NUMERICAL ANALYSIS OF TEMPERATURE DEVELOPMENT IN CONCRETE AT AN EARLY AGE

Anthony Patini

A research report submitted to the Faculty of Engineering and the Built Environment, University of the Witwatersrand, Johannesburg, in partial fulfilment of the requirements for the degree of Master of Science in Engineering

Johannesburg, 2011

DECLARATION

I declare that this research report is my own unaided work. It is being submitted for the Degree of Masters of Science in Engineering at the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination at any other University.

_____

Anthony Patini

_____ day of _____

ABSTRACT

The exothermic reaction associated with hydrating Portland Cement releases a significant amount of heat within concrete elements. These raised temperatures could give rise to thermal cracking which is a function of temperature differential and concrete stiffness.

In recent years computer-based modelling has become an intrinsic part of engineering. It has been employed to simulate the rise in temperature and distribution of heat within concrete elements. The prediction model developed in this project is based on the numerical finite element theory in combination with heat evolution curves obtained from adiabatic calorimetry. Predicted results are compared with two sets of measured data and comparisons are drawn. This model is also evaluated against the pre-existing finite difference numerical simulation (Ballim, 2004a). The finite element simulation provides engineers with temperature differentials from which generalised rules for cracking potential may be applied.

The implemented finite element model provides superior predictions to those of existing simulations and allows for future developments due to the advanced capabilities of the finite element theory.

ACKNOWLEDGEMENTS

The work described in this report was carried out at the School of Civil and Environmental Engineering at the University of the Witwatersrand, Johannesburg.

I would like to thank my supervisor, Professor Yunus Ballim, for his advice, guidance and patience.

I would also like to extend my thanks to the staff of the School of Civil and Environmental Engineering, especially Mr. David Blitenthall, for his support and encouragement towards completion of this project.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

ε = surface emissivity
Γ = boundary of continuum
η = x coordinate in a local coordinate system
σ = Stefan-Boltzmann constant
ρ = density of concrete
Ω = continuum of element
ξ = y coordinate in a local coordinate system
B = temperature gradient interpolation function
b = specification for boundary conditions
C = unknown functions of the independent variable
c = specific heat capacity
E = activation energy
G = differential operator
h = convective heat transfer coefficient
i = arbitrary finite element node number
j = arbitrary finite element node number
k = thermal conductivity
M = maturity of concrete
m = number of unknowns (counter)
N = interpolation functions
n = cosines of the outward normal vector
Q^I = internal heat generation rate per unit volume
$Q^I{}_M$ = maturity heat rate
q = heat flow rate vector
R = residual
R_g = universal gas constant
T = temperature
T_s = surface temperature
T_e = atmospheric temperature
T_{max} = maximum daily atmospheric temperature
T_{min} = minimum daily atmospheric temperature
T_n = temperature at current time step
T_{n+1} = temperature at next time step
t = time
t_{20} = equivalent maturity time
t_d = clock time of day
t_w = time at which minimum overnight temperature occurs
u = field variable
W = weighting functions

1 INTRODUCTION

1.1 BACKGROUND

The determination of heat evolved and temperature distribution in hardening and hydrating concrete is essential to designers and contractors. Potential for thermal cracking, deformation control and the evaluation of concrete strength development are of importance.

The hydration of cement in concrete is an exothermic reaction which liberates up to 500 joules of heat per gram of cement (Neville, 1981). Temperature variations within a concrete element can lead to the development of thermal stresses and strains which can result in cracking. The relatively low thermal conductivity of a concrete mass slows the rate of heat dissipation to the surroundings. Consequently a significant temperature rise is observed in large elements at an early age. Ambient environmental conditions may further contribute to heat gain in the concrete by solar radiation or high temperature curing. With an increase in concrete stiffness (i.e. gain in Young's Modulus), fluctuations in temperature cause strains to be induced in the concrete. If these thermally applied strains are greater than the allowed, deformation that results in cracking within the concrete element will occur. Conversely, if the rate of strain during the hydration period is larger than the creep capacity, restraint cracking arises. In mass concrete elements at an early age (defined as concrete after the first few days subsequent to casting) thermal cracking is a common problem that requires specific consideration at the design and construction phase.

