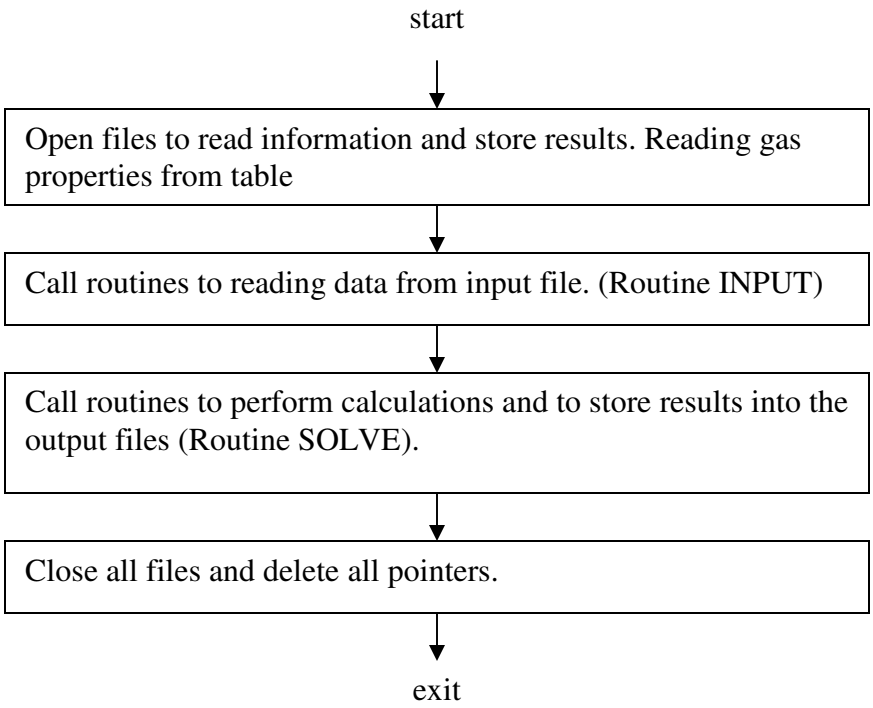


Figure 4-1: Program Flow for CONRAD Routine

4.2. Routine SOLVE

Routine SOLVE is used to perform calculations (see Figure 4-1).

List of arguments:

Icount – *(input/output value)* Current line number in input file(*.con)

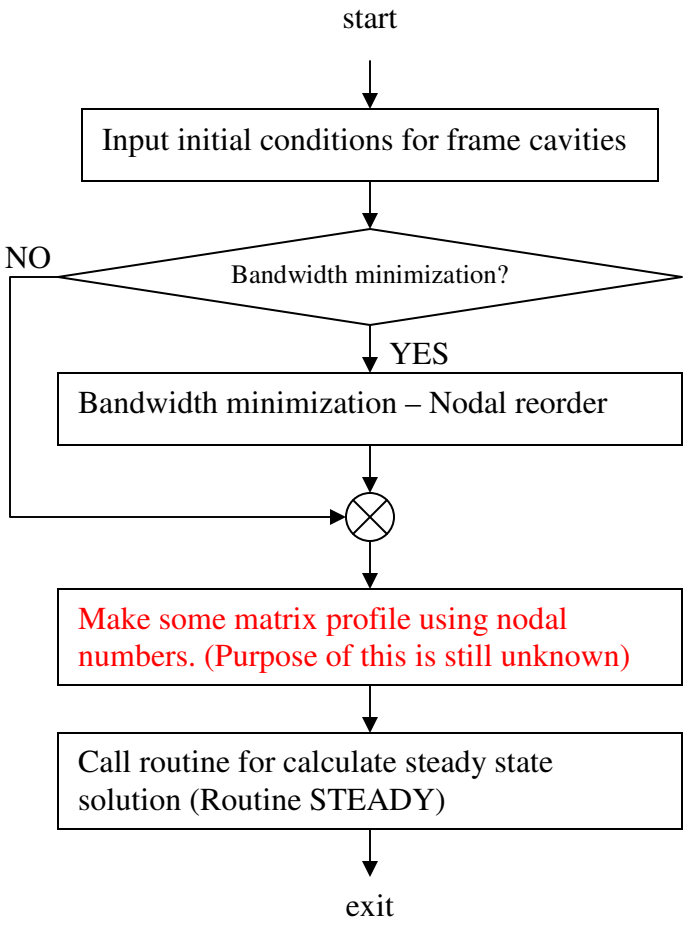


Figure 4-2: Program Flow for Routine SOLVE

4.3. Routine STEADY

Routine STEADY is used to calculate nodal temperatures according to element matrices and boundary conditions for *linear* and *nonlinear* problems. In this routine are called all routines which introduce element matrices and conditions at the defined boundaries. Routine STEADY also implement iterations for frame cavities (for both – *linear* and *nonlinear* problems) and automatic decrement of "relax" parameter.

List of arguments:

There is no list of arguments for this routine

Program flow diagram is shown in Figure 4-3, Figure 4-4 and Figure 4-5.

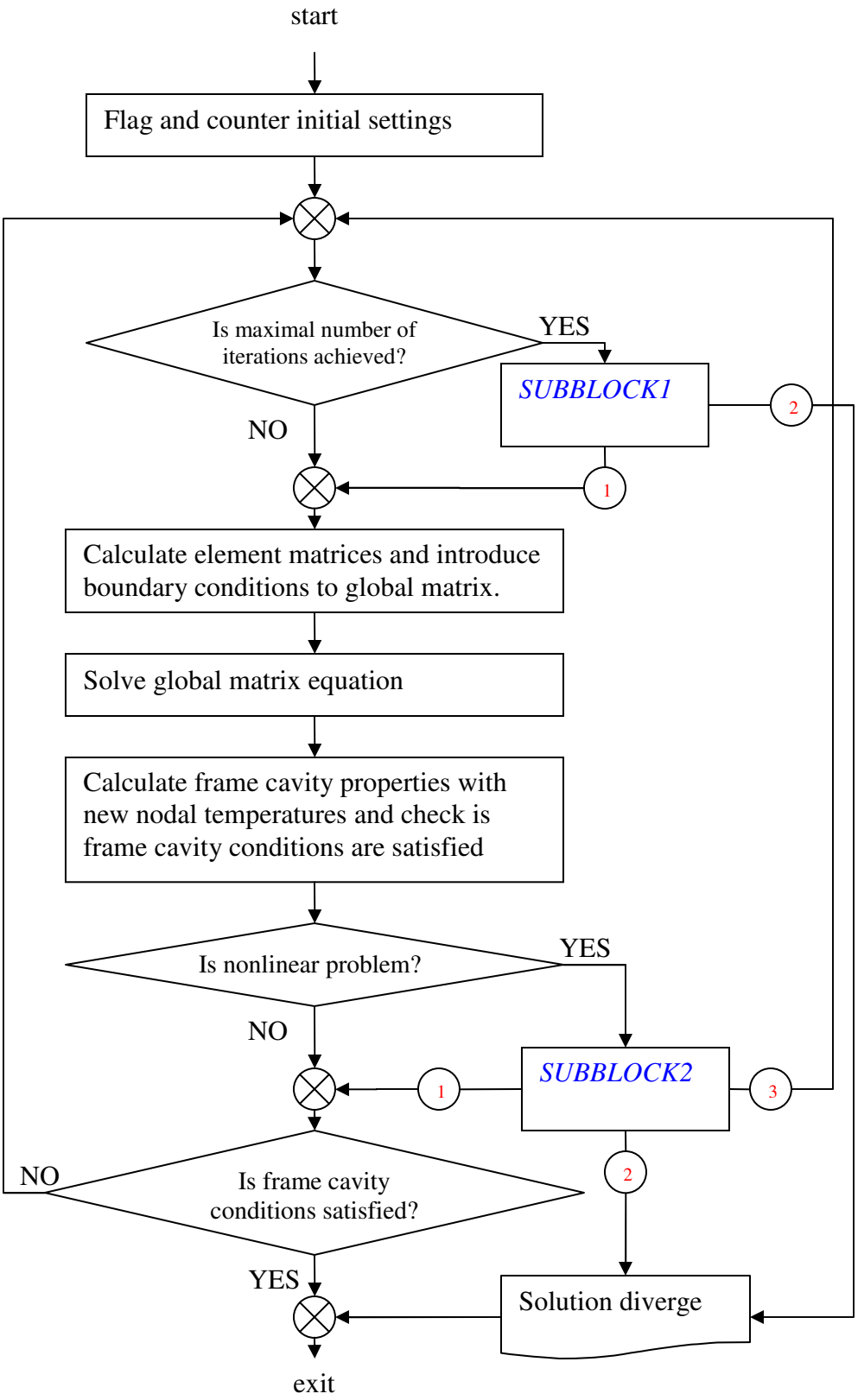


Figure 4-3: Program Flow for Routine Steady

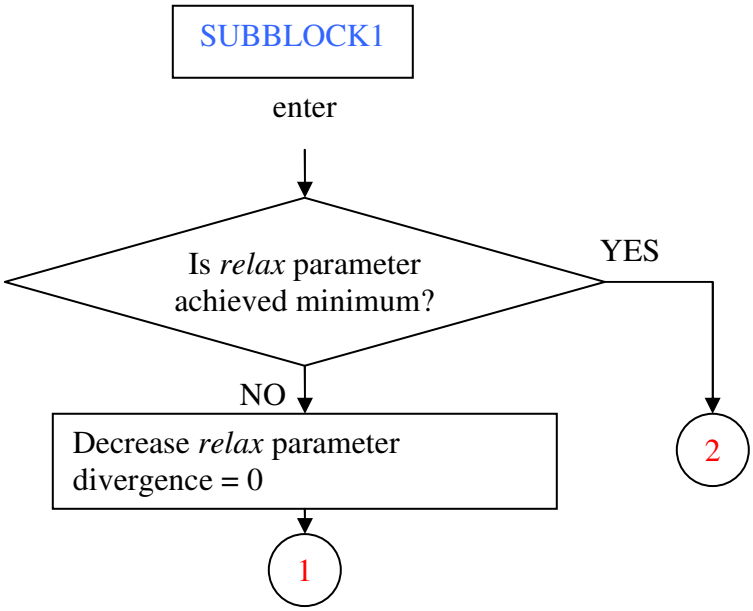


Figure 4-4: Program Flow for SUBBLOCK1 (Routine STEADY)

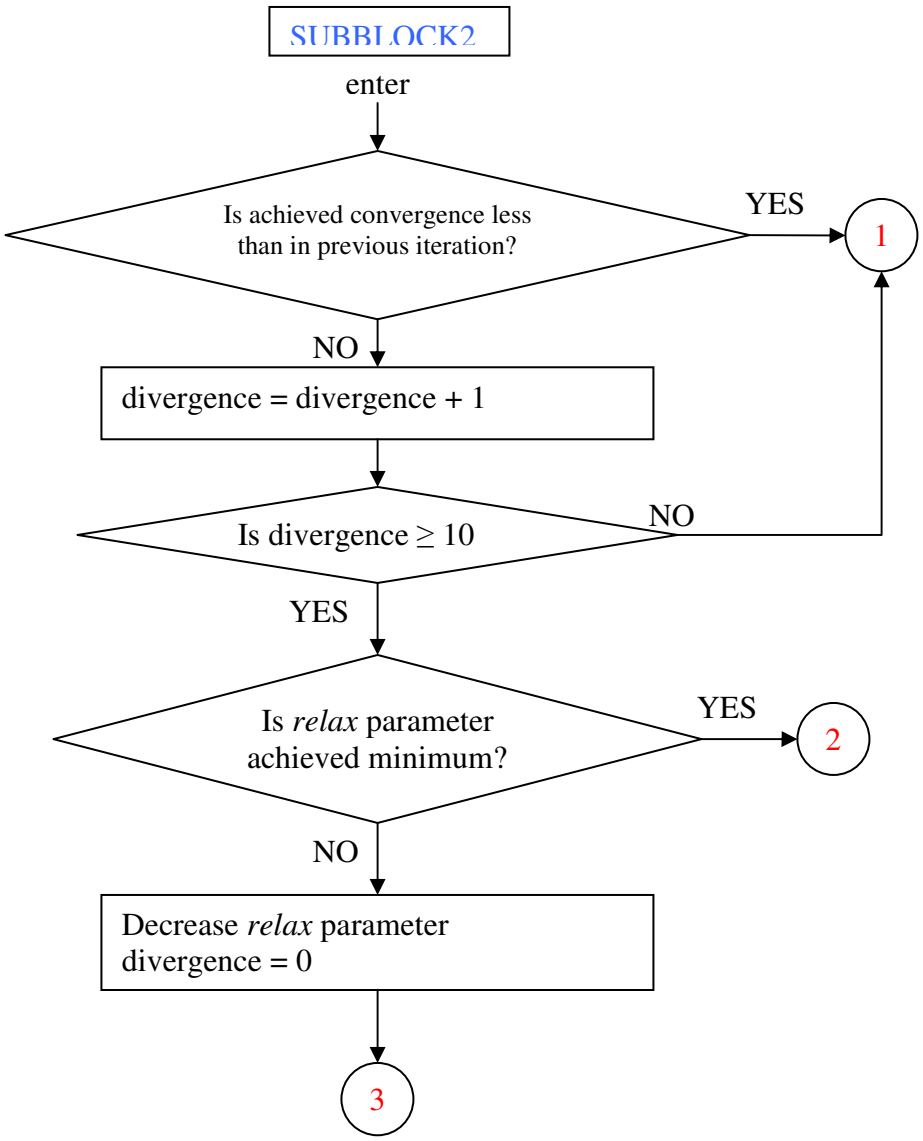


Figure 4-5: Program Flow for SUBBLOCK2 (Routine STEADY)

4.4. Routine BASIS

Routine **BASIS** calculates shape functions and derivatives for master rectangular element (Eq. 2.7-2) in Gauss point integration (Table 2.15-1).

List of arguments:

There is no list of arguments for this routine

Results are stored in following arrays:

h(i,j), dhdz (i,j) and dhde(i,j) (common block) where

i-number of Gauss point integration (for example i=1 mean first gauss point, i=2 means second gauss point etc.)

j-function number

For example:

$$h(2,3)=\tilde{\psi}_3^e(\xi_2,\eta_2)=\tilde{\psi}_3^e(g2) \qquad \xi_2=\frac{1}{\sqrt{3}},\ \eta_2=-\frac{1}{\sqrt{3}} \qquad \textit{Eq. 4.4-1}$$

$$dhdz(1,3)=\frac{\partial\tilde{\psi}_3^e(\xi_1,\eta_1)}{\partial\xi}=\frac{\partial\tilde{\psi}_3^e(g1)}{\partial\xi} \qquad \xi_1=-\frac{1}{\sqrt{3}},\ \eta_1=-\frac{1}{\sqrt{3}} \qquad \textit{Eq. 4.4-2}$$

$$dhde(2,4)=\frac{\partial\tilde{\psi}_4^e(\xi_2,\eta_2)}{\partial\eta}=\frac{\partial\tilde{\psi}_4^e(g2)}{\partial\eta} \qquad \xi_2=\frac{1}{\sqrt{3}},\ \eta_2=-\frac{1}{\sqrt{3}} \qquad \textit{Eq. 4.4-3}$$

Results are stored in common block /**blk11**/.

4.5. Routine SHAPE

Routine **SHAPE** calculates Jacobian matrix and determinant, and function derivatives in global coordinates (Eq. 2.10-10) for specified gauss point () for one element.

List of arguments:

ig – *(input value)* Gauss point number

ex – *(input value)* Vector of node coordinates of element

det – *(output value)* Jacobian determinant

sh – *(output value)* Vector of function derivatives in global coordinates

To shape function derivatives in global coordinates equations (Eq. 2.10-2), (Eq. 2.10-5) and (Eq. 2.10-10) are used. Jacobian matrix is stored in:

$$\begin{Vmatrix} xs(1,1) & xs(1,2) \\ xs(2,1) & xs(2,2) \end{Vmatrix} = J \qquad \textit{Eq. 4.5-1}$$

and

$$\begin{Vmatrix} sh(1,i) \\ sh(2,i) \end{Vmatrix} = \begin{Vmatrix} \dfrac{\partial\psi_i^e(x_{ig},y_{ig})}{\partial x} \\ \dfrac{\partial\psi_i^e(x_{ig},y_{ig})}{\partial y} \end{Vmatrix} \qquad \textit{Eq. 4.5-2}$$

where $\psi_i^e(x_{ig},y_{ig})$ is global element function in ig-th gauss point.

4.6. Routine SHAPEV

Routine **SHAPEV** calculates Jacobian matrix and determinant, and function derivatives in global coordinates (Eq. 2.10-10) for specified gauss point () for group of elements.

List of arguments:

ig – *(input value)* Gauss point number

llt – *(input value)* Number of elements

Results are also stored in common blocks:

/**v01**/ - vectors for Jacobian determinant, functions and function derivatives

/**v02**/ - vectors for node coordinates of elements

To shape function derivatives in global coordinates equations (Eq. 2.10-2), (Eq. 2.10-5) and (Eq. 2.10-10) are used. Jacobian matrix for i-th element is stored in:

$$\begin{Vmatrix} xs11(i) & xs12(i) \\ xs21(i) & xs22(i) \end{Vmatrix} = J \qquad \textit{Eq. 4.6-1}$$

and function derivatives in global coordinate system for i-th element

$$\begin{Vmatrix} sh1k(i) \\ sh2k(i) \end{Vmatrix} = \begin{Vmatrix} \dfrac{\partial \psi_k^e(x_{ig}, y_{ig})}{\partial x} \\ \dfrac{\partial \psi_k^e(x_{ig}, y_{ig})}{\partial y} \end{Vmatrix} \qquad \textit{Eq. 4.6-2}$$

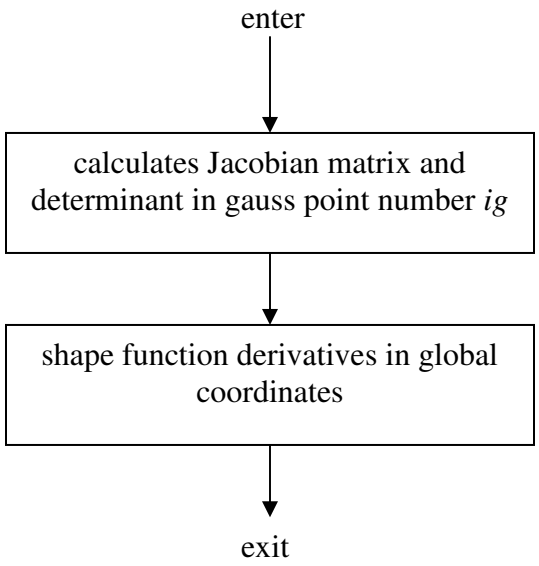


Figure 4-6: Program Flow for Routines SHAPE and SHAPEV

4.7. Routine FORMKF

Routine **FORMKF** calculates conduction matrix and internal heat generation matrix (equations 2.7.6 and 2.7.13) for all elements and assemble it in global matrix. Routine **FORMKF** also uses results evaluated in routine SHAPEV (common blocks /v01/ and /v02/).

List of arguments:

x – *(input vector)* Vector of node coordinates

km – *(input vector)* Vector of element data

angl – *(input vector)* Future use

mtype – *(input vector)* Vector of material types

ptemp – *(*

cv – *(input vector)* Vector for material heat capacity

cond1 – *(input vector)* Vector for thermal conductivity (=k for isotropic, =k11 for orthotropic materials)

cond2 – *(input vector)* Future use (=k22 for orthotropic materials)

ncgenm – *(input vector)* Future use. Vector of thermal generation rate curve numbers

genm – *(input vector)* Future use. Thermal generation rate (multiplier in future use with ncgenm)

ncgene – *(input vector)* Future use

gene – *(input vector)* Future use

dqgen - ?

curvx – *(input vector)* Future use. X-coordinates of functions data

curvy – *(input vector)* Future use. Y-coordinates of functions data

npc – *(input vector)* Future use. Number of points for curve

tmpc – *(*

tmpk – *(*

cvtr – *(*

cntr – *(*

jdiag – *(*

tn – *(input vector)* Node temperatures (or temperature difference for nonlinear problems) from previous iteration

tnp – *(input vector)* Node temperature derivatives. CHECK THIS

gf –

gk –

au –

ad –

nonl – *(input value)* =0 linear problem; =1 nonlinear problem

rhoelm –

Routine **FORMKF** uses common blocks /v01/ and /v02/ which are calculated in routine SHAPEV.

Conduction matrix (Eq. 2.9-16) is solved using function derivatives and Jacobian determinants calculated in routine SHAPEV (stored in common block /v01/) and material properties which are calculated in routine MATL. Conduction matrix is calculated in four Gauss points and stored by following equation (for i-th element):

$$\begin{Vmatrix} ek(1,i) & ek(2,i) & ek(4,i) & ek(7,i) \\ ek(2,i) & ek(3,i) & ek(5,i) & ek(8,i) \\ ek(4,i) & ek(5,i) & ek(6,i) & ek(9,i) \\ ek(7,i) & ek(8,i) & ek(9,i) & ek(10,i) \end{Vmatrix} = \begin{Vmatrix} K_{11} & K_{12} & K_{13} & K_{14} \\ K_{21} & K_{22} & K_{23} & K_{24} \\ K_{31} & K_{32} & K_{33} & K_{34} \\ K_{41} & K_{42} & K_{43} & K_{44} \end{Vmatrix} \qquad \textit{Eq. 4.7-1}$$

where K_{ij} is calculated by equations (Eq. 2.9-16) and (Eq. 2.15-1) in Gauss integration points. Internal heat generation matrix (Eq. 2.9-8) is also calculated in four Gauss point and stored by following equation (for i-th element):

$$\begin{Vmatrix} ef(1,i) \\ ef(2,i) \\ ef(3,i) \\ ef(4,i) \end{Vmatrix} = \begin{Vmatrix} Q_1 \\ Q_2 \\ Q_3 \\ Q_4 \end{Vmatrix} \qquad \textit{Eq. 4.7-2}$$

where Q_{ij} is calculated by equations (Eq. 2.9-8) and (Eq. 2.15-1) in Gauss integration points. Note that equations (Eq. 4.7-1) and (Eq. 4.7-2) are calculates for both (linear and nonlinear) types of problem. For linear problems equations (Eq. 4.7-1) and (Eq. 4.7-2) are assembled by equation (Eq. 2.9-6) into the global arrays. Each element of matrix **ek** and vector **ef** is assembled to corresponding element of global arrays which depends of node number. Matrix **ek** is assembled to left the left side of global matrix (Eq. 2.11-9) and vector **ef** is assembled to the right side of global matrix (Eq. 2.11-9).

For nonlinear problems one more matrix is calculated:

$$\begin{Vmatrix} egt(1,i) & egt(2,i) & egt(4,i) & egt(7,i) \\ egt(2,i) & egt(3,i) & egt(5,i) & egt(8,i) \\ egt(4,i) & egt(5,i) & egt(6,i) & egt(9,i) \\ egt(7,i) & egt(8,i) & egt(9,i) & egt(10,i) \end{Vmatrix} = \begin{Vmatrix} Q_{11}' & Q_{12}' & Q_{13}' & Q_{14}' \\ Q_{21}' & Q_{22}' & Q_{23}' & Q_{24}' \\ Q_{31}' & Q_{32}' & Q_{33}' & Q_{34}' \\ Q_{41}' & Q_{42}' & Q_{43}' & Q_{44}' \end{Vmatrix} \qquad \textit{Eq. 4.7-3}$$

where Q_{ij}' is calculated by equation (Eq. 2.9-14) and (Eq. 2.15-1) in Gauss integration points. For nonlinear problems equations (Eq. 4.7-1), (Eq. 4.7-2) and (Eq. 4.7-3) are assembled by equation (Eq. 2.9-13) into the global arrays.

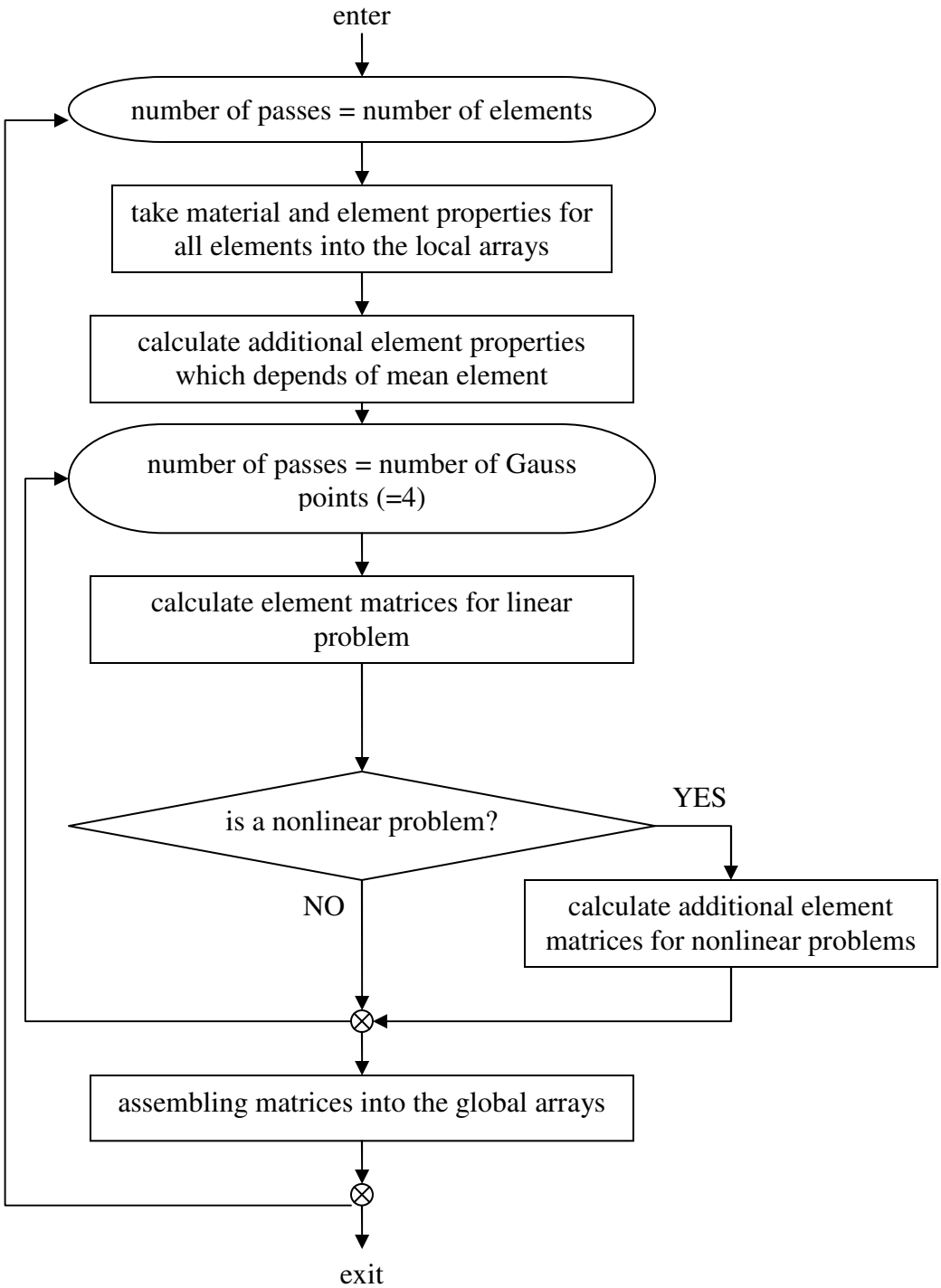


Figure 4-7: Program Flow for Routine FORMKF

4.8. Routine BCCONV

Routine **BCCONV** calculates convection boundary condition using equations (Eq. 2.12-3) for linear problems and equation (Eq. 2.12-14) for nonlinear problems. Integrate these equations in two Gauss points (for line elements) and finally assembling all coefficients into the global arrays.

List of arguments:

x – *(input vector)* Array of nodal coordinates

ndbc – *(input vector)* Node numbers of which is segment consist

EXAMPLE: ndbc(1,1,nseg) – first node of nsegth segment

ndbc(1,2,nseg) – second node of nsegth segment

nctinf – *(input vector)* Curve numbers

tinfm – *(input vector)* Outside film temperatures at segment

EXAMPLE: tinfm(1,nseg) – outside temperature at first node of nsegth segment

tinfm(2,nseg) – outside temperature at second node of nsegth segment

nch – *(input vector)* Curve numbers

hm – *(input vector)* Segment film coefficients

EXAMPLE: hm(nseg) – film coefficient at nsegth segment

freex – (input vector)

curvx – *(input vector)* x-axe values of curves

curvy – *(input vector)* y-axe values of curves

npc – *(input vector)* Number of points for curve

jdiag – (input vector)

tn – *(input vector)* Node temperatures (or temperature difference) from current iteration

tnp – *(input vector)* Temperature derivatives. CHECK THIS

gf – *(output vector)* Used for assembling right-hand side of equations

nctcbc – *(input vector)* Curve numbers

gk – (output vector)

au – (output vector)

ad – (output vector)

ncbc – number of Convection Boundary Condition segments

nonl – type of problem (0=linear; 1=nonlinear)

igeom – type of geometry

Matrix A (Eq. 2.12-4) is calculated for linear and nonlinear types of problem and results are stored in:

$$\begin{Vmatrix} bckl(1,l) & bckl(2,l) \\ bckl(2,l) & bckl(3,l) \end{Vmatrix} = \begin{Vmatrix} A_{11}^e & A_{12}^e \\ A_{21}^e & A_{22}^e \end{Vmatrix} \qquad \textbf{Eq. 4.8-1}$$

where l=1 (always) and coefficients A_{ij}^e are calculated by equation (Eq. 2.12-4). Matrix B (Eq. 2.12-5) is also calculated for linear and nonlinear types of problem and results are stored:

$$\begin{Vmatrix} bcf(1,l) \\ bcf(2,l) \end{Vmatrix} = \begin{Vmatrix} B_1^e \\ B_2^e \end{Vmatrix} \qquad \textbf{Eq. 4.8-2}$$

where l=1 (always) and coefficients B_i^e are calculated by equation (Eq. 2.12-5). If problem type is linear, assembling matrices A and B is done by equation (Eq. 2.12-3) or in matrix notation (Eq. 2.12-6).

For nonlinear type of problem there is three additional matrices which are calculated. First matrix is:

$$\begin{Vmatrix} bckn(1,l) & bckn(2,l) \\ bckn(2,l) & bckn(3,l) \end{Vmatrix} = \begin{Vmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{Vmatrix} \qquad \textbf{Eq. 4.8-3}$$

where l=1 (always) and coefficients C_{ij} are calculated by equation (Eq. 2.12-11).

Second matrix is:

$$\begin{Vmatrix} bcknp(1,l) & bcknp(2,l) \\ bcknp(2,l) & bcknp(3,l) \end{Vmatrix} = \begin{Vmatrix} D_{11} & D_{12} \\ D_{21} & D_{22} \end{Vmatrix} \qquad \textbf{Eq. 4.8-4}$$

where l=1 (always) and coefficients D_{ij} are calculated by equation (Eq. 2.12-12).

Third matrix is:

$$\begin{Vmatrix} bckf(1,l) & bckf(2,l) \\ bckf(2,l) & bckf(3,l) \end{Vmatrix} = \begin{Vmatrix} E_{11} & E_{12} \\ E_{21} & E_{22} \end{Vmatrix} \qquad \textbf{Eq. 4.8-5}$$

where l=1 (always) and coefficients E_{ij} are calculated by equation (Eq. 2.12-13).

For nonlinear type of problem, matrices are assembled by equation (Eq. 2.12-14).

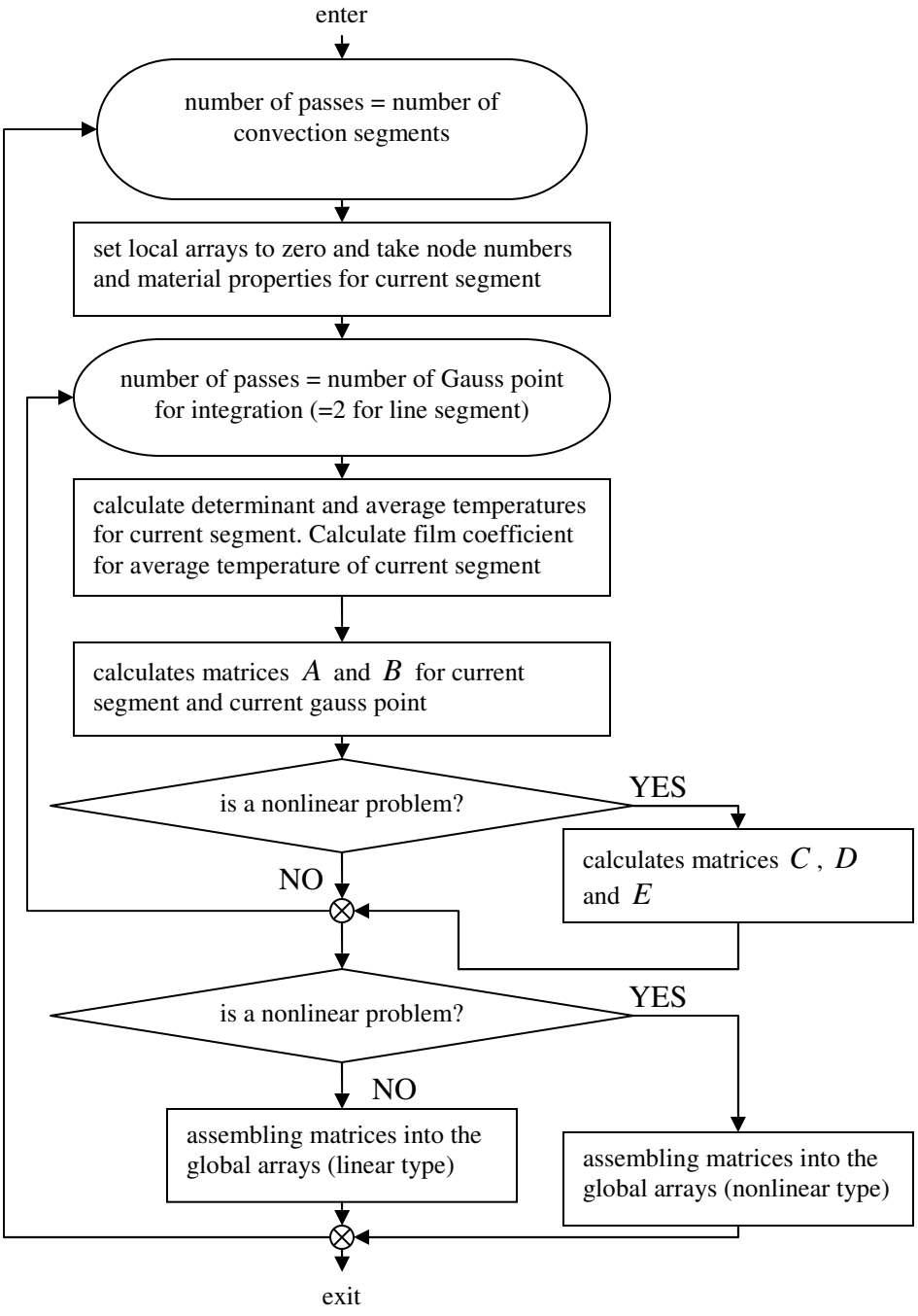


Figure 4-8: Program Flow for Routine BCCONV

4.9. Routine BCFLUX

Routine **BCFLUX** calculates flux boundary condition using equations (Eq. 2.12-15) for linear problems and equation (Eq. 2.12-19) for nonlinear problems. Integrate these equations in two Gauss points (for line elements) and finally assembling all coefficients into the global arrays.

List of arguments:

x – *(input vector)* Array of nodal coordinates

ndbc – *(input vector)* Node numbers of which is segment consist

EXAMPLE: ndbc(1,1,nseg) – first node of nsegth segment

ndbc(1,2,nseg) – second node of nsegth segment

ncf – (

fbcm – (

curvx – *(input vector)* x-axe values of curves

curvy – *(input vector)* y-axe values of curves

npc – *(input vector)* Number of points for curve

jdiag – (input vector)

tn – *(input vector)* Node temperatures (or temperature difference) from current iteration

tnp – *(input vector)* Temperature derivatives. CHECK THIS

gf – *(output vector)* Used for assembling right-hand side of equations

nctfbc – *(input vector)* Curve numbers

gk – (output vector)

au – (output vector)

ad – (output vector)

nbc – number of Flux Boundary Condition segments

nonl – type of problem (0=linear; 1=nonlinear)

igeom – type of geometry

Matrix Q (Eq. 2.12-16) is calculated for linear and nonlinear types of problem and results are stored in:

$$\begin{Vmatrix} bcf(1,l) \\ bcf(2,l) \end{Vmatrix} = \begin{Vmatrix} Q_1 \\ Q_2 \end{Vmatrix} \qquad \textit{Eq. 4.9-1}$$

where l=1(always) and coefficients Q_i are calculated by Eq. 2.12-16. Eq. 4.9-1 are assembled into the right-hand side of global matrices.

For nonlinerar type of problem there is additional matrix which is calculated:

$$\begin{Vmatrix} bckf(1,l) & bckf(2,l) \\ bckf(2,l) & bckf(3,l) \end{Vmatrix} = \begin{Vmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{Vmatrix} \qquad \textit{Eq. 4.9-2}$$

where l=1 (always) and coefficients F_{ij} are calculated by Eq. 2.12-22. For nonlinear type of problem, assembling is done according to Eq. 2.12-21.

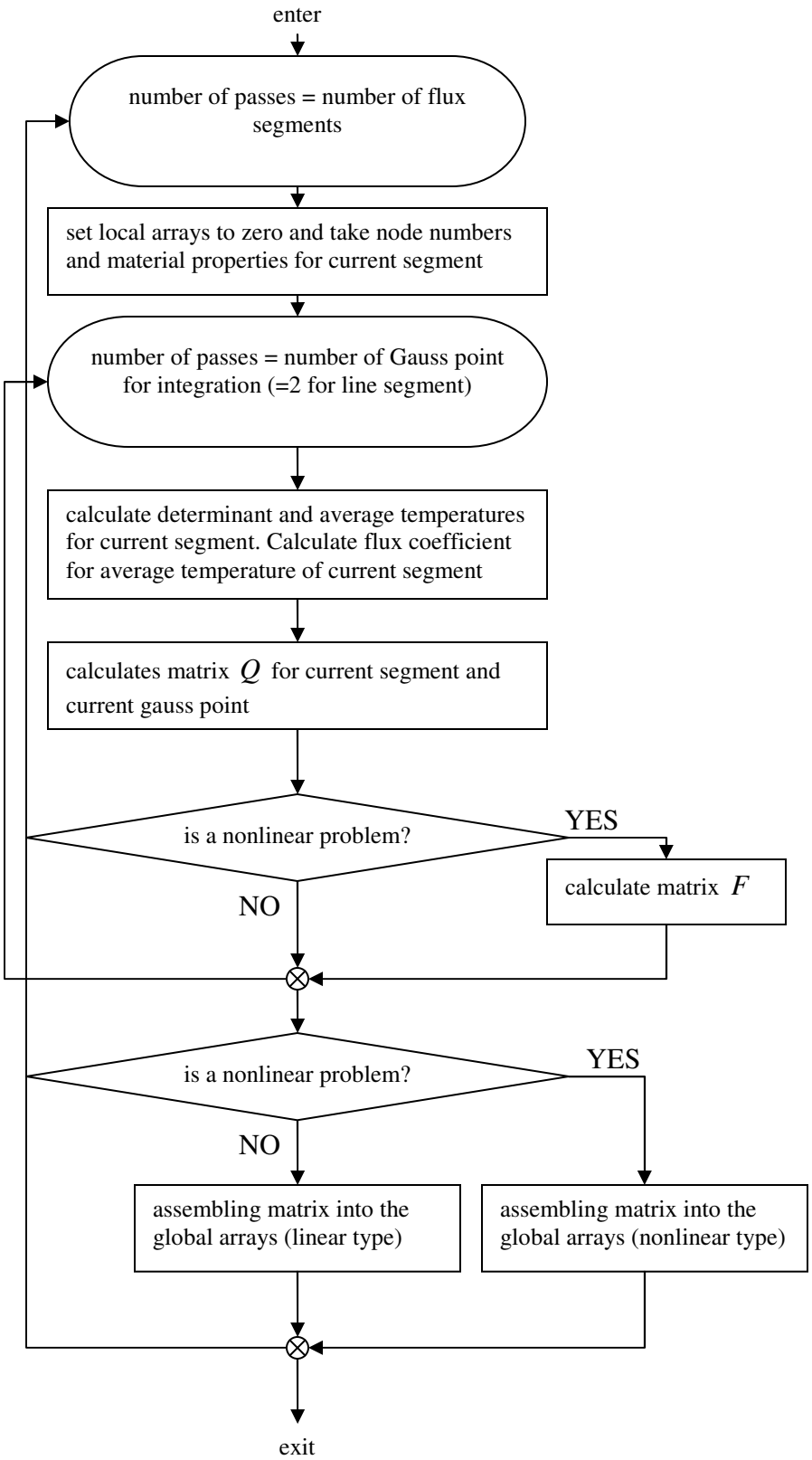


Figure 4-9: Program Flow for Routine BCCONV

4.10. Routine BCRAD1

Routine **BCRAD1** calculates black body boundary condition using linearized equation (Eq. 2.12-23) for nonlinear problems (radiation makes problem nonlinear) and integrate these equations in two Gauss points (for line elements) and finally assembling all coefficients into the global arrays.

List of arguments:

x – *(input vector)* Array of nodal coordinates

ndbc – *(input vector)* Node numbers of which is segment consist

EXAMPLE: ndbc(1,1,nseg) – first node of nsegth segment

ndbc(1,2,nseg) – second node of nsegth segment

nctinf – *(input vector)* Curve numbers – for temperature multiplier

tinfm – *(input vector)* Outside film temperatures at segment

EXAMPLE: tinfm(1,nseg) – outside temperature at first node of nsegth segment

tinfm(2,nseg) – outside temperature at second node of nsegth segment

nch – *(input vector)* Curve numbers – for linearized film coeficient

hm – *(input vector)* equal with $\sigma\varepsilon_{nseg}$ (ε_{nseg} = segment emissivity)

curvx – *(input vector)* x-axe values of curves

curvy – *(input vector)* y-axe values of curves

npc – *(input vector)* Number of points for curve

jdiag – (input vector)

tn – *(input vector)* Node temperatures (or temperature difference) from current iteration

tnp – *(input vector)* Temperature derivatives. CHECK THIS

gf – *(output vector)* Used for assembling right-hand side of equations

nctrbc – *(input vector)* Curve numbers

gk – (output vector)

au – (output vector)

ad – (output vector)

nbc – number of Black Body Radiation Boundary Condition segments

igeom – type of geometry

Matrices calculated in routine BCRAD1 are stored in following arrays:

$$\begin{Vmatrix} bckl(1,l) & bckl(2,l) \\ bckl(2,l) & bckl(3,l) \end{Vmatrix} = \begin{Vmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{Vmatrix} \qquad \textit{Eq. 4.10-1}$$

where A_{ij} is calculated by equation (Eq. 2.12-28),

$$\begin{Vmatrix} bcf(1,l) \\ bcf(2,l) \end{Vmatrix} = \begin{Vmatrix} B_1 \\ B_2 \end{Vmatrix} \qquad \textit{Eq. 4.10-2}$$

where B_i is calculated by equation (Eq. 2.12-29),

$$\begin{Vmatrix} bckn(1,l) & bckn(2,l) \\ bckn(2,l) & bckn(3,l) \end{Vmatrix} = \begin{Vmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{Vmatrix} \qquad \textit{Eq. 4.10-3}$$

where C_{ij} is calculated by equation (Eq. 2.12-30),

$$\begin{Vmatrix} bcknp(1,l) & bcknp(2,l) \\ bcknp(2,l) & bcknp(3,l) \end{Vmatrix} = \begin{Vmatrix} D_{11} & D_{12} \\ D_{21} & D_{22} \end{Vmatrix} \qquad \textit{Eq. 4.10-4}$$

where D_{ij} is calculated by equation (Eq. 2.12-31), and finally

$$\begin{Vmatrix} bckf(1,l) & bckf(2,l) \\ bckf(2,l) & bckf(3,l) \end{Vmatrix} = \begin{Vmatrix} E_{11} & E_{12} \\ E_{21} & E_{22} \end{Vmatrix} \qquad \textit{Eq. 4.10-5}$$

where E_{ij} is calculated by equation (Eq. 2.12-32) and l=1 in all previous equations in this chapter. Assembling is done according to Eq. 2.12-33.