1.2 THERMAL CRACKING OF CONCRETE

During early hydration (the first few days after casting a concrete element) the core of the concrete element increases in temperature, while the surface remains relatively cool. This disparity is associated with heat dissipation to the

surrounding environment (Ballim, 2004a). The internal concrete is hotter, resulting in a greater potential for expansion than for the cooler surface concrete. This probable difference produces tensile stresses and strains in the surface zone. If these are greater than the tensile capacity of the early age concrete, surface cracks will occur. Generally these cracks are reasonably shallow due to the steep temperature gradient existing close to the surface. This temperature differential is attributed to the low thermal conductivity of concrete. The surface cracks may close substantially after the cooling phase but could potentially have deleterious effects on durability of the concrete element. Once the concrete has reached its maximum temperature, the core of the element will enter a cooling phase as the liberated heat is dissipated to the surrounding environment. The cooler surface zone now acts as a restraint to the thermal shrinkage of the internal concrete. Thus the internal concrete is subjected to tensile stresses and strains which could lead to cracking at the core of the element. This internal cracking is more substantial than the surface cracks. In massive elements such as dam walls, the formation of internal cracks could result in leakage and thus failure of the wall.

The rate of a chemical reaction increases with an increase in temperature for exothermic processes. Therefore cement hydrates more rapidly when the temperature is elevated. Concrete at the core of a structure will experience a rise in thermal energy faster than that near the exposed surfaces at early age. Internal concrete cannot dissipate the induced heat efficiently due to the high thermal inertia of concrete. With this build up of heat, the internal concrete gains strength and stiffness whilst the creep capacity reduces at a rate higher than that of the surface concrete. As the internal concrete undergoes the cooling or contraction phase through the transfer of heat to the surrounding environment, it experiences a tensile load due to the restraint provided by the surface layer. Conversely, the surface stratum also experiences a compressive load on account of the contraction of the inner core. Thermal shrinkage can be resisted by the surface concrete as the ratio of tension to compression strength is approximately 1:10. The surface cracks that appear during early hydration will thus tend to close after the cooling phase as a result of the induced compressive stresses. Although the internal concrete can be

said to be stronger (i.e. due to the faster rate of hydration whereas the surface zone loses heat and moisture to the surrounding environment that hinders the advance of hydration and strength development) it is subjected to a tensile strain that can exceed the tensile strain capacity of the concrete, thus causing cracks to develop.

Depending on the dimensions of the concrete element and other prevailing conditions, namely concrete mixture and boundary conditions, cracks appear over weeks, months or even years, in extreme cases, after the concrete element was cast. Factors affecting temperature development and distribution can be classified as either intrinsic or extrinsic conditions. Intrinsic conditions include; binder and aggregate type, quantity and grading, water to cement ratio and admixture and extender type and quantity. While extrinsic conditions comprise; formwork type and removal time, construction sequencing, initial concrete temperature, ambient temperature, radiation, solar radiation, size of the concrete element, wind and any possible thermal insulation.

1.3 TEMPERATURE PREDICTION

Heat distribution within a concrete element is generally modelled using numerical techniques such as the finite difference method or the finite element method. Both techniques attempt to predict the flow of heat in concrete through the solution of the Fourier equation, generally in two dimensions.

The finite difference method is one of the oldest and simplest methods used to solve differential equations (Cope et al., 1984; Ugural, 1999). The domain in which the solution is sought is replaced by a finite set of points forming a regular grid. Approximate temperature values are then sought for each point. These values are required to satisfy finite difference equations in terms of partial difference quotients or through direct heat flow considerations. The equations for each point constitute a system of equations, the solution to which yields values for temperature. Provided exact boundary and initial conditions have been entered, solutions converge towards experimental solutions as the spacing between the grid

points contract. Despite the relative ease of formulation of the system of differential equations, the finite difference method is limited by the fact that irregular geometric shapes, unusual boundaries and extrinsic conditions cannot be modelled.