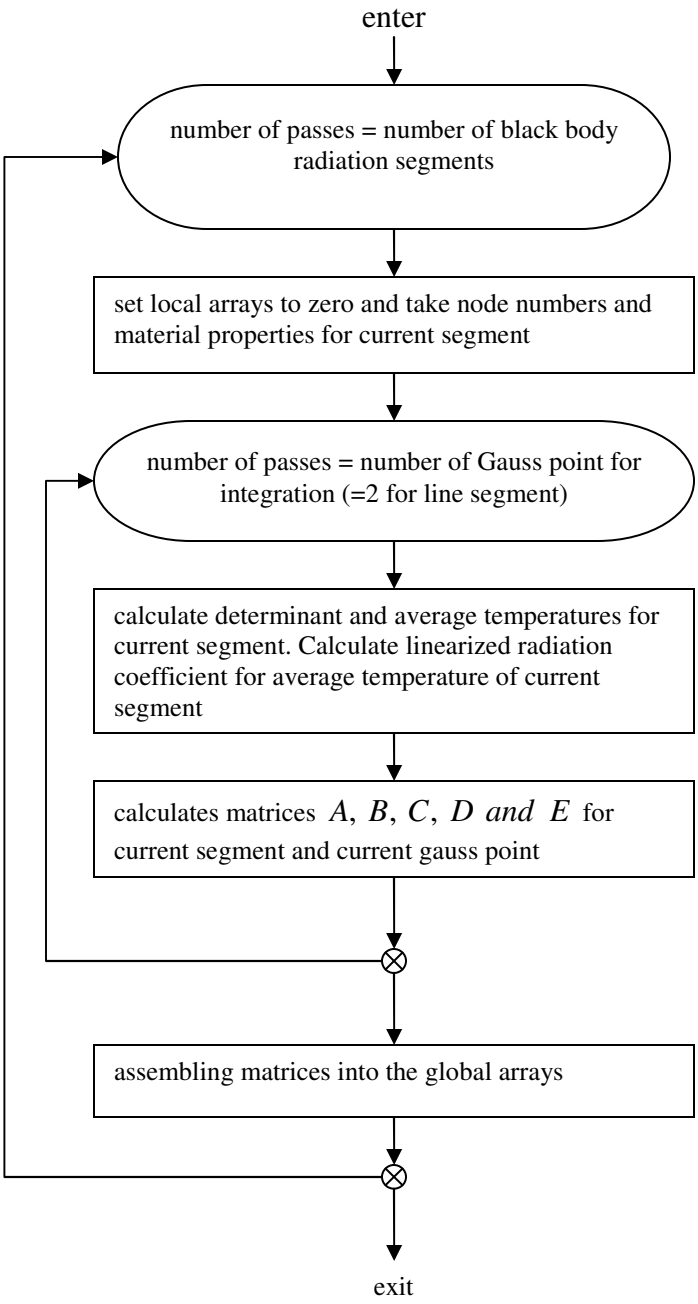


Figure 4-10: Program Flow for Routine BCRAD1

4.11. Routine RADIN2

Routine RADIN2 is input routine which get data for radiation enclosure boundary condition calculation. Routine RADIN2 also performed some calculations which are used in radiation enclosure boundary condition calculations.

List of arguments:

x – *(input vector)* Node coordinates

nodes – *(output vector)* Node numbers of radiation enclosure segments

nrcond – *(output vector)* Shows which part of radiation enclosure segment belongs
eq.0:segment is part of conduction el,
eq.1: segment is not part of conduction el

ncrad – Not used in this routine

thole – *(output vector)* Radiation enclosure surface temperature

emis – Not used in this routine

ipvt – *(output vector)* The pivot vector from sgeco or sgefa

work – *(input/output vector)* Working vector. Contents destroyed.

area – *(output vector)* Segment length

aef – *(input/output vector)* View factor matrix/Inverse AEF matrix

nband – Not used in this routine

necurv – Not used in this routine

sigma – Stefan-Boltzmann constant 5.6693×10^{-8} [W/(m^{2}K^4)]

irtype – =4 always

itmaxb – maximum number of radiosity iterations

tolb – radiosity convergence tolerance

igeom – type of geometry

nrdim – number of column in aef and afrow matrices Used only for definition aef and afrow!!!!

nebc – number of radiation enclosure surfaces

lcount – number of column in input text

longo – output type

labele – gray body radiation bc edge id

iconv – type of temperature scale
eq.1: Celsius (tscale='c' or 'C')
eq.2: Farenheit (tscale='f' or 'F')
eq.-1: Kelvin (tscale='k' or 'K')
eq.-2: Rankin (tscale='r' or'R')

emise – emissivity of the surface

iordr – reading flag

afrow – *(input vector)* View factor matrix CHECK THIS

Calculation which is performed in routine RADIN2 is to calculate inverse matrix $[AEF_{ij}]^{-1}$ according to equation (Eq. 2.12-35 and Eq. 2.12-36) and to store this matrix in:

$$\begin{Vmatrix} aef(1,1) & aef(1,2) & .... & aef(1,nebc) \\ aef(2,1) & aef(2,2) & .... & aef(2,nebc) \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ aef(nebd,1) & aef(nebc,2) & ... & aef(nebc,nebc) \end{Vmatrix} = \begin{Vmatrix} AEF_{11}^{-1} & AEF_{12}^{-1} & .... & AEF_{1,nebc}^{-1} \\ AEF_{21}^{-1} & AEF_{22}^{-1} & .... & AEF_{2,nebc}^{-1} \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ AEF_{nebd,1}^{-1} & AEF_{nebc,2}^{-1} & ... & AEF_{nebc,nebc}^{-1} \end{Vmatrix} = \quad \textit{Eq. 4.11-1}$$

$$= \begin{Vmatrix} AEF_{11} & AEF_{12} & .... & AEF_{1,nebc} \\ AEF_{21} & AEF_{22} & .... & AEF_{2,nebc} \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ AEF_{nebd,1} & AEF_{nebc,2} & ... & AEF_{nebc,nebc} \end{Vmatrix}^{-1} = [AEF_{ij}]^{-1}$$

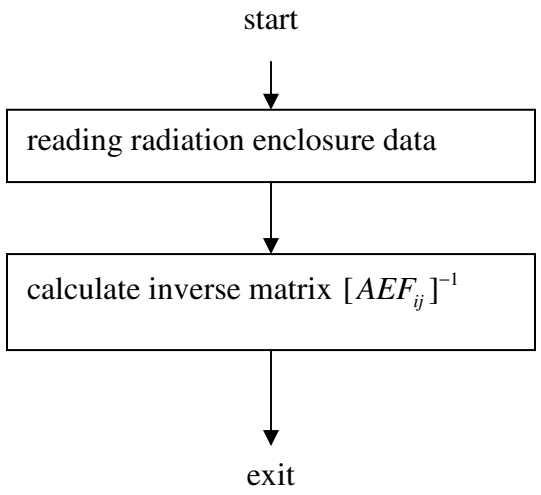


Figure 4-11: Program Flow for Routine RADIN2

4.12. Routine BCRAD2

Routine BCRAD2 calculates radiation enclosure (gray body) boundary conditions and assemble results into the global arrays.

List of arguments:

x – *(input vector)* Array of nodal coordinates

ndbc – *(input vector)* Node numbers of which is segment consist
EXAMPLE: ndbc(1,1,nseg) – first node of nsegth segment
ndbc(1,2,nseg) – second node of nsegth segment

ncrad – *(input vector)* Not used in this routine

thole – *(input vector)* Radiation enclosure surface temperature

emis – *(input vector)* Not used in this routine

qnet – *(input vector)* heat flow density of radiation enclosure segment

ts – *(input vector)* Segment temperature used for calculation

aef – *(input vector)* inverse AEF matrix calculated in routine RADIN2

b – *(output vector)* "radiation" matrix

jdiag – (input vector)

tn – *(input vector)* Node temperatures (or temperature difference) from current iteration

tnp – *(input vector)* Temperature derivatives. CHECK THIS

gf – *(output vector)* Used for assembling right-hand side of equations

gk – (output vector)

au – (output vector)

ad – (output vector)

nebc – number of Enclosure (Gray Body) Radiation Boundary Condition segments

igeom – type of geometry

sigma – Stefan-Boltzmann constant 5.6693×10^{-8} [W/(m^{2}K^4)]

nrdim – number of column in aef and afrow matrices CHECK THIS

emise – (input vector) Radiation enclosure surface emissivity

nrcond – *(input vector)* Shows which part of radiation enclosure segment belongs
eq.0:segment is part of conduction el,
eq.1: segment is not part of conduction el

Matrices calculated in routine BCRAD2 are stored in following arrays:

$$\begin{Vmatrix} b(1) \\ b(2) \\ \cdot \\ \cdot \\ \cdot \\ b(n) \end{Vmatrix} = \{K_j\}_{j=1,\ldots,n} = \begin{Vmatrix} K_1 \\ K_2 \\ \cdot \\ \cdot \\ \cdot \\ K_n \end{Vmatrix} = [AEF_{\substack{ij \\ i\neq j}}]^{-1} * \sigma * \{T^4_{\substack{iPREV \\ i\neq j}}\} \qquad \textit{Eq. 4.12-1}$$

which is equal with equation (Eq. 2.12-37).

Second matrix:

$$\begin{Vmatrix} bckl(1,l) & bckl(2,l) \\ bckl(2,l) & bckl(3,l) \end{Vmatrix} = \begin{Vmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{Vmatrix} \qquad \textit{Eq. 4.12-2}$$

where G_{ij} is calculated by equation (Eq. 2.12-50).

Third matrix:

$$\begin{Vmatrix} bcf(1,l) \\ bcf(2,l) \end{Vmatrix} = \begin{Vmatrix} N_1 \\ N_2 \end{Vmatrix} \qquad \textit{Eq. 4.12-3}$$

where N_i is calculated by equation (Eq. 2.12-57).

Fourth matrix:

$$\begin{Vmatrix} bckn(1,l) & bckn(2,l) \\ bckn(2,l) & bckn(3,l) \end{Vmatrix} = \begin{Vmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{Vmatrix} \qquad \textit{Eq. 4.12-4}$$

where H_{ij} is calculated by equation (Eq. 2.12-51).

Fifth matrix:

$$\begin{Vmatrix} bcknp(1,l) & bcknp(2,l) \\ bcknp(2,l) & bcknp(3,l) \end{Vmatrix} = \begin{Vmatrix} I_{11} & I_{12} \\ I_{21} & I_{22} \end{Vmatrix} \qquad \textit{Eq. 4.12-5}$$

where I_{ij} is calculated by equation (Eq. 2.12-52).

Sixth matrix:

$$\begin{Vmatrix} bckf(1,l) & bckf(2,l) \\ bckf(2,l) & bckf(3,l) \end{Vmatrix} = \begin{Vmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{Vmatrix} \qquad \textit{Eq. 4.12-6}$$

where M_{ij} is calculated by equation (Eq. 2.12-56), and l=1 (always) in all previous equations.

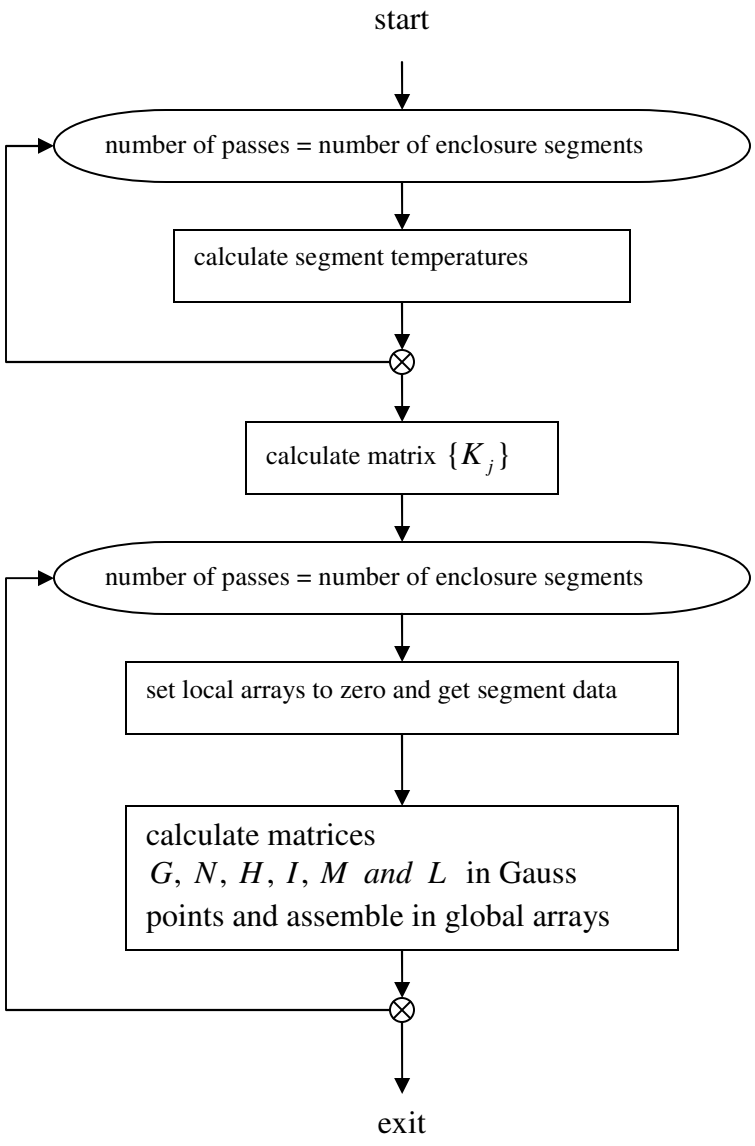


Figure 4-12: Program Flow for Routine BCRAD2

4.13. Routine BCTEMP1

Routine BCTEMP1 introduces temperature boundary conditions and assemble it into the global arrays. Assembling into the global matrices (Eq. 2.11-9 and Eq. 2.11-10) for i^{th} node (at which temperature is defined and equal T_{KNOWN}) is working according to next equations (for i^{th} node):

4.13.1. Linear Problem

$$K_{i,1} * T_1 + K_{2,i} * T_2 + ... + K_{i,i} * T_i + ... + K_{n,i} * T_n = P_i \qquad \textit{Eq. 4.13-1}$$

and after assembling (in BCTEMP1) temperature for i[th] node equation (Eq. 4.13-1) becomes:

$$K_{i,1} * T_1 + K_{i,2} * T_2 + ... + 1e18 * T_i + ... + K_{i,n} * T_n = 1e18 * T_{KNOWN} \qquad \textit{Eq. 4.13-2}$$

which lead that in global matrix solution:

$$T_i = T_{KNOWN} \qquad \textit{Eq. 4.13-3}$$

because following equation is satisfied:

$$K_{i,1},\ K_{i,2,....,}\ K_{i,n} << 1e18 \qquad \textit{Eq. 4.13-4}$$

4.13.2. Nonlinear Problem

$$K_{i,1} * \Delta T_1 + K_{2,i} * \Delta T_2 + ... + K_{i,i} * \Delta T_i + ... + K_{n,i} * \Delta T_n = P_i \qquad \textit{Eq. 4.13-5}$$

and after assembling (in BCTEMP1) temperature for i[th] node equation (Eq. 4.13-1) becomes:

$$\Delta T_i = 0 \qquad \textit{Eq. 4.13-6}$$

because temperature difference for node at which is temperature defined is zero.

List of arguments:

ndbc – (input vector) Segment (Node) at which temperature is defined

ncbc – (input vector) Curve number

tbcm – (input vector) Segment (node) temperatures

curvx – (input vector) x-axe values of curves

curvy – (input vector) y-axe values of curves

npc – (input vector) Number of points for curve

jdiag –

gf –

gk –

au –

ad –

ntbc – Number of segments (Nodes) with temperature bc

nonl – type of problem (0=linear; 1=nonlinear)

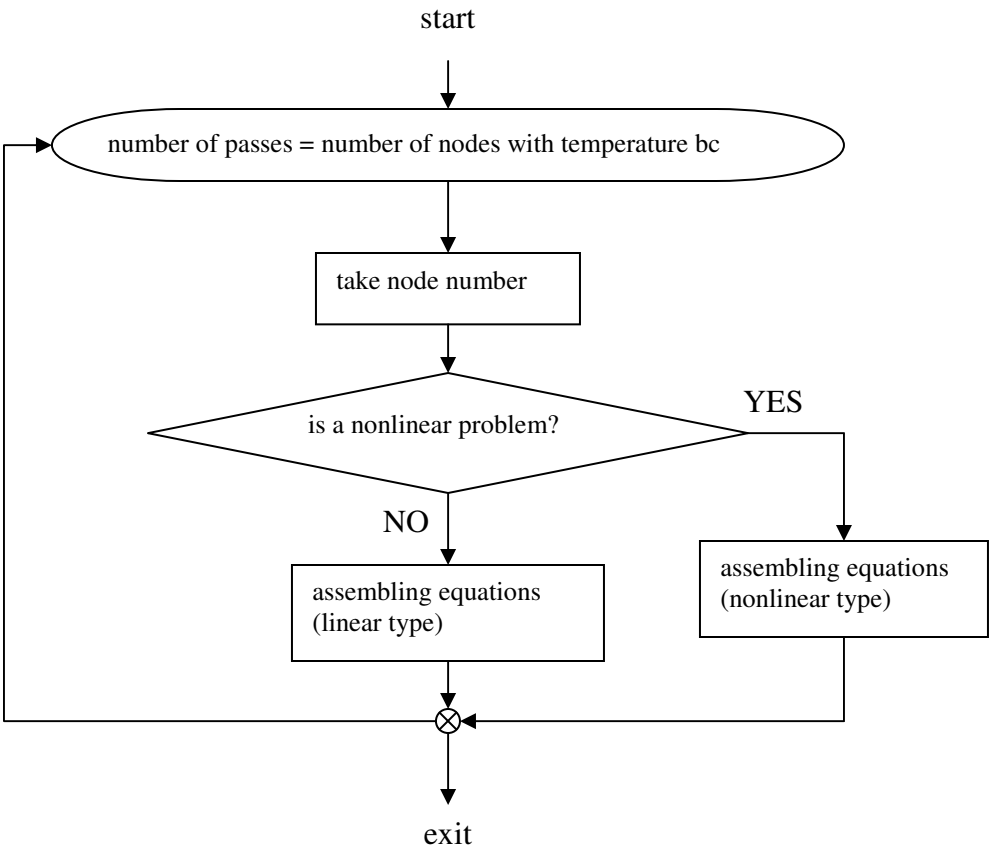


Figure 4-13: Program Flow for Routine BCTEMP1

4.14. Routine VARH

Routine VARH calculates convective and radiative part of surfaces which have interaction with gasses (or gas mixtures), like surfaces inside IGU. Equations used in routine VARH are described in section 1.3.3.4.

List of arguments:

wl – *(input value)* Glazing cavity (IGU) width

x – *(input value)* Segment distance from Starting (Departing) Corner

t1 – *(input value)* Temperature of side number 1

t2 – *(input value)* Temperature of side number 2

e1 – *(input value)* Emissivity of side number 1

e2 – *(input value)* Emissivity of side number 2

is – *(input value)* Flag for cavity orientation

eq.1 – Frame Cavity orientation DOWN

eq.-1 – Frame Cavity orientation UP

height – *(input value)* Height of IGU

cond – *(output value)* Thermal conductivity (gas property)

dvisc – *(output value)* Dynamic viscosity (gas property)

rho – *(output value)* Density (gas property)

cp – *(output value)* Specific heat (gas property)

pr – *(output value)* Prandtl number (gas property)

tm – *(input value)* Gas mean temperature

sigma – *(input value)* Stefan-Boltzmann constant 5.6693×10^{-8} [W/(m^{2}K^4)]

grav – *(input value)* Gravity acceleration constant 9.81 [m/s^2]

h – *(output value)* Film coefficient of segment

icrrad – *(input value)* radiation flag (eq.1 – include radiation; eq.0 – omit radiation)

tscale – *(input value)* Temperature scale

tscale='c' or 'C' - Celsius
tscale='f' or 'F' - Fahrenheit
tscale='k' or 'K' - Kelvin
tscale='r' or 'R' - Rankin

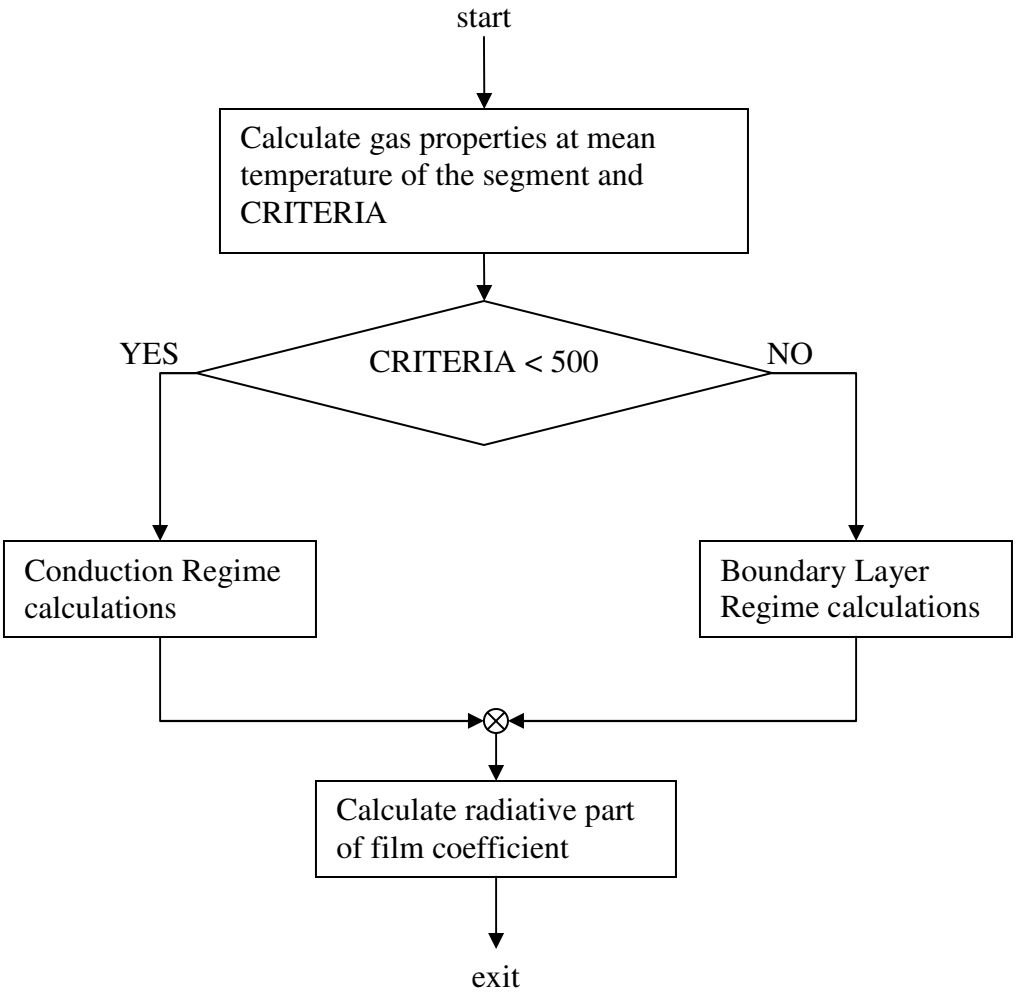


Figure 4-14: Program Flow for Routine VARH

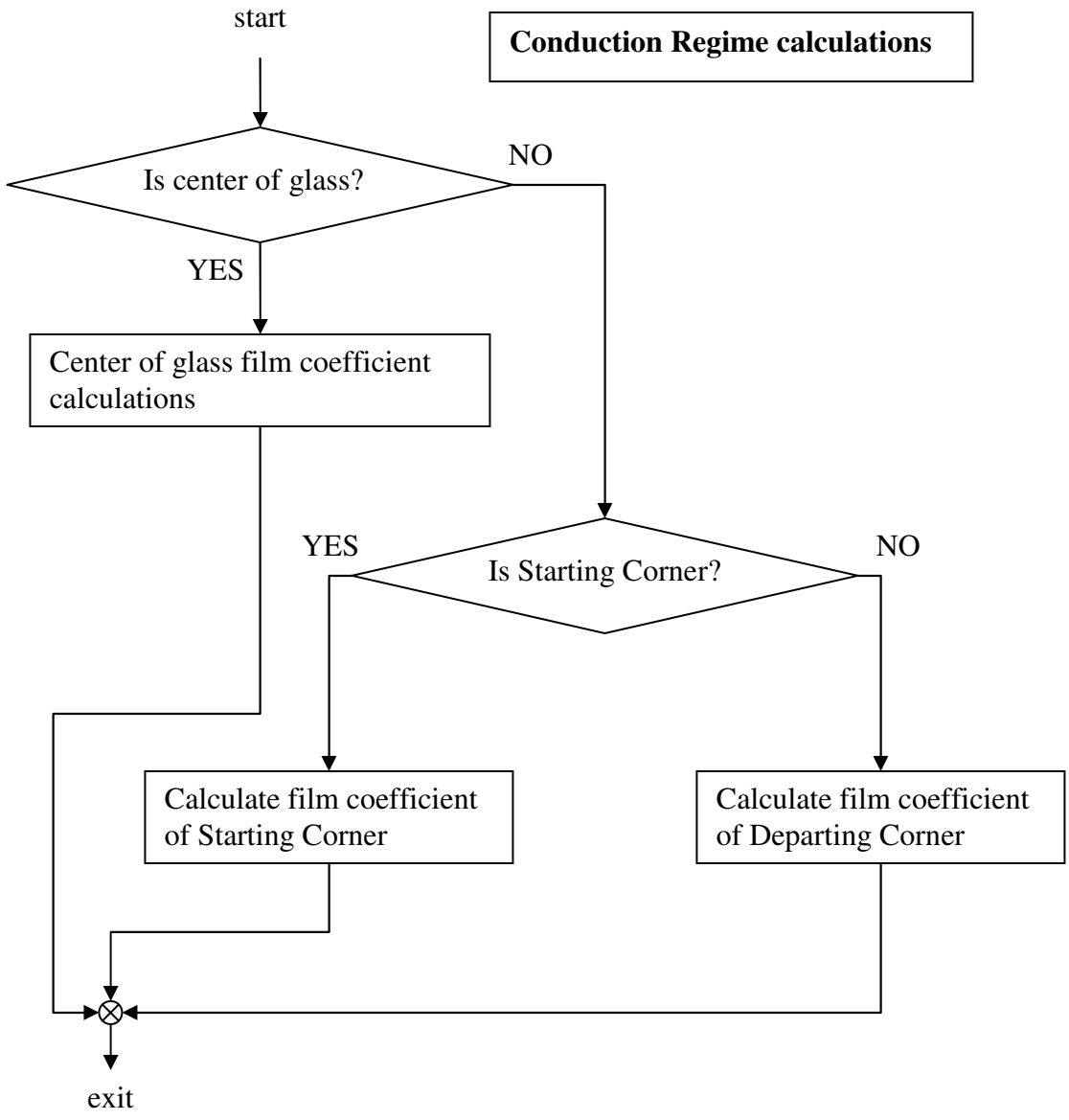


Figure 4-15: Conduction Regime Calculation Program Flow

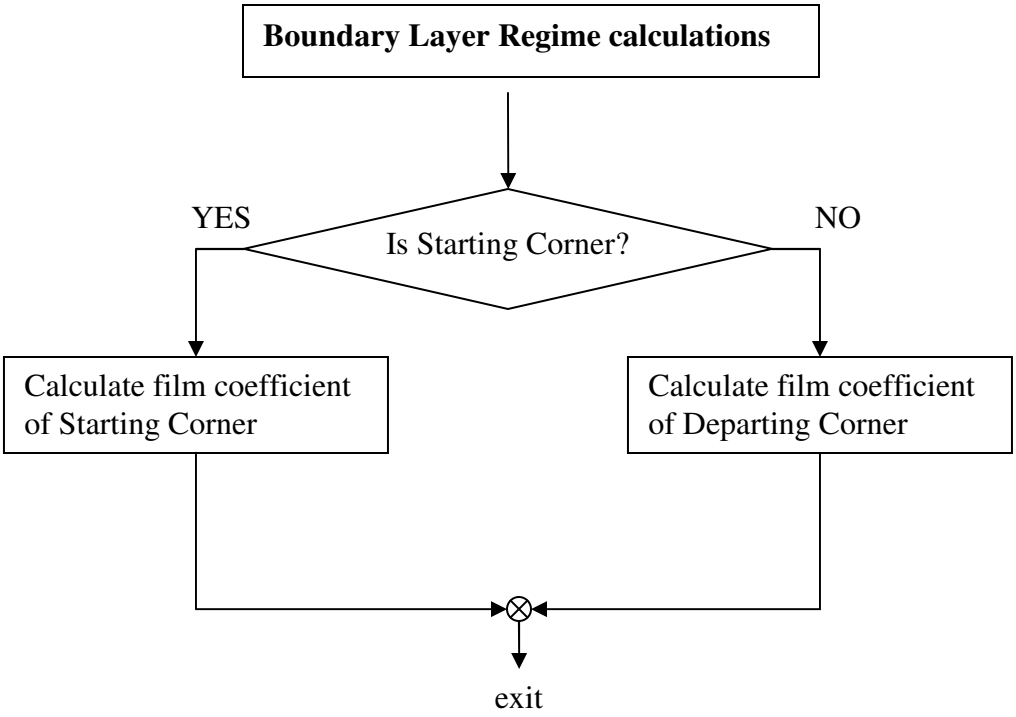


Figure 4-16: Boundary Layer Regime Calculation Program Flow

4.15. Routine BANDW

Routine BANDW implement bandwidth minimization described in 3.1. This routine determine does minimization take effect or not and if there is any effect. If there is effect then this routine determines two vectors which is used for renumbering.

List of arguments:

nrv – *(output vector)* nodal reorder vector – transformation from reorder to original numbering system

id – *(output vector)* inverse nodal reorder vector – transformation from original to reorder numbering system

lcount – *(input value)* number of column in input text

longo – *(input value)* output type

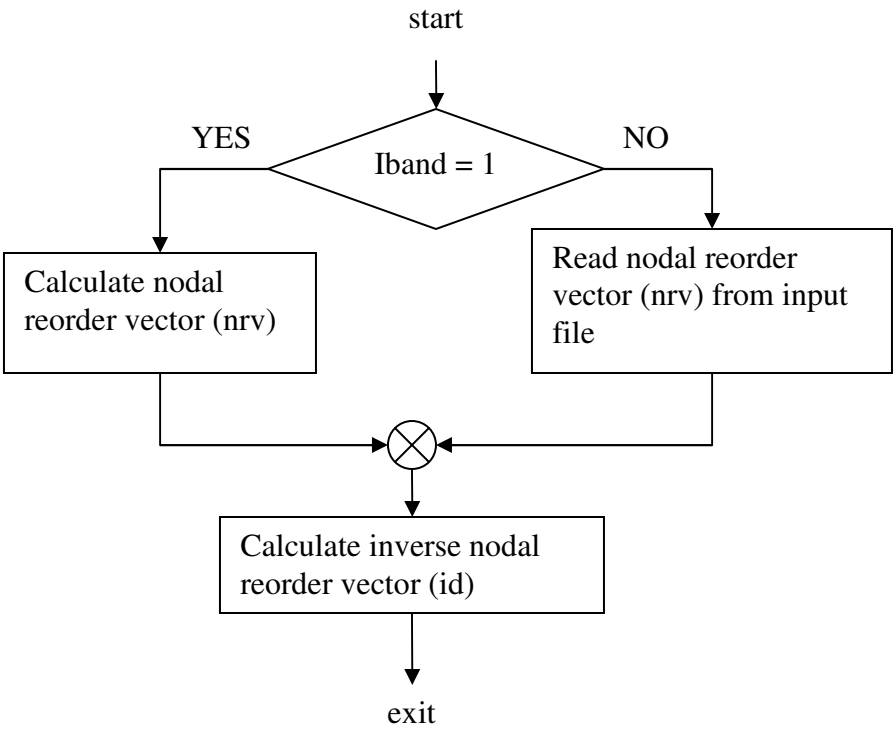


Figure 4-17: Program Flow for Routine BANDW

4.16. Routine RENUM

Routine RENUM is used to renumber input data (node, element and boundary conditions data) from original to reorder numbering system.

List of arguments:

x – *(input/output vector)* Node coordinates

km – *(input/output vector)* Element information

ndtbc – *(input/vector vector)* Nodes with temperature boundary conditions

ndfbc – *(input/vector vector)* Nodes with flux boundary conditions

ndcbc – *(input/vector vector)* Nodes with convection boundary conditions

ndrad – *(input/vector vector)* Nodes with black body radiation boundary conditions

ndrbc – *(input/vector vector)* Nodes with enclosure radiation boundary conditions

id – *(input vector)* Inverse nodal reorder vector

theta – *(input/vector vector)* Node temperatures

dum2d – *()* Reference vector

numnp – *(input value)* Number of node points

numelt – *(input value)* Number of elements

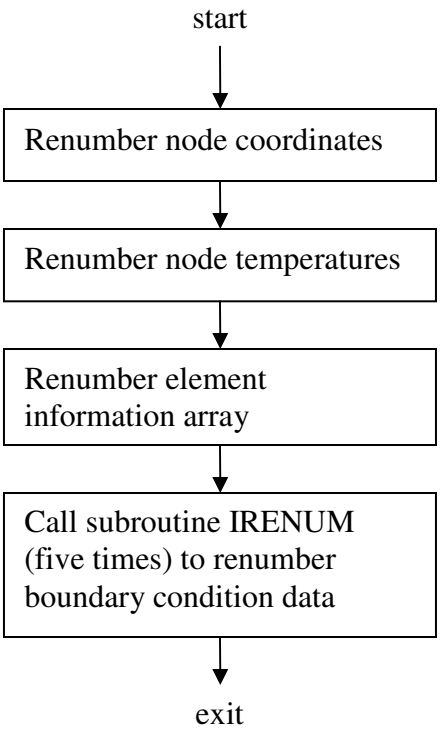


Figure 4-18: Program Flow for Routine RENUM

4.17. Routine IRENUM

Routine IRENUM is used to renumber boundary conditions data from original to reorder (bandwidth) numbering system.

4.18. Routine CALCEFFK1

Routine CALCEFFK1 is used to renumber data used for frame cavity calculations from reorder (bandwidth) to original numbering system, to call routine *CALCEFFK* which calculates frame cavity properties (calculations are performed in original numbering system) and after that to renumber data from original to reorder numbering system.

List of arguments:

x – *(input vector)* Node coordinates

km – *(input vector)* Element information

frcav - *(input vector)* Flag which is used to show if element side belong to edge of frame cavity, and if side belong to edge than to show which side of equivalent rectangularized frame cavity this side belong. This array contain 4 flags for each element (first for side between nodes *km1* and *km2*, second for side between node *km2* and *km3*, third side between *km3* and *km4*, fourth side between *km4* and *km1*). Depending of value side is transformed to one of sides of rectangularized frame cavity:

eq.1: transform to left side

eq.2: transform to down side

eq.3: transform to right side

eq.4:transform to upper side

sideemis - *(input vector)* Show element side emissivity if side belong to edge of frame cavity, and equal with zero if not.

cond1 - *(input/output vector)* used to store results – effective conductivitues of frame cavities.

theta - *(input vector)* node temperatures.

id – *(input vector)* inverse nodal reorder vector – transformation from original to reorder numbering system.

nrv – *(input vector)* nodal reorder vector – transformation from reorder to original numbering system.

idir - *(input/output vector)* stored initial gravity heat flow direction and after calculations, stored calculated heat flow direction (it depends of frame cavity type).

iscr - *(input/output vector)* stored initial screen heat flow direction and after calculations, stored calculated heat flow direction (it depends of frame cavity type).

tc - *(input/output vector)* initial temperatures of rectangularized frame cavities and after calculations, stored calculated temperatures (it depends of frame cavity type).

ec - *(input vector)* emissivity of element side. Equal with zero if side not belong to edge.

pressure - *(input vector)* gas (or gas mixture) pressure in frame cavity.

nmix - *(input vector)* number of gases in gas mixture.

iprop - *(input vector)* Indicate which gasses are implement in gas mixtures. Built in values:

eq.1: Air

eq.2: Argon

eq.3: Krypton

eq.4: Xenon

frct - *(input vector)* Fraction part of gasses in gas mixture

cavmod - *(input vector)* Cavity model:

eq.0: NFRC97

eq.1: CENISO

eq.2: CENISO VENTILATED

eq.3: USER DIMENSION

eq.4: ISO15099

eq.5: ISO15099 VENTILATED

cheight - **(input vector) *jambheight of frame cavity (see***

Figure 3-16)

radiationflag - *(input vector)* flag for radiation calculations

eq.0: omit radiation

eq.1: include radiation

MaxXDimension - *(input vector)* Equivalent “x” dimension of rectangularized cavity

MaxYDimension - *(input vector)* Equivalent “y” dimension of rectangularized cavity

CavArea - *(input vector)* frame cavity area

t1old, t2old - *(input/output vector)* used to store side (of rectangularized frame cavity) temperatures from previous iteration.

Innum - *(input vector)* frame cavity ID

Nusselt - *(output vector)* Calculated Nusselt number

CavKeff - *(output vector)* Calculated Effective conductivity

changehf - *(input/output vector)* detect change of heat flow direction

eq.0: heat flow direction in current iteration is same as in previous

eq.1: heat flow direction in current iteration is not same as in previous

oscillate - *(input/output vector)* sign that heat flow direction on frame cavity side oscillate (change in every iteration). This flag is used to stop oscillations and cause that solution converge.

iconv - *(input value)* temperature scale:

eq.1: Celsius

eq.2: Fahrenheit

eq.-1: Kelvin

eq.-2: Rankin

numnp - *(input value)* number of node points

numel - *(input value)* number of elements

iband - *(input value)* is bandwidth minimization performed

eq.0: no bandwidth minimization

eq.1: bandwidth minimization is performed

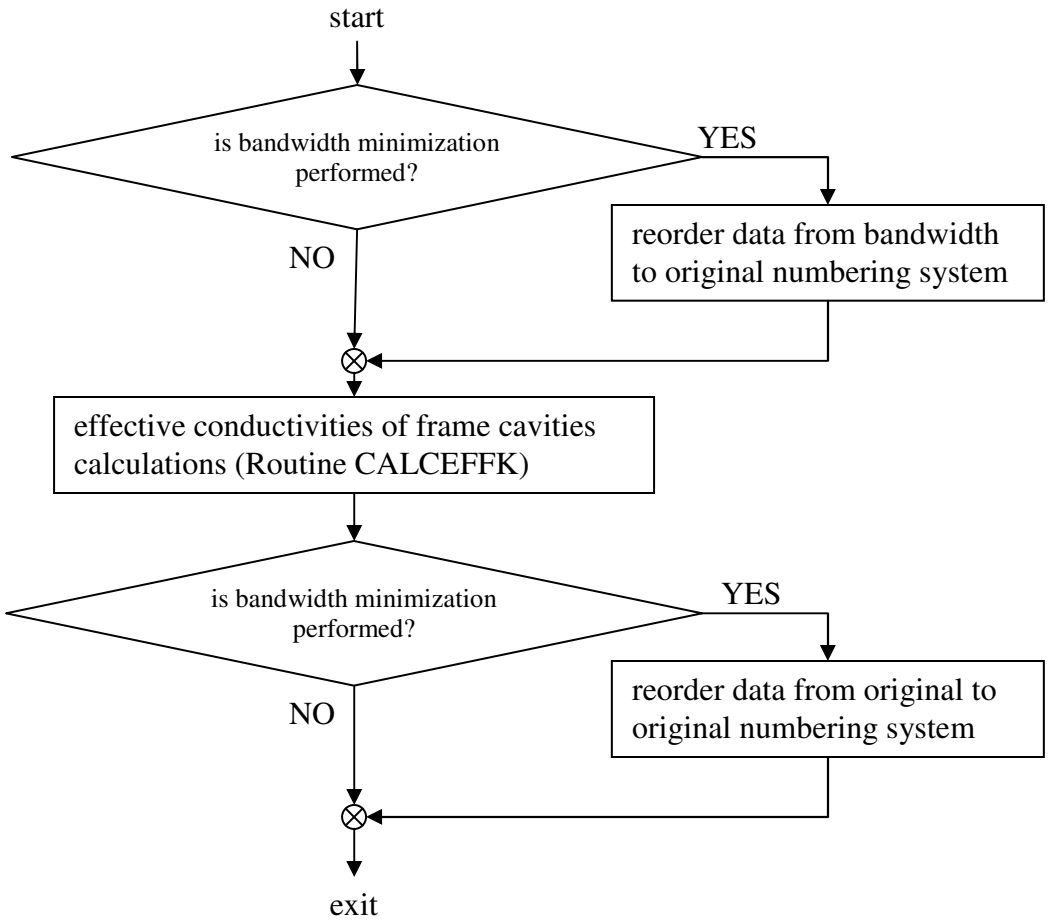


Figure 4-19: Program Flow for Routine CALCEFFK1

4.19. Routine CALCEFFK

Routine CALCEFFK perform calculations described in ISO15099 standard [1].