With the finite element method the domain is replaced with an assemblage of discreet elements rather than a grid of points. The temperature is then approximated over each individual element by way of an assumed function. This interpolation function is defined in terms of the temperature at specified points called nodes. Nodes are typically located on element boundaries where individual elements are connected to one another. The governing differential equation and its individual matrices can then be formulated for every element using the method of weighted residuals. Matrices associated with each individual element can then be assembled to form a system of equations for the entire domain. Consequently, the temperature throughout the domain can then be solved using this system of equations.

The fundamental difference between the two numerical techniques is that the finite difference method provides a point-wise approximation while the finite element method generates a piecewise approximation for the governing differential equations.

Despite the more complex formulation of differential equations, the finite element method has numerous advantages over other methods, the most important being that irregular geometric shapes (elements can be put together in a variety of ways), unusual boundaries and extrinsic conditions can be represented. Boundary conditions, selected as the most appropriate conditions for common problems, are convection and radiation.

1.4 SCOPE AND LAYOUT OF REPORT

This research project will attempt to improve and expand upon a pre-existing two-dimensional model (Ballim et al., 2005) which predicts the temperature development and distribution in setting concrete. Ballim (2005) employs the finite difference method as an approximate mathematical solution while this research report will endeavour to utilise the finite element method. The two-dimensional finite element numerical model uses various intrinsic and extrinsic factors upon which heat liberated and distributed are dependent.

The computer based package, Matlab®, was employed to generate the finite element numerical model. Matlab® is a high-level language (mainly applicable to engineering and science) and interactive environment that enables users to perform computationally intensive tasks faster than with conventional programming languages such as C, C++, or Fortran.

The main objective of this research project is to develop a numerical model that will be able to predict accurately the heat liberated in hardening concrete in order to assess the possibility of thermal cracking in the final structure. This proposed model could be used in the selection of both intrinsic and extrinsic factors, thus ensuring that thermal cracking does not occur or is at least controlled, limited or reduced.

Results generated by the finite element numerical model, could be used to predict the stresses and strains within a mass concrete element and thus the cracking potential. This research project is seen as the first step in an overall design tool requiring the accurate prediction of temperature gradients in mass concrete elements.

2 LITERATURE REVIEW

2.1 INTRODUCTION

Concrete is a material extensively used in the construction industry, consisting of a solid particulate substance known as aggregate (typically different types of crushed rock and gravel) that is cemented together using a binder and water. Cements utilised in the construction industry generally display similar chemical compositions to one another. They contain limestone and siliceous clays as the principal ingredients.

Assyrian and Babylonian civilisations were among the earliest recorded users of clay as the binding material in ancient times. In 1756, John Smeaton is credited with the creation of the first modern concrete which he achieved by adding pebbles as a coarse aggregate and mixing powered brick into the cement. This was the earliest discovery of the benefits of limestone as a cementitious binder (White, 1977). In 1824, Joseph Aspdin invented Portland Cement, which has remained the principal binder used in concrete production, although in present times cement composition is chemically far different. Aspdin created the first artificial cement by burning ground limestone and clay together at high temperatures. This burning process modified the chemical properties of the materials, resulting in stronger cement than that which plain crushed limestone would generate (Ghosh, 1991).

Following Joseph Aspdin's patent, vast literature concerning the characteristics of cement and concrete have been published. The fundamental concept relevant in this research report is the reaction between cement and water. These chemical components combine in an exothermic process to produce a cement hydrate that releases thermal energy known as "heat of hydration". The heat evolving properties of cement are thoroughly documented from practical experience and theoretical considerations (van Breugel, 1998). It is well known that early age temperatures and temperature-induced stresses have far-reaching implications for

the durability, functionality and overall performance of concrete structures. Control over these temperatures and the associated thermal stresses are crucial in the development of concrete with improved properties.

Numerical modelling techniques can be used to address suspected thermal quality issues. These techniques are used to predict the heat liberation and distribution within a concrete element in order to assess the risk of early age cracking. Tetmayer (1883) was one of the first researchers to apply experimental techniques to document the significance of temperature effects in early age concrete. In the 1920's and 1930's several researchers discussed the issue of thermal problems in relation to the optimization of concrete casting sequences. Their findings were applied in the reduction of maximum hydration temperatures in mass concrete structures to predict and prevent thermal cracking.