List of arguments:

see Routine CALCEFFK1

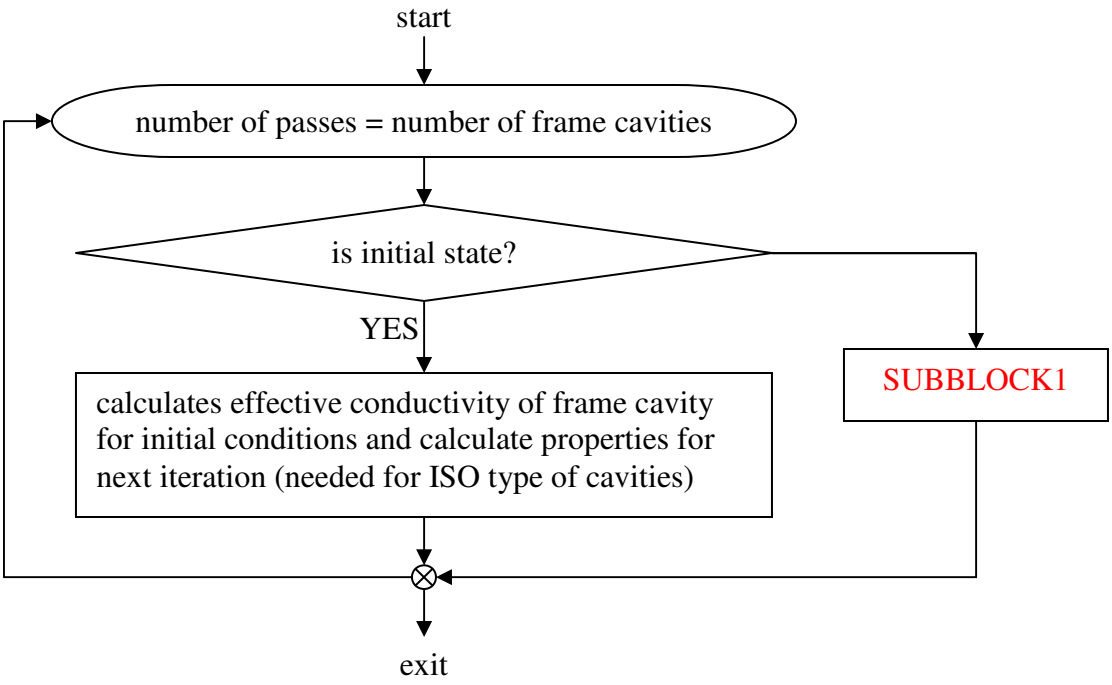


Figure 4-20: Program Flow for Routine CALCEFFK

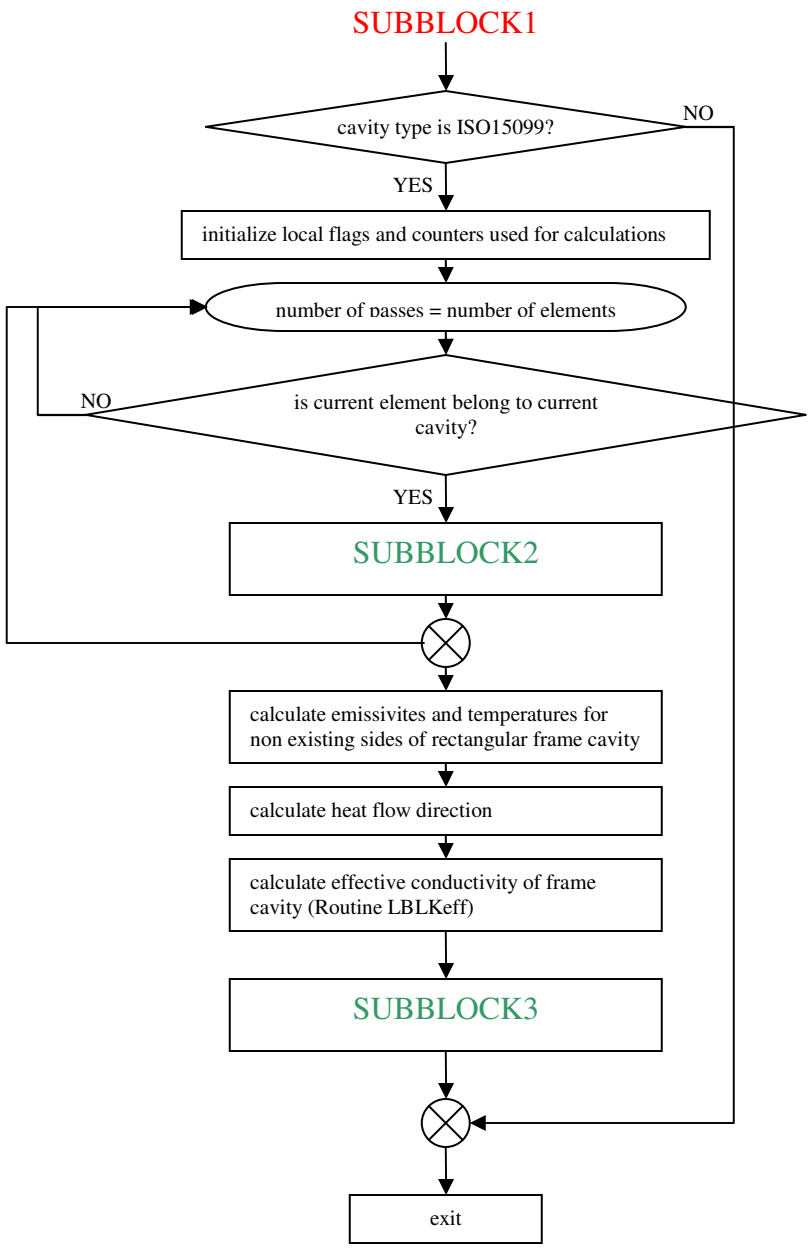


Figure 4-21: SUBBLOCK1

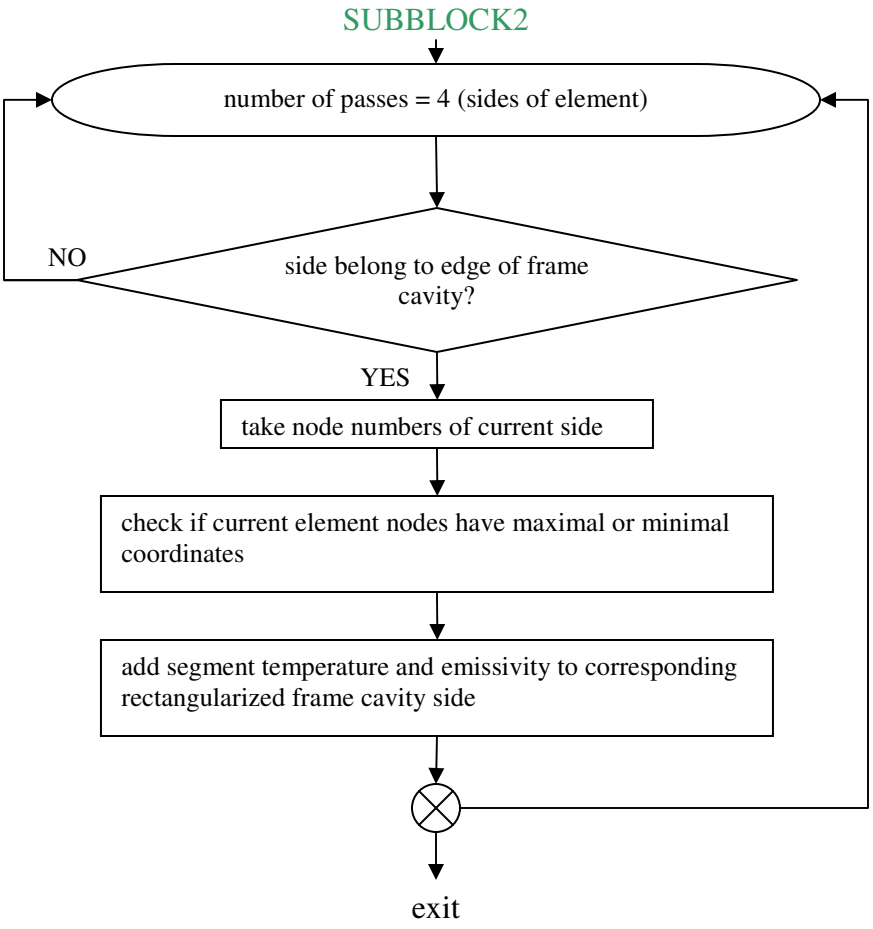


Figure 4-22: SUBBLOCK2

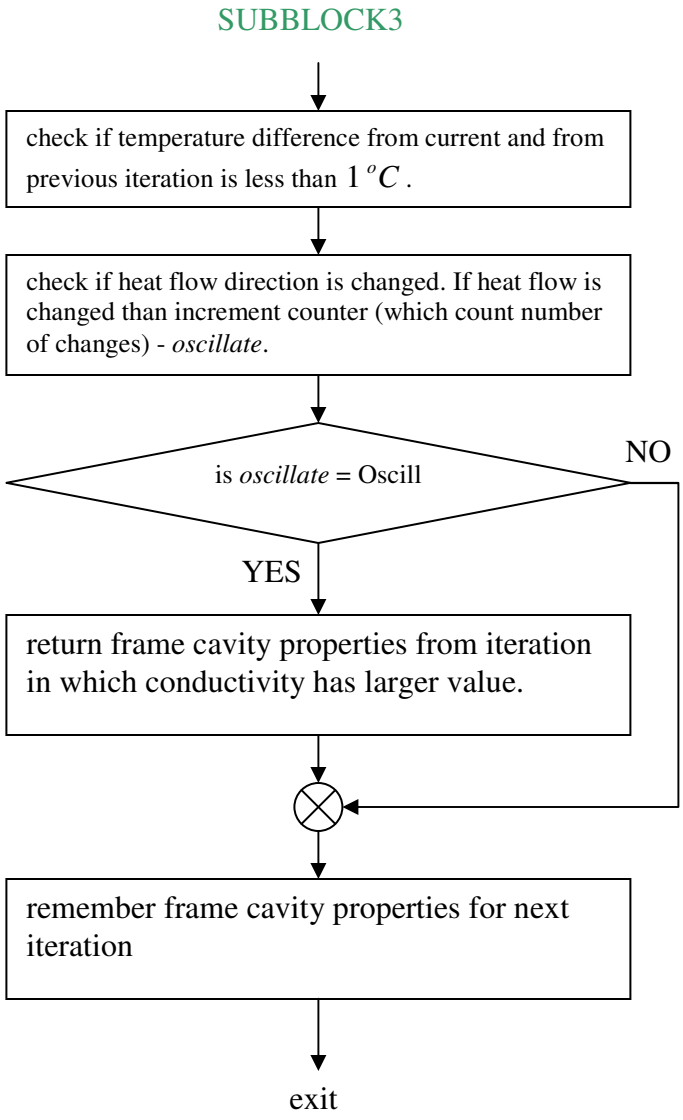


Figure 4-23: SUBBLOCK3

5. Description of Viewer Subroutines

This chapter describes routines which are used by viewer.

5.1. Routine SEE

Routine SEE is used to calculate self shadowing between two surfaces using equations and algorithm described in chapter 3.4.2.

List of arguments:

x – *(input vector)* x-coordinates of nodes

y - *(input vector)* y-coordinates of nodes

z - *(input vector)* z-coordinates of nodes (not used in this version)

xn – *(input vector)* x-coordinates of surface normal

yn - *(input vector)* y-coordinates of surface normal

zn - *(input vector)* z-coordinates of surface normal (not used in this version)

km - *(input vector)* node numbers of which are segment consist

iseg - *(input value)* number of iseg-th segment

jseg - *(input value)* number of jseg-th segment

ndim - *(input value)* type of problem (2 = two dimensional)

iedge - *(output value)* number of same nodes of i-th and j-th surfaces

isee - *(output value)* flag which shows self shadowing type

eq.-1: partial shadowing

eq.0: total shadowing

eq.1: no shadowing

ibug - *(output value)* debug information

eq.0: no debug information

eq.1: minimal debug information

eq.2: maximum debug information

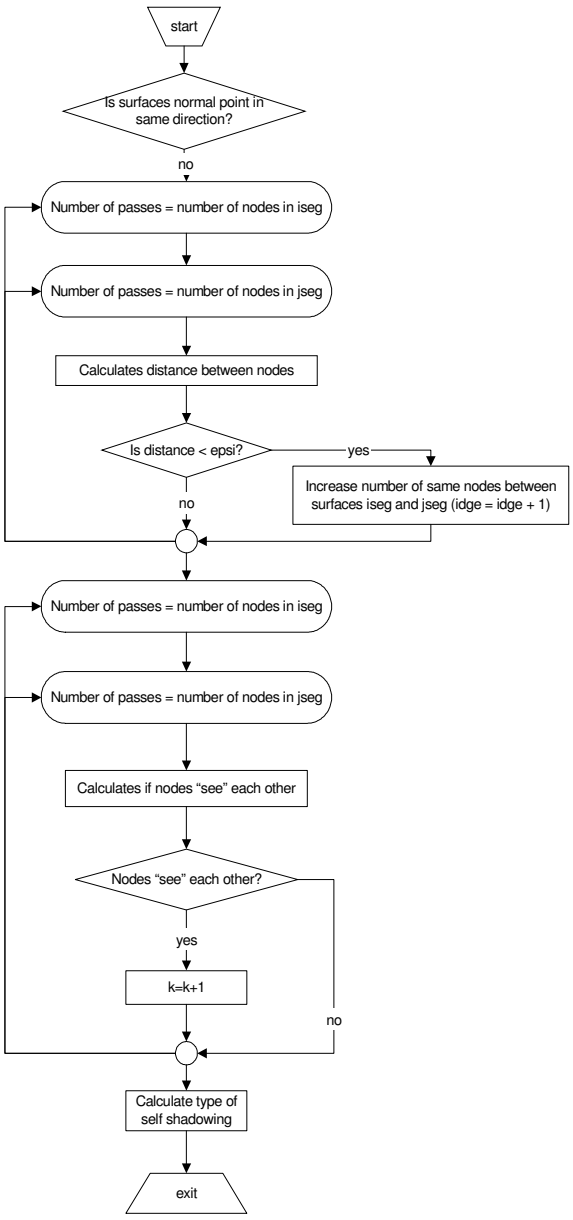


Figure 5-1: Program Flow of Routine See

5.2. Routine INTSEC2

Routine INTSEC2 is used to determine if two lines has intersection point. Algorithm used in this routine is described in chapter 3.4.3.

List of arguments:

x_1 – *(input value)* x-coordinate of first node in line1

y_1 – *(input value)* y-coordinate of first node in line1

x_2 – *(input value)* x-coordinate of second node in line1

y_2 – *(input value)* x-coordinate of second node in line1

x_3 – *(input value)* x-coordinate of first node in line2

y_3 – *(input value)* y-coordinate of first node in line2

x_4 – *(input value)* x-coordinate of second node in line2

y_4 – *(input value)* x-coordinate of second node in line2

int - *(output value)* show if intersection exist

eq.0: no intersection

eq.1: intersection exist

5.3. Routine GRL2D

Routine GRL2D is used to determine number and coordinates of grid cells that line pass through (see Figure 3-35 and Figure 3-36).

List of arguments:

x_1 – *(input value)* x-coordinate of first node

y_1 – *(input value)* y-coordinate of first node

x_2 – *(input value)* x-coordinate of second node

y_2 – *(input value)* x-coordinate of second node

xgrid - *(input vector)* x-coordinates of grid net

ygrid - *(input vector)* y-coordinates of grid net

xp - *(output vector)* x-coordinates of intersection points between grid net and line

yp - *(output vector)* y-coordinates of intersection points between grid net and line

nwk1 - *(output vector)* cells coordinates that line pass through: nwk1(1,i) contain cell coordinate of grid net and dimension of nwk1 is (1,numcell) where numcell is number of cells that line pass through.

nwk2 - *(output vector)* cells coordinates that line pass through (in different form than in nwk1): nwk2(1,j) = 1 if line pass through cell with coordinate j and nwk2(1,j) if not. Dimension of nwk2 is (1,NumberOfCellsInGrid).

nxg - *(input value)* number of grid cells in x direction

nyg - *(input value)* number of grid cells in y direction

numcell - *(output value)* number of cells that line pass through

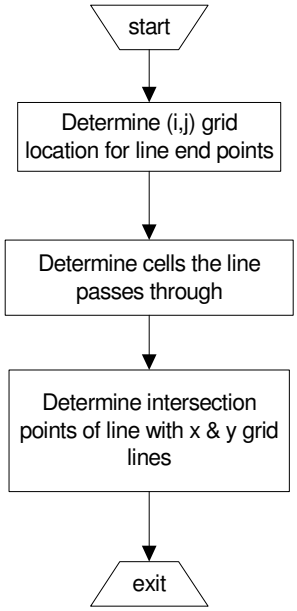


Figure 5-2: Program Flow for Routine Grl2d

5.4. Routine GRID

Routine GRID is used to determine through which grid cells blocking surfaces pass. This is used for "Grid" Algorithm which is described in chapter 3.3.

List of arguments:

x – *(input vector)* x-coordinates of nodes

y - *(input vector)* y-coordinates of nodes

z - *(input vector)* z-coordinates of nodes (not used in this version)

km - *(input vector)* blocking surfaces information array

kbkl - *(input vector)* blocking surfaces number

xgrid – *(input vector)* x-coordinates of grid net

ygrid - *(input vector)* y-coordinates of grid net

zgrid - *(input vector)* z-coordinates of grid net (not used in this version)

xp – *(input vector)* x-coordinates of intersection points of ray between surfaces with x, y and z grid planes

yp - *(input vector)* y-coordinates of intersection points of ray between surfaces with x, y and z grid planes

zp - *(input vector)* z-coordinates of intersection points of ray between surfaces with x, y and z grid planes (not used in this version)

nwk1 - *(temporary used vector)* cells coordinate that blocking surface pass through

nwk2 - *(temporary used vector)* cells coordinate that blocking surface pass through

ngr1 - *(output vector)* staring reading positions form array ngr3

ngr2 - *(output vector)* number of blocking surfaces (for each grid cells)

ngr3 - *(output vector)* blocking surfaces numbers (for each grid cells)

igrid - *(temporary used matrix)* blocking surfaces number in grid cells

ndim - *(input value)* type of problem (2 = two dimensional)

nxg - *(input value)* number of grid cells in x direction

nyg - *(input value)* number of grid cells in y direction

nzg - *(input value)* number of grid cells in z direction (not used in this version)

nxyz -

numnp - *(input value)* number of node points

numel - (input value)

nl -

nblk - *(input value)* number of blocking surfaces

ibug - *(input value)* debug information flag

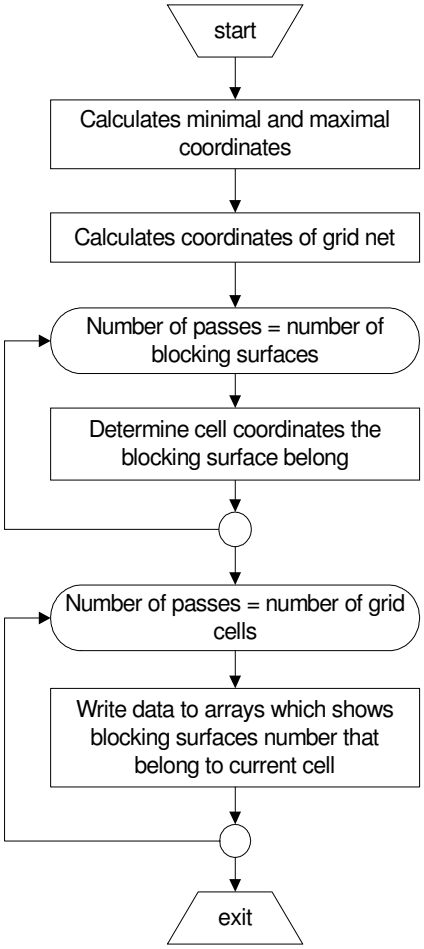


Figure 5-3: Program Flow for Routine Grid

5.5. Routine OBSTR

Routine OBSTR is used to determine if ray between two surfaces is blocked by third (blocking) surface.

List of arguments:

xl1 – *(input value)* x-coordinate of beginning of ray

yl1 – *(input value)* y-coordinate of beginning of ray

zl1 – *(input value)* y-coordinate of beginning of ray

xl2 – *(input value)* x-coordinate of end of ray

yl2 – *(input value)* y-coordinate of end of ray

zl2 – *(input value)* y-coordinate of end of ray

x – *(input vector)* x-coordinates of nodes

y - *(input vector)* y-coordinates of nodes

z - *(input vector)* z-coordinates of nodes (not used in this version)

xn – *(input vector)* x-coordinates of surface normal

yn - *(input vector)* y-coordinates of surface normal

zn - *(input vector)* z-coordinates of surface normal (not used in this version)

km - *(input vector)* blocking surfaces information array

nwk1 - *(temporary used vector)* cells coordinate that blocking surface pass through

ngr1 - *(input vector)* staring reading positions form array ngr3

ngr2 - *(input vector)* number of blocking surfaces (for each grid cells)

ngr3 - *(input vector)* blocking surfaces numbers (for each grid cells)

iseg - *(input value)* number of i-th segment

jseg - *(input value)* number of j-th segment

numcel - *(input value)* number of grid cells

ndim - *(input value)* type of problem (2 = two dimensional)

int - *(output value)* show if intersection exist

eq.0: no intersection

eq.1: intersection exist

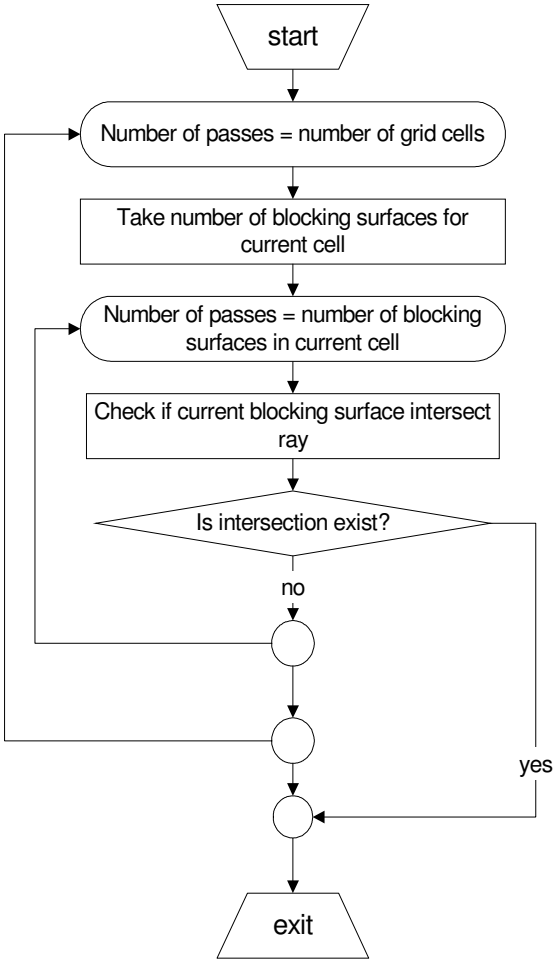


Figure 5-4: Program Flow for Routine Obstr

5.6. Routine VIEW2D

Routine VIEW2D is used to calculate view factors by Eq. 1.3-49 for 2D planar geometry. If third surface shadowing exists, surfaces are divided into the subsurfaces which are used for partial view factor calculation (Eq. 1.3-50)

List of arguments:

x – *(input vector)* x-coordinates of nodes

y - *(input vector)* y-coordinates of nodes

z - *(input vector)* z-coordinates of nodes (not used in this version)

xn – *(input vector)* x-coordinates of surface normal

yn - *(input vector)* y-coordinates of surface normal

zn - *(input vector)* z-coordinates of surface normal (not used in this version)

xc – *(input vector)* x-coordinates of surfaces center

yc - *(input vector)* y-coordinates of surfaces center

zc - *(input vector)* z-coordinates of surfaces center (not used in this version)

km - *(input vector)* node numbers of which are segment consist

area - *(input vector)* segment length

frow - ?

xgrid – *(input vector)* x-coordinates of grid net

ygrid - *(input vector)* y-coordinates of grid net

zgrid - *(input vector)* z-coordinates of grid net (not used in this version)

xp – *(input vector)* x-coordinates of intersection points of ray between surfaces with x, y and z grid planes

yp - *(input vector)* y-coordinates of intersection points of ray between surfaces with x, y and z grid planes

zp - *(input vector)* z-coordinates of intersection points of ray between surfaces with x, y and z grid planes (not used in this version)

nwk1 - *(input vector)* cells coordinate that line pass through

nwk2 - *(input vector)* cells coordinate that line pass through

ngr1 - *(input vector)* staring reading positions form array ngr3

ngr2 - *(input vector)* number of blocking surfaces (for each grid cells)

ngr3 - *(input vector)* blocking surfaces numbers (for each grid cells)

ibug - *(input value)* debug information flag

nrcond - *(input vector)* show to which enclosure segment belongs

eq.n0: segment belong to n-th enclosure and it is part of conduction element (type of enclosure is manual)

eq.n1: segment belong to n-th enclosure and it is not part of conduction element (type of enclosure is manual)

eq.n2: segment belong to n-th enclosure and it is part of conduction element (type of enclosure is automatic)

eq.n3: segment belong to n-th enclosure and it is not part of conduction element (type of enclosure is automatic)

nrdim - *(input value)* type of problem (2 = two dimensional)

f - *(output vector)* view factors

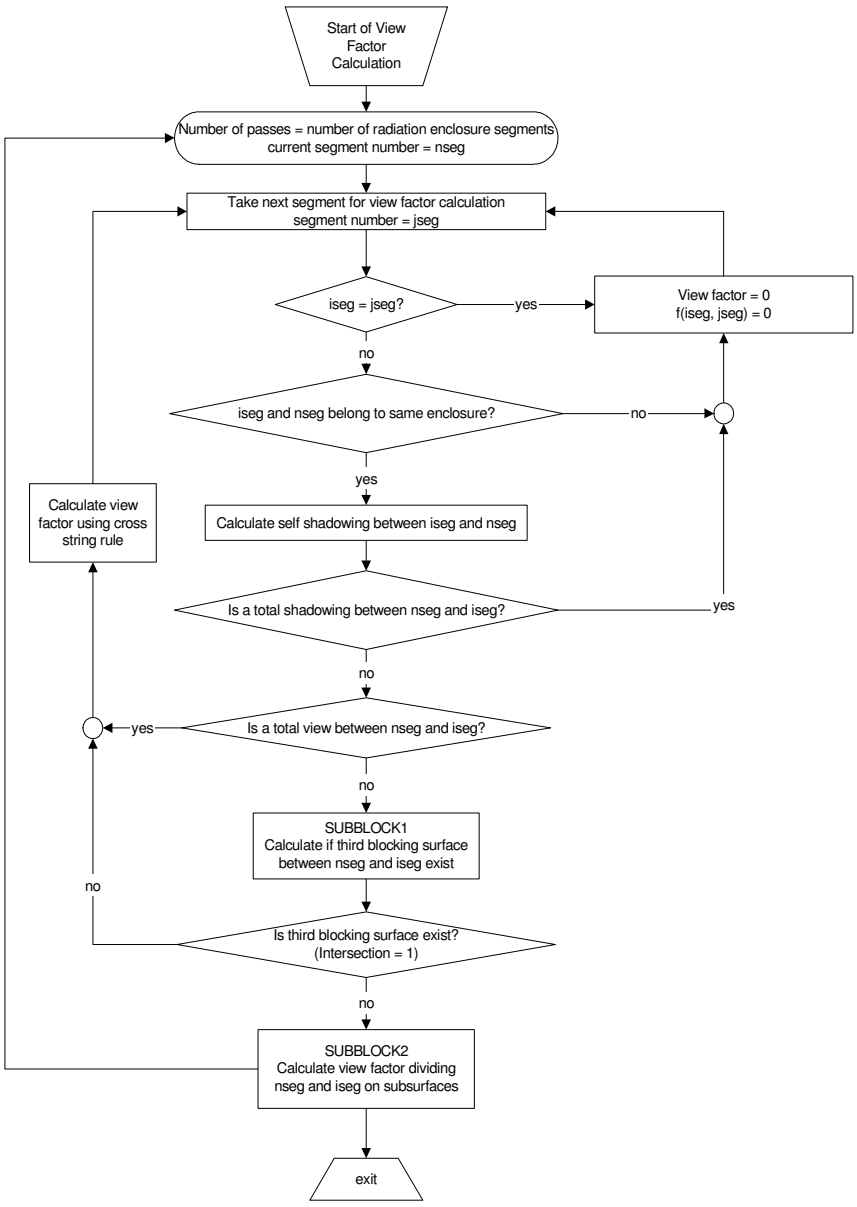


Figure 5-5: Program Flow for Routine View2d

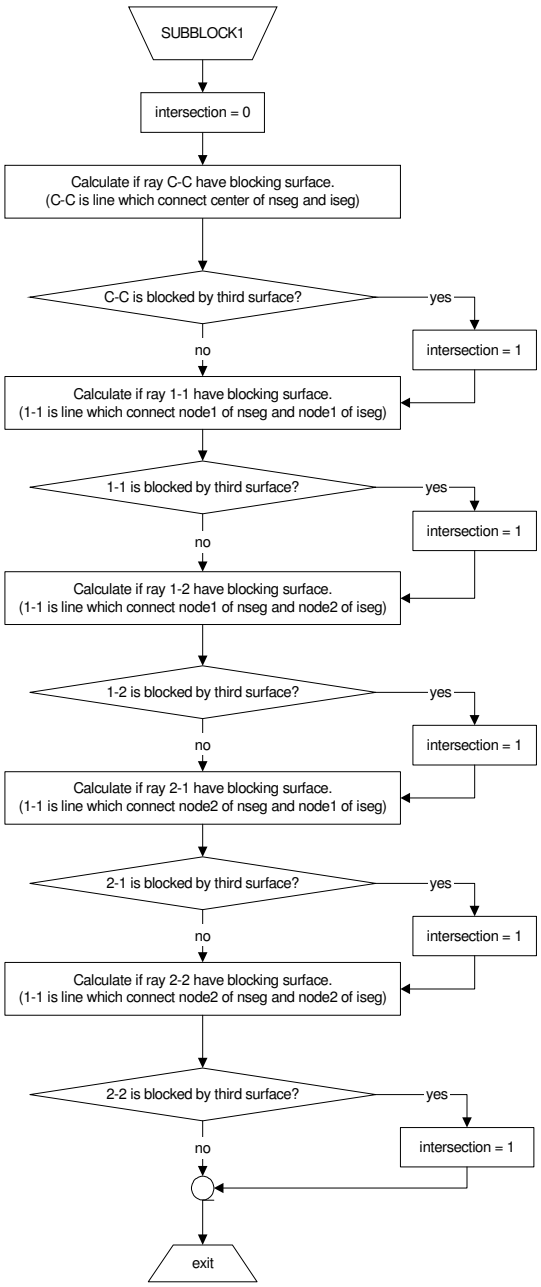


Figure 5-6: Subblock1

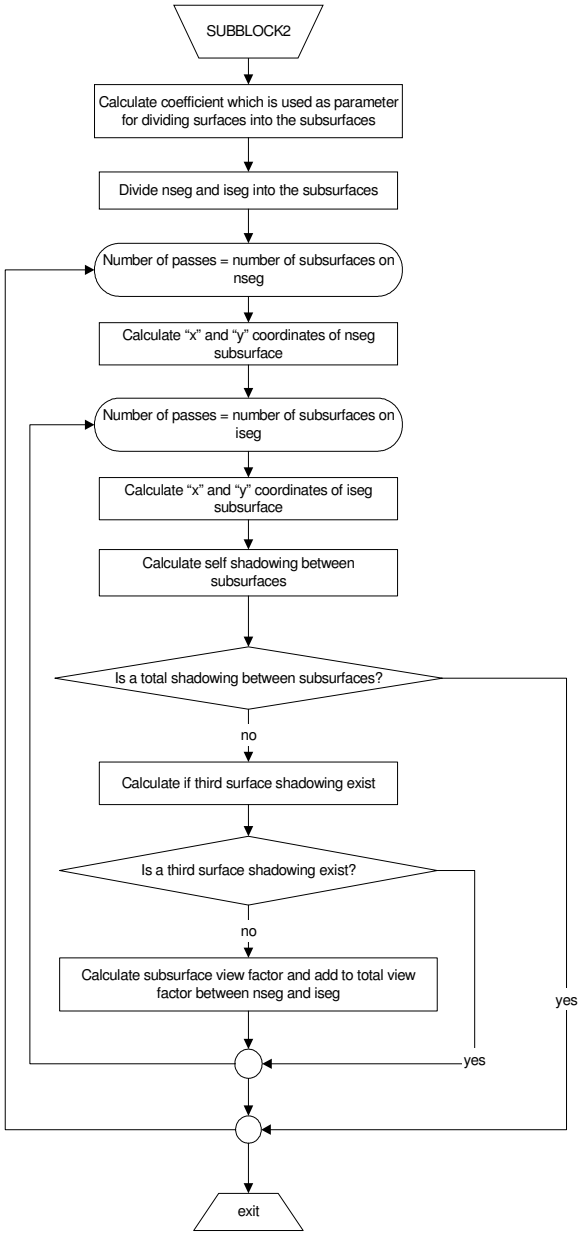


Figure 5-7: Subblock2

5.7. Routine GEOMVW

Routine GEOMVW is used to calculate surfaces normal and center.

List of arguments:

x – *(input vector)* x-coordinates of nodes

y - *(input vector)* y-coordinates of nodes

z - *(input vector)* z-coordinates of nodes (not used in this version)

xn – *(output vector)* x-coordinates of surface normal

yn - *(output vector)* y-coordinates of surface normal

zn - *(output vector)* z-coordinates of surface normal (not used in this version)

xc – *(output vector)* x-coordinates of surfaces center

yc - *(output vector)* y-coordinates of surfaces center

zc - *(output vector)* z-coordinates of surfaces center (not used in this version)

km - *(input vector)* node numbers of which are segment consist

area - *(output vector)* segment length

numel - *(input value)* number of radiation enclosure segments

ndim - *(input value)* dimension of problem (in this version is =2 or 2D planar)

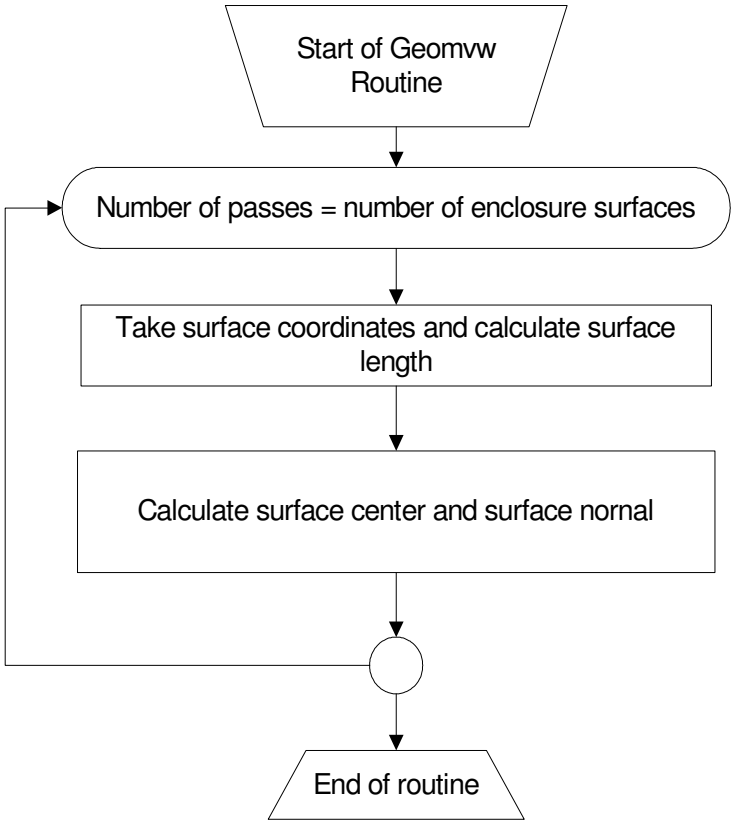


Figure 5-8: Program Flow for Routine Geomvw

6. Examples

6.1. Bandwidth Minimization

Aim of bandwidth minimization is explained in section 3.1.

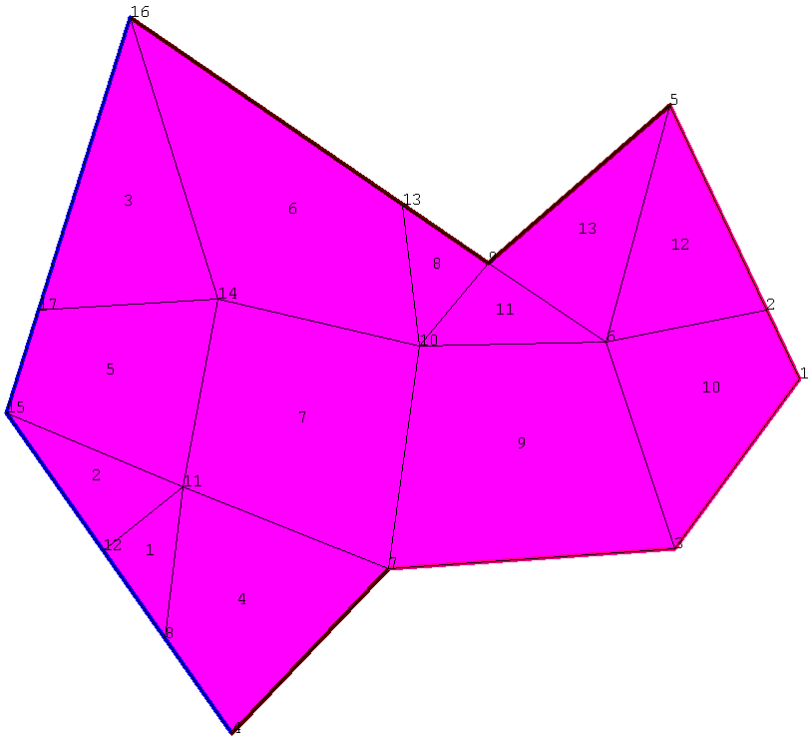


Figure 6-1: Bandwidth Minimization Example in Original Numbering System (before minimization)

In Table 6.1-1 are shown results obtained for this example. Vector ***nrv*** is used for BWM^{-1} transformation and vector ***id*** is used for BWM transformation. For example, node numbered with 3 in original numbering system becomes 4 in reorder numbering system (Figure 6-2).

Table 6.1-1: Nodal Reorder and Inverse Nodal Reorder Vectors

node number in original numbering system	nrv(1,i)	id(1,i)-node number in reorder numbering system
1	1	1
2	2	2
3	5	4
4	3	13
5	6	3
6	9	5
7	13	9
8	16	14
9	7	6
10	10	10
11	14	15
12	17	17
13	4	7
14	8	11
15	11	16
16	15	8
17	12	12

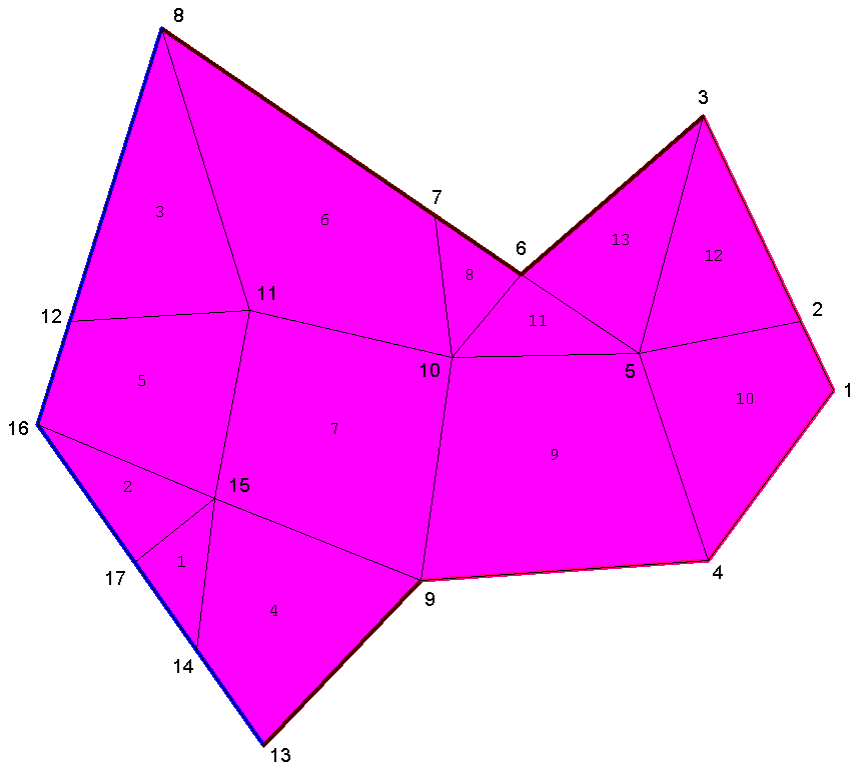


Figure 6-2: Node Numbers in Reordering System

6.2. Assembling to Global Matrix Equation

Example which is used for assembling is shown in Figure 6-3:

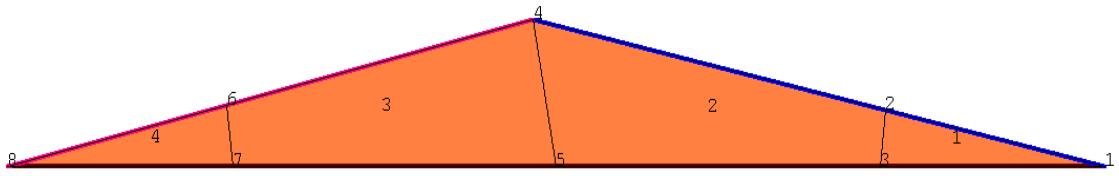


Figure 6-3: Example

Elements with local nodes numbering are shown in Figure 6-4, Figure 6-6, Figure 6-7 and Figure 6-8.

For first element, node coordinates (signed in local domain) are:

$$\begin{aligned} x_1 &= 0.02712\,m, \quad y_1 = -0.00515\,m \\ x_2 &= 0.0459\,m, \quad y_2 = -0.00515\,m \\ x_3 &= 0.02754\,m, \quad y_3 = -0.001194\,m \end{aligned} \qquad \textit{Eq. 6.2-1}$$

and material conductivity (all elements) $cond = 0.2233\frac{W}{mK}$. Shape functions for triangular element is calculated by Eq. 2.7-3 and for first element coordinates coefficients (Eq. 2.7-4) are equal:

$$\alpha_1 = 8.7E-05, \quad \alpha_2 = -0.00011, \quad \alpha_3 = 9.67E-05,$$
$$\beta_1 = -0.00396, \quad \beta_2 = 0.003956, \quad \beta_3 = 0, \qquad \textit{Eq. 6.2-2}$$
$$\gamma_1 = -0.01836, \quad \gamma_2 = -0.00042, \quad \gamma_3 = 0.01878$$

To obtain conductivity matrix, first derivative of element shape functions are equal:

$$\frac{\partial \psi_i^e}{\partial x} = \frac{1}{2A_e} * \beta_i^e$$
$$\textit{Eq. 6.2-3}$$
$$\frac{\partial \psi_i^e}{\partial y} = \frac{1}{2A_e} * \gamma_i^e$$

when replacing in Eq. 2.9-7 following equation is obtained:

$$K_{ij}^e = \frac{k}{4A_e^2} \int_{\Omega_m} (\beta_i\beta_j + \gamma_i\gamma_j) dxdy = \frac{k}{4A_e^2} (\beta_i\beta_j + \gamma_i\gamma_j) \int_{\Omega_m} dxdy =$$
$$\textit{Eq. 6.2-4}$$
$$= \frac{k}{4A_e^2} (\beta_i\beta_j + \gamma_i\gamma_j) * A_e = \frac{k}{4A_e} (\beta_i\beta_j + \gamma_i\gamma_j)$$

and for first element conductivity matrix is:

$$\left\| K_{ij}^I \right\| = \begin{Vmatrix} K_{11}^I & K_{12}^I & K_{13}^I \\ K_{21}^I & K_{22}^I & K_{23}^I \\ K_{31}^I & K_{32}^I & K_{33}^I \end{Vmatrix} = \begin{Vmatrix} 0.5298665 & -0.011925 & -0.517941 \\ -0.011925 & 0.023774 & -0.011848 \\ -0.517941 & -0.011848 & 0.5297897 \end{Vmatrix} \qquad \textit{Eq. 6.2-5}$$

Conduction matrix in Eq. 6.2-5 is obtained using shape element equations described in section 2.7.1.3. Important note is that Conrad uses equations for linear rectangular element to integrate over the triangular element. Local nodes numbering for triangular element in therm is shown in Figure 6-5.