Harrison (1981) also corroborates the finding that early age thermal cracking associated with concrete quality is caused by resistance to contraction on cooling. During the hydration process thermal energy is retained in the concrete which results in a rise in temperature within the body of the concrete element. As this heat begins to dissipate to the surrounding environment, the restraint provided by the stiff surface layer induces the potential for thermal cracking.

Recently, the focus of research has turned towards the pre-construction phase and how concrete mixture design affects the amount of heat liberated and the final temperature reached within a concrete element. Developments have been made with the introduction of simulation models which are able to predict the rise in temperature within a concrete element relative to concrete mixture composition and environmental conditions (i.e. intrinsic and extrinsic factors). One piece of information required by these simulation models is the rate of heat liberated for a particular concrete mixture. Various experimental methods have been introduced to determine this rate. Most of the computer models that have been generated to address thermal problems in hardening concrete require the adiabatic hydration curve (heat liberated plot) of the proposed concrete mixture (Koenders et al., 1994).

2.2 ADIABATIC CALORIMETER

Several methods of measuring the heat of hydration for specific concrete mixtures are available, namely:

- The Heat of Solution Method
- Adiabatic Calorimetry
- Isothermal Calorimetry
- Conduction Calorimetry

Each of these methods measure the heat released from a specific concrete mixture from which the heat of hydration can be calculated. Only the adiabatic calorimetry method will be discussed as it was the only test facility available to the investigation reported in this research report. Interested readers should consult relevant literature for further information regarding the other possible methods of measuring heat of hydration.

As a means of determining the heat of hydration of a specific concrete mixture, adiabatic calorimetry has significant advantages over other calorimetric methods (Morabito et al., 1993). An adiabatic calorimeter is an apparatus used to examine a runaway reaction. A runaway reaction in this instance can be defined as a reaction in which none of the energy produced by the reaction is lost and allows the maximum rate of hydration to be obtained. Since the calorimeter runs in an adiabatic environment (no heat is lost or gained in the system), any heat generated by the specific concrete sample under test causes the sample to increase in temperature, thus fuelling the reaction. No adiabatic calorimeter is truly adiabatic as some heat will always be lost by the sample to the sample holder. This method closely replicates the hydration temperatures in actual mass concrete elements.

Figure 2.1 shows the components of the adiabatic calorimeter test apparatus. The principle of the apparatus is to eliminate the effects of the surrounding environment in order to accurately assess the heat produced by a specific concrete mixture.

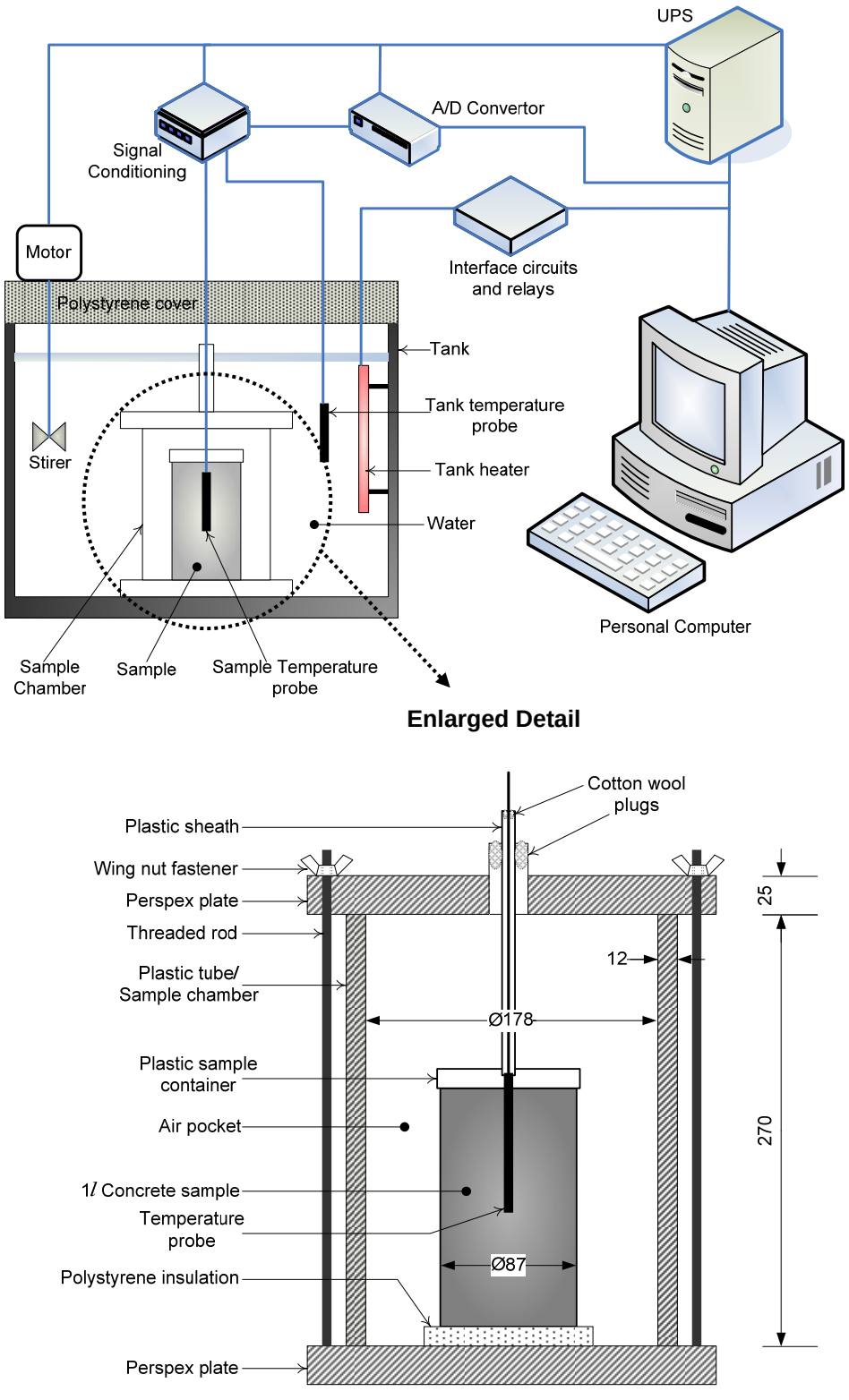


Figure 2.1 Adiabatic Calorimeter schematic arrangement (after Greensmith, 2005)

Figures 2.2 shows an example of a temperature plot produced from the adiabatic calorimeter experiment and Figure 2.3 is an example of the heat evolution curve

calculated from the experimental results. The heat evolution data is fundamentally the key input data in most macro-level numerical modelling techniques.