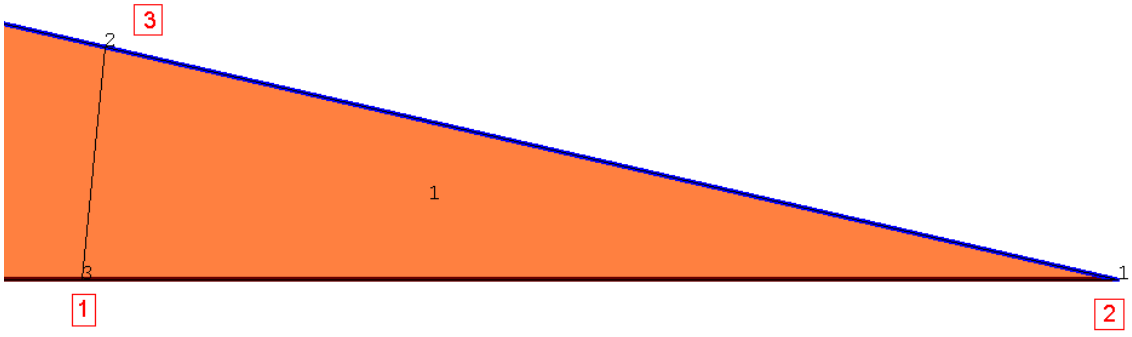


Figure 6-4: First Element in Local Node Numbering

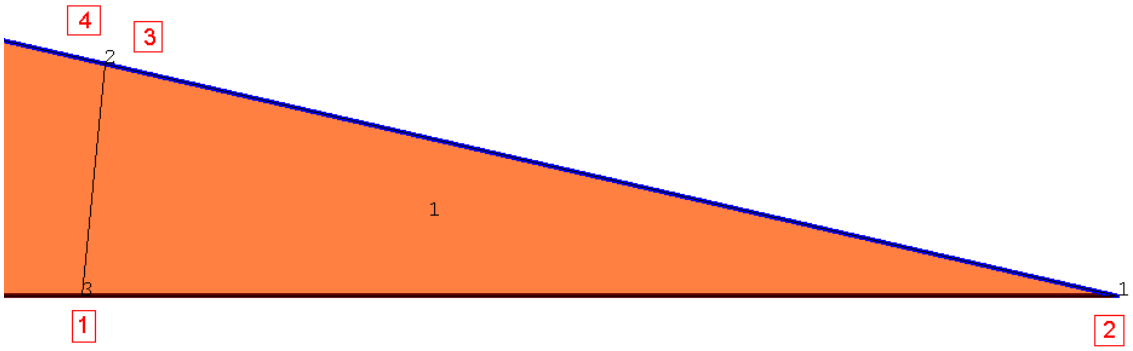


Figure 6-5: Local Node Numbering in Conrad

Conduction matrix obtained in Conrad (using equations for rectangular element) is:

$$\left\| K_{ij}^{I} \right\| = \begin{Vmatrix} 0.52998 & -1.193e-2 & -0.52998 & 1.193e-2 \\ -1.193e-2 & 2.378e-2 & 1.193e-2 & -2.378e-2 \\ -0.52998 & 1.193e-2 & 1.0837 & -0.5657 \\ 1.193e-2 & -2.378e-2 & -0.5657 & 0.5775 \end{Vmatrix} \qquad \textit{Eq. 6.2-6}$$

and this local system of linear equations for triangular element can be presented as:

$$\begin{Vmatrix} 0.52998 & -1.193e-2 & -0.52998 & 1.193e-2 \\ -1.193e-2 & 2.378e-2 & 1.193e-2 & -2.378e-2 \\ -0.52998 & 1.193e-2 & 1.0837 & -0.5657 \\ 1.193e-2 & -2.378e-2 & -0.5657 & 0.5775 \end{Vmatrix} * \begin{Vmatrix} T_1^I \\ T_2^I \\ T_3^I \\ T_4^I \end{Vmatrix} = \begin{Vmatrix} 0 \\ 0 \\ 0 \\ 0 \end{Vmatrix} \qquad \textit{Eq. 6.2-7}$$

because $T_3^I = T_4^I$, Eq. 6.2-7 can be presented as following system:

$$\begin{Vmatrix} 0.52998 & -1.193e-2 & -0.51805 \\ -1.193e-2 & 2.378e-2 & -1.185e-2 \\ -0.51805 & -1.185e-2 & 0.5298 \end{Vmatrix} * \begin{Vmatrix} T_1^I \\ T_2^I \\ T_3^I \end{Vmatrix} = \begin{Vmatrix} 0 \\ 0 \\ 0 \end{Vmatrix} \qquad \textit{Eq. 6.2-8}$$

where conduction matrix is approximately equal with conduction matrix in Eq. 6.2-5. Replacing notation from local to global Eq. 6.2-8 becomes:

$$\begin{Vmatrix} 0.52998 & -1.193e-2 & -0.51805 \\ -1.193e-2 & 2.378e-2 & -1.185e-2 \\ -0.51805 & -1.185e-2 & 0.5298 \end{Vmatrix} * \begin{Vmatrix} T_3 \\ T_1 \\ T_2 \end{Vmatrix} = \begin{Vmatrix} 0 \\ 0 \\ 0 \end{Vmatrix} \qquad \textit{Eq. 6.2-9}$$

which leads that conductivity coefficients are assembled into the global matrix by following:

$$K_{33} = 0.52998,\ K_{31} = -1.193e-2,\ K_{32} = -0.51805,$$
$$K_{13} = -1.193e-2,\ K_{11} = 2.378e-2,\ K_{12} = -1.185e-2, \qquad \textit{Eq. 6.2-10}$$
$$K_{23} = -0.51805,\ K_{21} = -1.185e-2,\ K_{22} = 0.5298$$

To obtain element conductivity matrices is explained in chapter 2.9. For rectangular element shown in Figure 6-6 following results are obtained from equations described in 2.9:

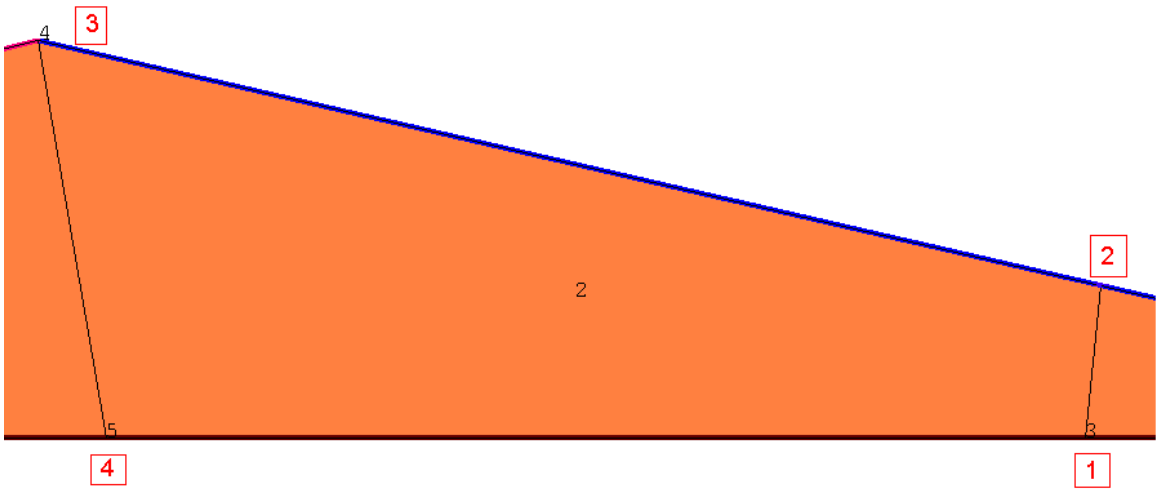


Figure 6-6: Second Element in Local Node Numbering

Node coordinates of element number 2 in local domain notation:

$$x_1 = 0.02712\,m,\quad y_1 = -0.00515\,m,$$
$$x_2 = 0.02754\,m,\quad y_2 = -0.001194\,m, \qquad \textit{Eq. 6.2-11}$$
$$x_3 = -0.0019\,m,\quad y_3 = 0.00515\,m,$$
$$x_4 = 0\,m,\quad y_4 = -0.00515\,m$$

Gauss point integration is calculated in following points:

Table 6.2-1: Gauss Points for Numerical Integration

point	ξ	η
I	0.57735027	0.57735027
II	0.57735027	-0.5773503
III	-0.5773503	0.57735027
IV	-0.5773503	-0.5773503

Inverse Jacobian Matrices in Gauss points are:

$$POINT\ \ I$$
$$J^{-1} = \begin{Vmatrix} 39.6591972 & -71.016513 \\ 229.470604 & -11.174225 \end{Vmatrix} \qquad \textit{Eq. 6.2-12}$$

POINT II

$$J^{-1} = \begin{Vmatrix} 65.4098248 & -69.24406 \\ 378.465351 & -0.9187015 \end{Vmatrix}$$ ***Eq. 6.2-13***

POINT III

$$J^{-1} = \begin{Vmatrix} 10.9226275 & -72.994492 \\ 224.948995 & -11.485454 \end{Vmatrix}$$ ***Eq. 6.2-14***

POINT IV

$$J^{-1} = \begin{Vmatrix} 18.3464512 & -72.4835 \\ 377.840934 & -0.961681 \end{Vmatrix}$$ ***Eq. 6.2-15***

Shape function derivatives in local domain are:

Table 6.2-2: Function Derivatives in Local Domain (for element numbered with 2)

	point I		point II		point III		point IV
	-0.1056624		-0.3943376		-0.1056624		-0.3943376
$\frac{\partial \hat{\psi}_i^e}{\partial \xi}$	0.10566243	$\frac{\partial \hat{\psi}_i^e}{\partial \xi}$	0.39433757	$\frac{\partial \hat{\psi}_i^e}{\partial \xi}$	0.10566243	$\frac{\partial \hat{\psi}_i^e}{\partial \xi}$	0.39433757
	0.39433757		0.10566243		0.39433757		0.10566243
	-0.3943376		-0.1056624		-0.3943376		-0.1056624
	-0.1056624		-0.1056624		-0.3943376		-0.3943376
$\frac{\partial \hat{\psi}_i^e}{\partial \eta}$	-0.3943376	$\frac{\partial \hat{\psi}_i^e}{\partial \eta}$	-0.3943376	$\frac{\partial \hat{\psi}_i^e}{\partial \eta}$	-0.1056624	$\frac{\partial \hat{\psi}_i^e}{\partial \eta}$	-0.1056624
	0.39433757		0.39433757		0.10566243		0.10566243
	0.10566243		0.10566243		0.39433757		0.39433757

Table 6.2-3: Function Derivatives in Global Domain (for element numbered with 2)

	point I		point II		point III		point IV
	3.31329026		-18.477055		27.6303592		21.3482721
$\frac{\partial \psi_i^e}{\partial x}$	32.1949662	$\frac{\partial \psi_i^e}{\partial x}$	53.0990854	$\frac{\partial \psi_i^e}{\partial x}$	8.86688704	$\frac{\partial \psi_i^e}{\partial x}$	14.8934779
	-12.365368		-20.394173		-3.4055733		-5.7202523
	-23.142889		-14.227857		-33.091673		-30.521498
	-23.065726		-149.14603		-19.239512		-148.61765
$\frac{\partial \psi_i^e}{\partial y}$	28.6528388	$\frac{\partial \psi_i^e}{\partial y}$	149.605384	$\frac{\partial \psi_i^e}{\partial y}$	24.982239	$\frac{\partial \psi_i^e}{\partial y}$	149.098488
	86.0824631		39.6272912		87.4922585		39.8219787
	-91.669575		-40.086642		-93.234985		-40.302819

Conduction matrices at gauss points are:

$$\left\| K_{ij}^{first} \right\| = \begin{Vmatrix} 0.007649 & -0.00781 & -0.02854 & 0.028703 \\ -0.00781 & 0.026164 & 0.029135 & -0.04749 \\ -0.02854 & 0.029135 & 0.106531 & -0.10712 \\ 0.028703 & -0.04749 & -0.10712 & 0.12591 \end{Vmatrix} \qquad \textit{Eq. 6.2-16}$$

$$\left\| K_{ij}^{second} \right\| = \begin{Vmatrix} 0.192892 & -0.19894 & -0.04726 & 0.053306 \\ -0.19894 & 0.215228 & 0.041383 & -0.05767 \\ -0.04726 & 0.041383 & 0.016963 & -0.01109 \\ 0.053306 & -0.05767 & -0.01109 & 0.015453 \end{Vmatrix} \qquad \textit{Eq. 6.2-17}$$

$$\left\| K_{ij}^{third} \right\| = \begin{Vmatrix} 0.015535 & -0.00323 & -0.02436 & 0.012052 \\ -0.00323 & 0.00963 & 0.02954 & -0.03594 \\ -0.02436 & 0.02954 & 0.105061 & -0.11024 \\ 0.012052 & -0.03594 & -0.11024 & 0.134132 \end{Vmatrix} \qquad \textit{Eq. 6.2-18}$$

$$\left\| K_{ij}^{fourth} \right\| = \begin{Vmatrix} 0.183921 & -0.17819 & -0.04928 & 0.043552 \\ -0.17819 & 0.18318 & 0.047746 & -0.05274 \\ -0.04928 & 0.047746 & 0.013205 & -0.01167 \\ 0.043552 & -0.05274 & -0.01167 & 0.020853 \end{Vmatrix} \qquad \textit{Eq. 6.2-19}$$

and finally

$$\left\| K_{ij} \right\| = \left\| K_{ij}^{first} \right\| + \left\| K_{ij}^{second} \right\| + \left\| K_{ij}^{third} \right\| + \left\| K_{ij}^{fourth} \right\| \qquad \textit{Eq. 6.2-20}$$

after substituting

$$\left\| K_{ij} \right\| = \begin{Vmatrix} 0.399997 & -0.38817 & -0.14944 & 0.137613 \\ -0.38817 & 0.434203 & 0.147803 & -0.19384 \\ -0.14944 & 0.147803 & 0.24176 & -0.24012 \\ 0.137613 & -0.19384 & -0.24012 & 0.296347 \end{Vmatrix} \qquad \textit{Eq. 6.2-21}$$

Therefore, local system of linear equation is:

$$\begin{Vmatrix} 0.399997 & -0.38817 & -0.14944 & 0.137613 \\ -0.38817 & 0.434203 & 0.147803 & -0.19384 \\ -0.14944 & 0.147803 & 0.24176 & -0.24012 \\ 0.137613 & -0.19384 & -0.24012 & 0.296347 \end{Vmatrix} * \begin{Vmatrix} T_1^{II} \\ T_2^{II} \\ T_3^{II} \\ T_4^{II} \end{Vmatrix} = \begin{Vmatrix} 0 \\ 0 \\ 0 \\ 0 \end{Vmatrix} \qquad \textit{Eq. 6.2-22}$$

which is in global notation equal:

$$\begin{Vmatrix} 0.399997 & -0.38817 & -0.14944 & 0.137613 \\ -0.38817 & 0.434203 & 0.147803 & -0.19384 \\ -0.14944 & 0.147803 & 0.24176 & -0.24012 \\ 0.137613 & -0.19384 & -0.24012 & 0.296347 \end{Vmatrix} * \begin{Vmatrix} T_3 \\ T_2 \\ T_4 \\ T_5 \end{Vmatrix} = \begin{Vmatrix} 0 \\ 0 \\ 0 \\ 0 \end{Vmatrix} \qquad \textit{Eq. 6.2-23}$$

Eq. 6.2-23 is calculated by equations (using EXCEL spreadsheet) presented in previous chapters. Results from Conrad (in global notation) are:

$$\begin{Vmatrix} 0.3999 & -0.3881 & -0.1494 & 0.13758 \\ -0.3881 & 0.4341 & 0.14777 & -0.1938 \\ -0.1494 & 0.14777 & 0.2417 & -0.2401 \\ 0.13758 & -0.1938 & -0.2401 & 0.2963 \end{Vmatrix} * \begin{Vmatrix} T_3 \\ T_2 \\ T_4 \\ T_5 \end{Vmatrix} = \begin{Vmatrix} 0 \\ 0 \\ 0 \\ 0 \end{Vmatrix} \qquad \textit{Eq. 6.2-24}$$

which leads that conductivity coefficients are assembled into the global matrix by following:

$$\begin{aligned} &K_{33} = 0.3999, \;\; K_{32} = -0.3881, \;\; K_{34} = -0.1494, \;\; K_{35} = 0.13758, \\ &K_{23} = -0.3881, \;\; K_{22} = 0.4341, \;\; K_{24} = 0.14777, \;\; K_{25} = -0.1938, \\ &K_{43} = -0.1494, \;\; K_{42} = 0.14777, \;\; K_{44} = 0.2417, \;\; K_{45} = -0.2401, \\ &K_{53} = 0.13758, \;\; K_{52} = -0.1938, \;\; K_{54} = -0.2401, \;\; K_{55} = 0.2963 \end{aligned} \qquad \textit{Eq. 6.2-25}$$

Third element calculations are same as for second element. Node coordinates (in local domain) are:

$$\begin{aligned} &x_1 = -0.027\,m, \quad y_1 = -0.00515\,m, \\ &x_2 = 0\,m, \quad y_2 = -0.00515\,m, \\ &x_3 = -0.0019\,m, \quad y_3 = 0.00515\,m, \\ &x_4 = -0.02754\,m, \quad y_4 = -0.00085\,m \end{aligned} \qquad \textit{Eq. 6.2-26}$$

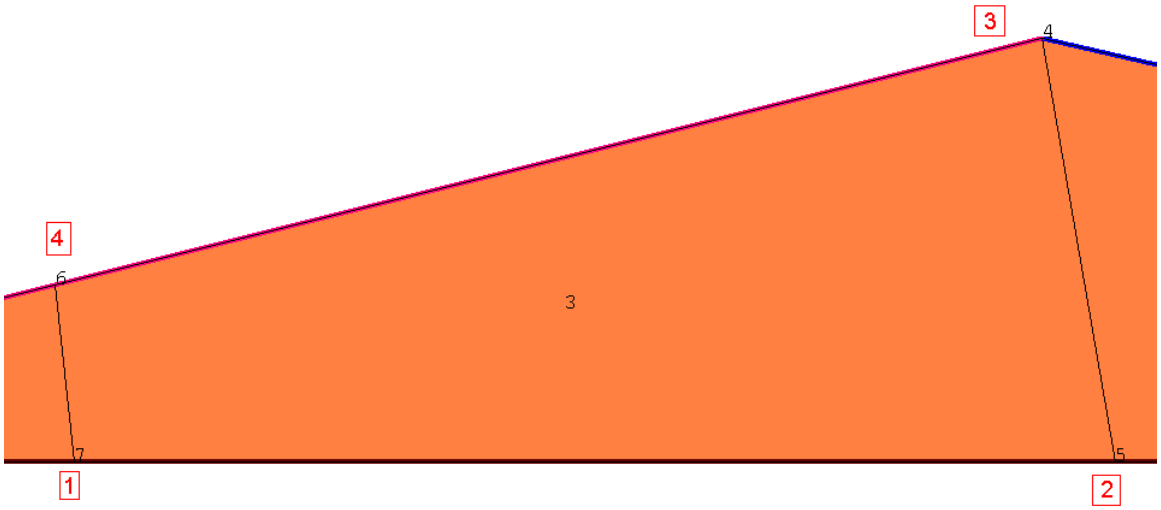


Figure 6-7: Third Element in Local Node Numbering

Using same equations as for previous element, following system of linear equation is obtained:

$$\begin{Vmatrix} 0.372469 & 0.122276 & -0.15791 & -0.33684 \\ 0.122276 & 0.235493 & -0.21067 & -0.1471 \\ -0.15791 & -0.21067 & 0.248329 & 0.120245 \\ -0.33684 & -0.1471 & 0.120245 & 0.363696 \end{Vmatrix} * \begin{Vmatrix} T_1^{III} \\ T_2^{III} \\ T_3^{III} \\ T_4^{III} \end{Vmatrix} = \begin{Vmatrix} 0 \\ 0 \\ 0 \\ 0 \end{Vmatrix} \qquad \textit{Eq. 6.2-27}$$

or in global notation

$$\begin{Vmatrix} 0.372469 & 0.122276 & -0.15791 & -0.33684 \\ 0.122276 & 0.235493 & -0.21067 & -0.1471 \\ -0.15791 & -0.21067 & 0.248329 & 0.120245 \\ -0.33684 & -0.1471 & 0.120245 & 0.363696 \end{Vmatrix} * \begin{Vmatrix} T_7 \\ T_5 \\ T_4 \\ T_6 \end{Vmatrix} = \begin{Vmatrix} 0 \\ 0 \\ 0 \\ 0 \end{Vmatrix} \qquad \textit{Eq. 6.2-28}$$

Eq. 6.2-28 is calculated using excel spreadsheet, and results from Conrad are:

$$\begin{Vmatrix} 0.37246 & 0.12249 & -0.15779 & -0.33716 \\ 0.12249 & 0.23568 & -0.21078 & -0.1475 \\ -0.15779 & -0.21078 & 0.24829 & 0.12028 \\ -0.33716 & -0.1475 & 0.12028 & 0.36428 \end{Vmatrix} * \begin{Vmatrix} T_7 \\ T_5 \\ T_4 \\ T_6 \end{Vmatrix} = \begin{Vmatrix} 0 \\ 0 \\ 0 \\ 0 \end{Vmatrix} \qquad \textit{Eq. 6.2-29}$$

which leads that conductivity coefficients are assembled into the global matrix by following:

$$\begin{aligned} &K_{77} = 0.37246,\ K_{75} = 0.12249,\ K_{74} = -0.15779,\ K_{76} = -0.33716, \\ &K_{57} = 0.12249,\ K_{55} = 0.23568,\ K_{54} = -0.21078,\ K_{56} = -0.1475, \\ &K_{47} = -0.15779,\ K_{45} = -0.21078,\ K_{44} = 0.24829,\ K_{46} = 0.12028, \\ &K_{67} = -0.33716,\ K_{65} = -0.1475,\ K_{64} = 0.12028,\ K_{66} = 0.36428 \end{aligned} \qquad \textit{Eq. 6.2-30}$$

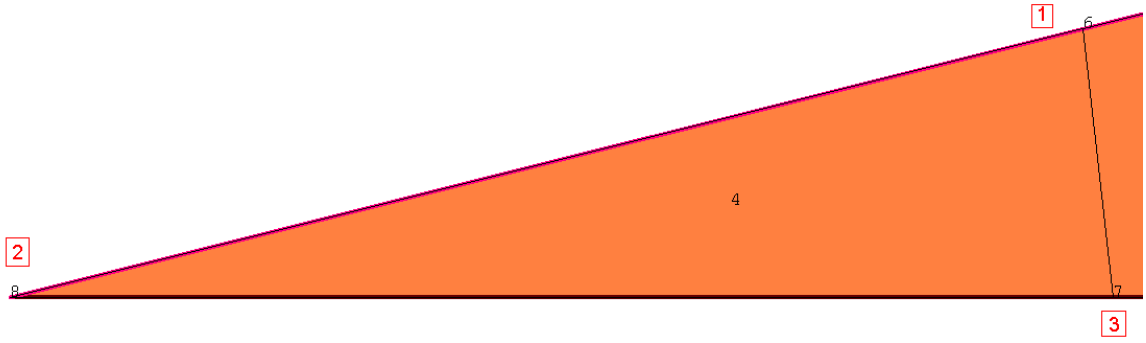


Figure 6-8: Fourth Element in Local Node Numbering

Fourth element is triangular and same equations as for first element are used. Node coordinates (in local domain) are:

$$x_1 = -0.02754\,m, \quad y_1 = -0.00085\,m,$$
$$x_2 = -0.0459\,m, \quad y_2 = -0.00515\,m, \qquad \textit{Eq. 6.2-31}$$
$$x_3 = -0.02702\,m, \quad y_3 = -0.00515\,m$$

and system of linear equation (from EXCEL spreadsheet) is:

$$\begin{Vmatrix} 0.490142411 & -0.013422876 & -0.476719535 \\ -0.013422876 & 0.025800453 & -0.012377576 \\ -0.476719535 & -0.012377576 & 0.489097111 \end{Vmatrix} * \begin{Vmatrix} T_1^{IV} \\ T_2^{IV} \\ T_3^{IV} \end{Vmatrix} = \begin{Vmatrix} 0 \\ 0 \\ 0 \end{Vmatrix} \qquad \textit{Eq. 6.2-32}$$

or in global domain:

$$\begin{Vmatrix} 0.490142411 & -0.013422876 & -0.476719535 \\ -0.013422876 & 0.025800453 & -0.012377576 \\ -0.476719535 & -0.012377576 & 0.489097111 \end{Vmatrix} * \begin{Vmatrix} T_6 \\ T_8 \\ T_7 \end{Vmatrix} = \begin{Vmatrix} 0 \\ 0 \\ 0 \end{Vmatrix} \qquad \textit{Eq. 6.2-33}$$

and results from Conrad (in global domain)

$$\begin{Vmatrix} 0.4903 & -1.3427e-2 & -0.4903 & 1.34271e-2 \\ -1.3427e-2 & 2.5784e-2 & 1.34271e-2 & -2.578e-2 \\ -0.4903 & 1.34271e-2 & 1.00637 & -0.52951 \\ 1.34271e-2 & -2.578e-2 & -0.52951 & 0.54186 \end{Vmatrix} * \begin{Vmatrix} T_6 \\ T_8 \\ T_7 \\ T_7 \end{Vmatrix} = \begin{Vmatrix} 0 \\ 0 \\ 0 \\ 0 \end{Vmatrix} \qquad \textit{Eq. 6.2-34}$$

which is equal with:

$$\begin{Vmatrix} 0.4903 & -1.3427e-2 & -0.4769 \\ -1.3427e-2 & 2.5784e-2 & -0.01235 \\ -0.4769 & -0.01235 & 0.48921 \end{Vmatrix} * \begin{Vmatrix} T_6 \\ T_8 \\ T_7 \end{Vmatrix} = \begin{Vmatrix} 0 \\ 0 \\ 0 \end{Vmatrix} \qquad \textit{Eq. 6.2-35}$$

and this is assembled by following:

$$K_{66} = 0.4903, \;\; K_{68} = -1.3427e-2, \;\; K_{67} = -0.4769,$$
$$K_{86} = -1.3427e-2, \;\; K_{88} = 2.5784e-2, \;\; K_{87} = -0.01235, \qquad \textit{Eq. 6.2-36}$$
$$K_{76} = -0.4769, \;\; K_{78} = -0.01235, \;\; K_{77} = 0.48921$$