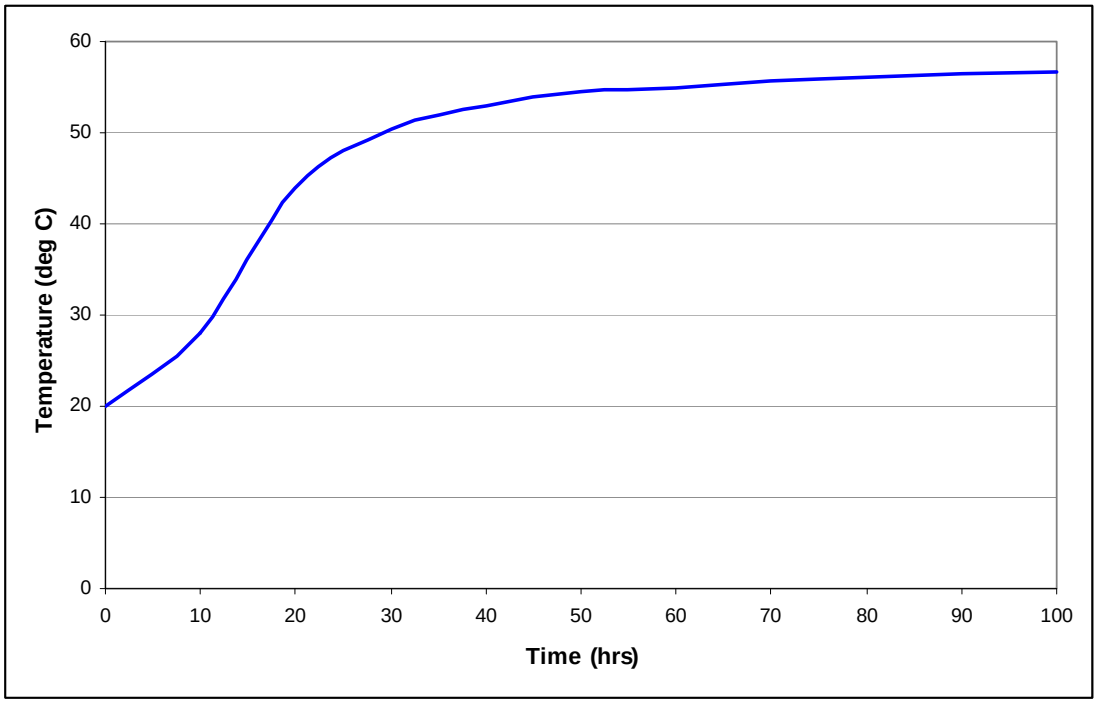


Figure 2.2 Example of a hydration temperature rise curve obtained from an Adiabatic Calorimeter experiment

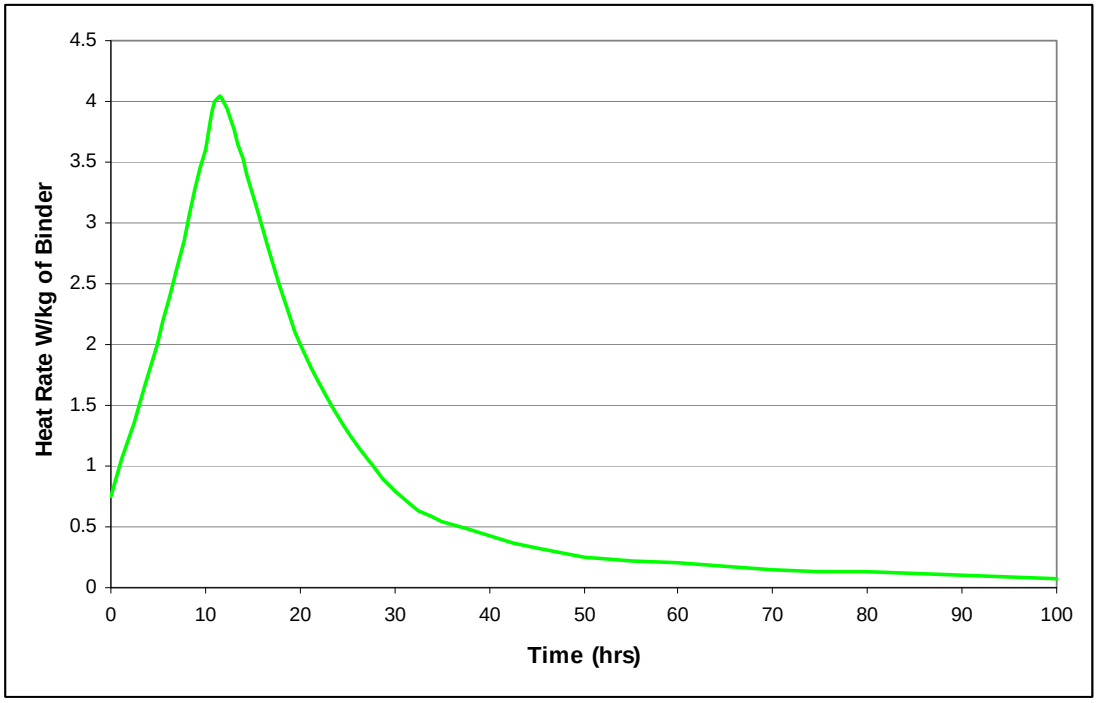


Figure 2.3 Example of a rate of heat evolution curve obtained from an Adiabatic Calorimeter experiment