All previous matrices are assembled into the Left-Hand side of global matrix. Next step is assembling boundary conditions. This example contains only convection boundary condition. Equations for convection boundary condition are linear type and main equation is Eq. 2.12-6 which solution is assembled into the Left and Right-Hand sides of global matrix. Example contain hot and cold surfaces (Figure 6-9 and Figure 6-10) with following properties:

Cold Surface:

$$h_c = 78\ [\frac{W}{K * m\text{^}2}]$$
$$T_\infty = -16\ ^{O}C$$

Hot Surface:

$$h_c = 32\ [\frac{W}{K * m\text{^}2}]$$

$$T_\infty = 21\ ^O C$$

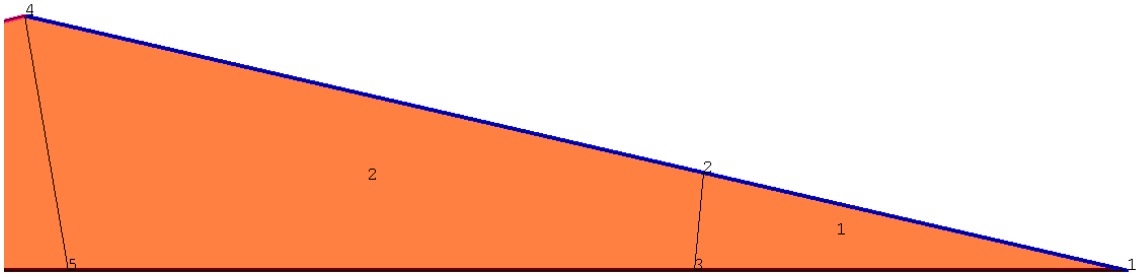


Figure 6-9: Cold Surface

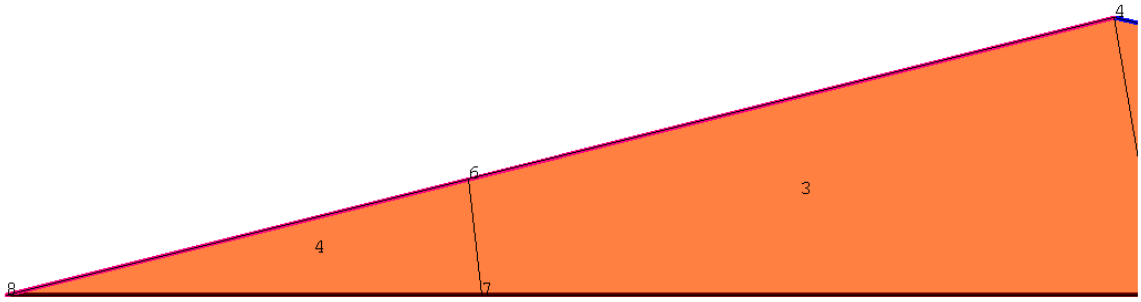


Figure 6-10: Hot Surface

Solutions of matrix equation for cold and hot segment are:

Segment with nodes 1 and 2:

Node coordinates (in global domain):

$$x_1 = 0.0459\,m, \quad y_1 = -0.00515\,m,$$

$$x_2 = 0.02712\,m, \quad y_2 = -0.0011938\,m$$

From EXCEL spreadsheet:

$$\begin{Vmatrix} 0.488316 & 0.244158 \\ 0.244158 & 0.488316 \end{Vmatrix} * \begin{Vmatrix} T_1 \\ T_2 \end{Vmatrix} = \begin{Vmatrix} 188.3559 \\ 188.3559 \end{Vmatrix} \qquad \textit{Eq. 6.2-37}$$

$$\det = 0.009390702$$

and results from Conrad are:

$$\begin{Vmatrix} 0.48831 & 0.24415 \\ 0.24415 & 0.48831 \end{Vmatrix} * \begin{Vmatrix} T_1 \\ T_2 \end{Vmatrix} = \begin{Vmatrix} 188.356 \\ 188.356 \end{Vmatrix} \qquad \textit{Eq. 6.2-38}$$

which is assembled into the global matrix by:

$$\begin{aligned} &K_{11} = 0.48831, \;\; K_{12} = 0.24415, \\ &K_{21} = 0.24415, \;\; K_{22} = 0.48831 \\ &P_1 = 188.356, \\ &P_2 = 188.356 \end{aligned} \qquad \textit{Eq. 6.2-39}$$

Segment with nodes 2 and 4:

Node coordinates (in global domain):

$$\begin{aligned} &x_2 = 0.02712\,m, \quad y_2 = -0.0011938\,m \\ &x_4 = -0.0019\,m, \quad y_4 = 0.00515\,m \end{aligned} \qquad \textit{Eq. 6.2-40}$$

From EXCEL spreadsheet:

$$\begin{Vmatrix} 0.782988 & 0.391494 \\ 0.391494 & 0.782988 \end{Vmatrix} * \begin{Vmatrix} T_2 \\ T_4 \end{Vmatrix} = \begin{Vmatrix} 302.0181 \\ 302.0181 \end{Vmatrix} \qquad \textit{Eq. 6.2-41}$$

$$\det = 0.015057467$$

and results from Conrad are:

$$\begin{Vmatrix} 0.783 & 0.3915 \\ 0.3915 & 0.783 \end{Vmatrix} * \begin{Vmatrix} T_2 \\ T_4 \end{Vmatrix} = \begin{Vmatrix} 302.026 \\ 302.026 \end{Vmatrix} \qquad \textit{Eq. 6.2-42}$$

which is assembled into the global matrix by:

$$\begin{aligned} &K_{22} = 0.783, \;\; K_{24} = 0.3915, \\ &K_{42} = 0.3915, \;\; K_{44} = 0.783 \\ &P_2 = 302.026, \\ &P_4 = 302.026 \end{aligned} \qquad \textit{Eq. 6.2-43}$$

Segment with nodes 4 and 6:

Node coordinates (in global domain):

$$\begin{aligned} &x_4 = -0.0019\,m, \quad y_4 = 0.00515\,m, \\ &x_6 = -0.02754\,m, \quad y_6 = -0.00085\,m \end{aligned}$$

From EXCEL spreadsheet:

$$\begin{Vmatrix} 0.280882 & 0.140441 \\ 0.140441 & 0.280882 \end{Vmatrix} * \begin{Vmatrix} T_4 \\ T_6 \end{Vmatrix} = \begin{Vmatrix} 123.9321 \\ 123.9321 \end{Vmatrix} \qquad \textit{Eq. 6.2-44}$$

$$\det = 0.013166336$$

and results from Conrad are:

$$\begin{Vmatrix} 0.28089 & 0.14044 \\ 0.14044 & 0.28089 \end{Vmatrix} * \begin{Vmatrix} T_4 \\ T_6 \end{Vmatrix} = \begin{Vmatrix} 123.934 \\ 123.934 \end{Vmatrix} \qquad \textit{Eq. 6.2-45}$$

which is assembled into the global matrix by:

$$\begin{aligned} &K_{44} = 0.28089, \;\; K_{46} = 0.14044, \\ &K_{64} = 0.14044, \;\; K_{66} = 0.28089 \\ &P_4 = 123.934, \\ &P_6 = 123.934 \end{aligned} \qquad \textit{Eq. 6.2-46}$$

Segment with nodes 6 and 8:

Node coordinates (in global domain):

$$\begin{aligned} &x_6 = -0.02754\, m, \quad y_6 = -0.00085\, m, \\ &x_8 = -0.0459\, m, \quad y_8 = -0.00515\, m \end{aligned}$$

From EXCEL spreadsheet:

$$\begin{Vmatrix} 0.201139 & 0.10057 \\ 0.10057 & 0.201139 \end{Vmatrix} * \begin{Vmatrix} T_6 \\ T_8 \end{Vmatrix} = \begin{Vmatrix} 88.74773 \\ 88.74773 \end{Vmatrix} \qquad \textit{Eq. 6.2-47}$$

$$\det = 0.009428409$$

and results from Conrad are:

$$\begin{Vmatrix} 0.20113 & 0.10057 \\ 0.10057 & 0.20113 \end{Vmatrix} * \begin{Vmatrix} T_6 \\ T_8 \end{Vmatrix} = \begin{Vmatrix} 88.7455 \\ 88.7455 \end{Vmatrix} \qquad \textit{Eq. 6.2-48}$$

which is assembled into the global matrix by:

$$\begin{aligned} &K_{66} = 0.20113, \;\; K_{68} = 0.10057, \\ &K_{86} = 0.10057, \;\; K_{88} = 0.20113 \\ &P_6 = 88.7455, \\ &P_8 = 88.7455 \end{aligned} \qquad \textit{Eq. 6.2-49}$$

Assembling all previous equations into the global matrix, following matrices are obtained:

$$Left\ Side = \begin{Vmatrix} 5.12E-01 & 2.32E-01 & -1.19E-02 & 0 & 0 & 0 & 0 & 0 \\ 2.32E-01 & 2.23521 & -0.90615 & 0.53927 & -0.1938 & 0 & 0 & 0 \\ -1.19E-02 & -0.90615 & 0.92988 & -0.1494 & 0.13758 & 0 & 0 & 0 \\ 0 & 0.53927 & -0.1494 & 1.55388 & -0.45088 & 0.26072 & -0.15779 & 0 \\ 0 & -0.1938 & 0.13758 & -0.45088 & 0.53198 & -0.1475 & 0.12249 & 0 \\ 0 & 0 & 0 & 0.26072 & -0.1475 & 1.3366 & -0.81406 & 8.71E-02 \\ 0 & 0 & 0 & -0.15779 & 0.12249 & -0.81406 & 0.86167 & -0.01235 \\ 0 & 0 & 0 & 0 & 0 & 8.71E-02 & -0.01235 & 2.27E-01 \end{Vmatrix}$$

$$Right\ Side = \begin{Vmatrix} 188.4 \\ 490.4 \\ 0 \\ 426 \\ 0 \\ 212.7 \\ 0 \\ 88.75 \end{Vmatrix}$$

which lead to temperature solution (in Celsius):

$T_1 = \text{-}14.715\ ^{o}C,\ T_2 = \text{-}18.9567\ ^{o}C,\ T_3 = \text{-}18.7447\ ^{o}C,\ T_4 = \text{-}4.70532\ ^{o}C,$

$T_5 = \text{-}4.50922\ ^{o}C,\ T_6 = 24.8956\ ^{o}C,\ T_7 = 23.5935\ ^{o}C,\ T_8 = 19.638\ ^{o}C,$

and results from Conrad (THERM5) are:

$T_1 = \text{-}14.70458\ ^{o}C,\ T_2 = \text{-}18.99834\ ^{o}C,\ T_3 = \text{-}18.77889\ ^{o}C,\ T_4 = \text{-}4.692912^{o}C,$

$T_5 = \text{-}4.574031\ ^{o}C,\ T_6 = 24.92907\ ^{o}C,\ T_7 = 23.62488\ ^{o}C,\ T_8 = 19.78036^{o}C,$

6.3. Speed of View Factors Calculation

This chapter presents speed of view factor calculation on several examples with different density of net grid.

6.3.1. B928mr04

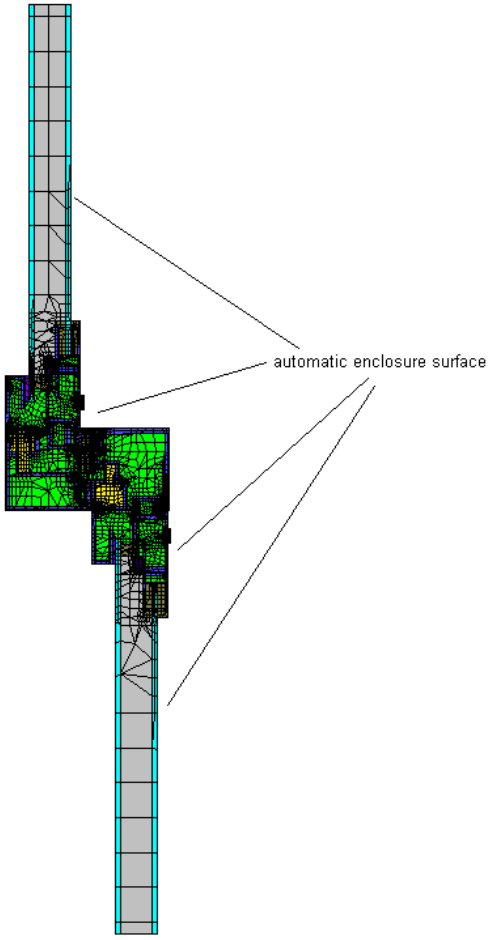


Figure 6-11: Example 1

Table 6.3-1

number of enclosure radiation bc segments	652
number of blocking surfaces	236

Table 6.3-2

n_{xgrid}=	n_{ygrid}=	Calculation Time [seconds]
1	1	12
2	2	12
3	3	8
5	5	10
7	7	6
10	10	6
15	15	5
20	20	4
25	25	5
30	30	6
40	40	7
50	50	11
100	100	40

6.3.2. Trr99_mr_manual

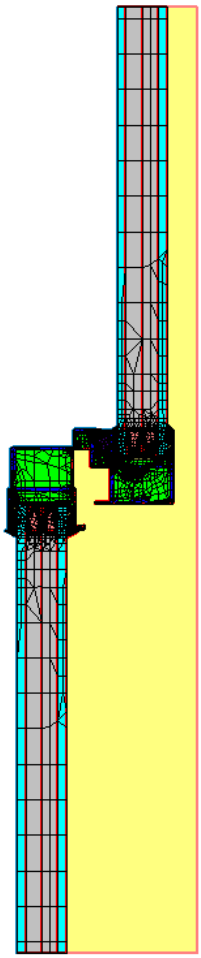


Figure 6-12: Example 2

Table 6.3-3

number of enclosure radiation bc segments	621
number of blocking surfaces	614

Table 6.3-4

n_{xgrid}=	n_{ygrid}=	Calculation Time [seconds]
1	1	41
2	2	21
3	3	15
5	5	12
7	7	11
10	10	9
15	15	9
20	20	9
25	25	10
30	30	12
40	40	15
50	50	24
100	100	86

6.3.3. Pfm01_h_rf_manual

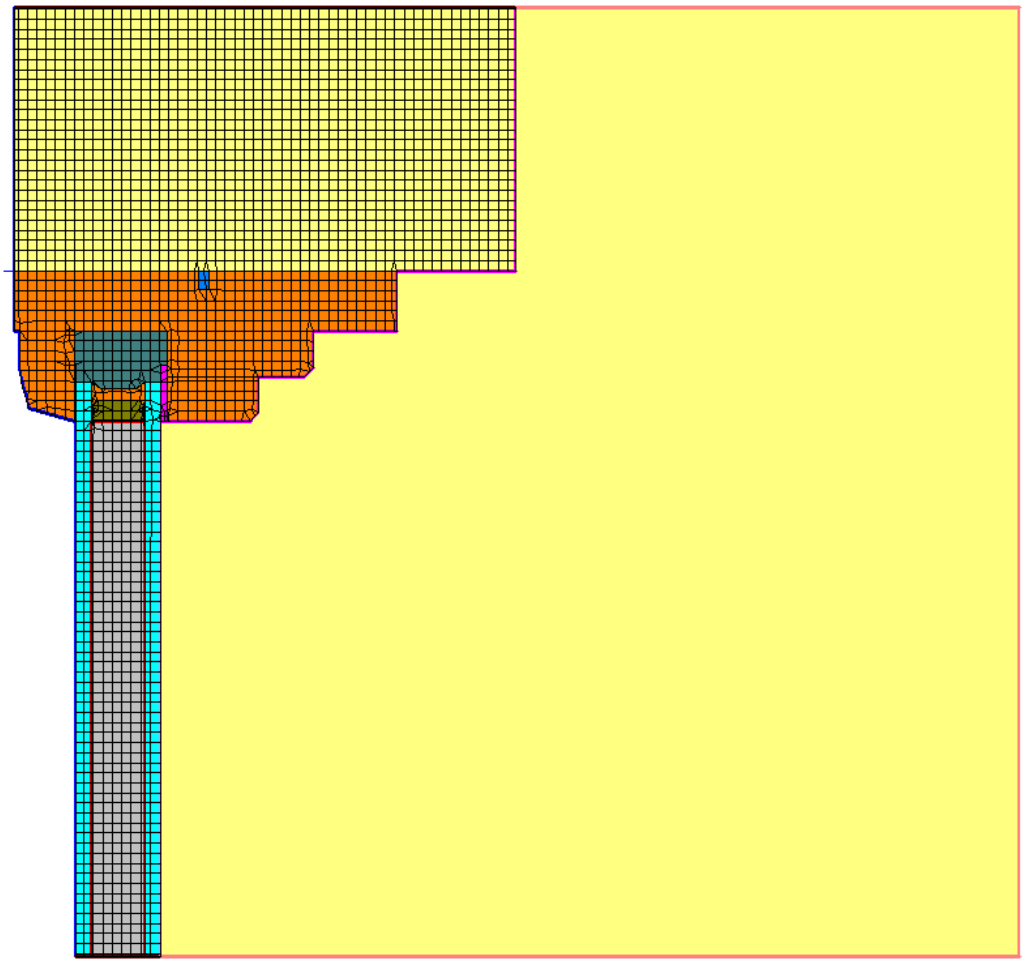


Figure 6-13: Example 3

Table 6.3-5

number of enclosure radiation bc segments	286
number of blocking surfaces	282

Table 6.3-6

n_{xgrid}=	n_{ygrid}=	Calculation Time [seconds]
1	1	15
2	2	11
3	3	10
5	5	7
7	7	4
10	10	4
15	15	4
20	20	6
25	25	6
30	30	7
40	40	12
50	50	17
100	100	63

6.3.4. trr01sill_CI (CI run)

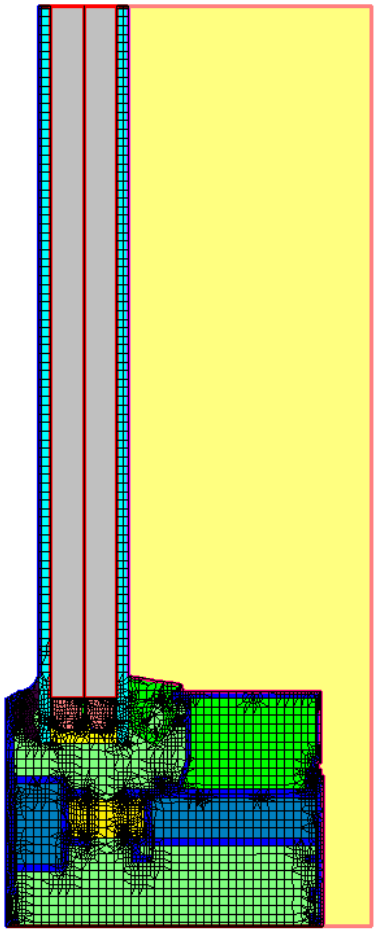


Figure 6-14: Example 4

Table 6.3-7

number of enclosure radiation bc segments	1297
number of blocking surfaces	551

Table 6.3-8

n_{xgrid}=	n_{ygrid}=	Calculation Time [seconds]
1	1	79
2	2	51
3	3	32
5	5	26
7	7	18
10	10	17
15	15	14
20	20	17
25	25	20
30	30	24
40	40	36
50	50	53
100	100	191

6.3.5. trr99_mr_Cl (Cl run)

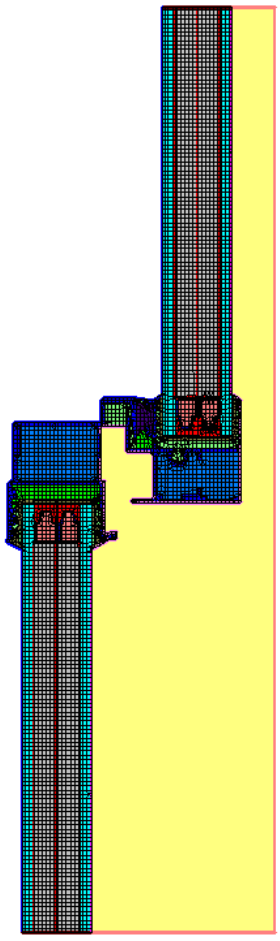


Figure 6-15: Example 5

Table 6.3-9

number of enclosure radiation bc segments	1279
number of blocking surfaces	801

Table 6.3-10

n_{xgrid}=	n_{ygrid}=	Calculation Time [seconds]
1	1	256
2	2	135
3	3	95
5	5	73
7	7	59
10	10	49
15	15	41
20	20	43
25	25	47
30	30	54
40	40	76
50	50	107
100	100	447

REFERENCES:

[1] ISO. 2002. "ISO/FDIS 15099 - Thermal Performance of Windows, Doors and Shading Devices – Detailed Calculations", International Standards Organization. Geneva, 2002.
[2] Hayrettin Kardestuncer, "Finite Element Handbook"
[3] J.N.Reddy, D.K.Gartling, "The Finite Element Method in Heat Transfer Fluid and Fluid Dynamics"