2.3 MODELLING TEMPERATURES IN CONCRETE ELEMENTS

An accurate assessment of the rate of heat evolution from hydrating cement over time is required to correctly predict the temperature profile within a concrete element. Previous researchers have provided direction on the rate of heat evolution for use as input into numerical temperature prediction models. Estimated and generalised values of the total heat released over the early hydration period for different binder types (Addis, 1986) or cement constituents (Scanlon et al., 1994) have been published. Guide equations were introduced by Wang (1994), following which prediction models founded on the chemistry and crystallography of cement (Maekawa et al., 1999), referred to as hydration models were also developed to present the heat released from hydrating cement. These hydration models are discussed following which more reliable laboratory-based measurements of the rate of heat evolution coupled with numerical analysis techniques are reviewed.

2.3.1 Cement Hydration Models

Typically, cement hydration models deal directly with the microstructural elements found in specific concrete mixtures. The aim of these microstructural models is to predict the time-dependent behaviour of concrete with respect to heat production, thermal deformation, autogenous and drying shrinkage, creep behaviour. The models simulate the cement particles as spheres packed into an "analysis box" of known size and employ periodic boundary conditions. The positioning of the cement spheres is carried out by a specific mathematical or statistical process. The four most commonly discussed microstructural models are:

- HYMOSTRUC (Koenders et al., 1994)
- CEMHYD3D (Bentz et al., 1993)
- DuCOM (Maekawa et al., 1999)
- CCBM (Maruyama et al., 2007)

The first model, HYMOSTRUC, predicts the degree of hydration utilising the information of water to cement ratio, chemical components of cement, particle size distribution and curing condition. The programme uses a statistical approach in obtaining the particle size distribution. The programme functions by assuming a spherical shape for the cement particles that grow during the hydration reaction and interconnect or make contact with other particles. The rate of hydration is determined by the amount of available C_3S and C_2S, taking into account the density of the hydrates, residual water and relative humidity in the pore structure.

CEMHYD3D creates a local chemical reaction rule which is a function of the density of substances in target and neighbouring cells. This prediction model can take into account many chemical reactions at once. This method measures the accumulation of the reactions in local cells to represent the total reaction of the cement paste and provides a 3-dimensional cement paste structure. The cement particles, once positioned, can be optionally moved closer together to represent a flocculated state or can be placed during positioning in a manner that achieves a minimum separation. The method has successfully been interfaced with many engineering problems.

DuCOM is a finite element programme developed to determine the hydration of concrete at any given time step and boundary conditions. The heat generation rate per unit volume of blended cements is represented as the sum of the specific heat generation rates of the individual clinker components. The temperature dependent heat generation ratio of each clinker (unrefined cement) component is based on the Arrhenius law. It also provides results for porosity shrinkage and strength gain. With the use of finite elements, the heat transfer and distribution occurs on a macro scale rather than the microstructural scale used to model the time dependent behaviour of the cement. The limitation of the programme is that the pore structure is simulated by a consistent distribution of average sized binder grains. Blaine air permeability values are then used to calculate the distance between grains.

CCBM is based on a kinetic model expressed as a single equation composed of four rate-determining coefficients which represent the rate of surface solution, formation and destruction of the initial impermeable layer, and the resulting diffusion-controlled process. The model assumes the cement as spherical bodies that keep their shape during hydration. The model also focuses on the problem of early age thermal cracking in concrete elements. The model parameters of hydration are determined according to the experimental data of X-ray diffraction/Rietvelt analysis.

In the above models, aggregate influence may be considered but the representation is often in a weak fashion using a surface of voxels (volumetric pixels) a single layer thick. The assignment of spherical shapes to the cement particles generates a continuum far removed from reality. The angular shape associated with the impact during grinding is far more regularly found in cement particle analysis. This obvious disparity between the actual and modelled shapes constitutes an initial error for all the models concerned. Thus the attempts to modify models to incorporate more realistic cement particle shapes are underway. A further assumption that departs from reality is the fact that fresh cement paste tends to be densely flocculated unless heavily superplasticized. Only CEMHYD3D can account for this flocculation.

2.3.2 Temperature Prediction Models

The hydration, microstructure, moisture states and time-dependent deformation in fine pores have been extensively studied over the past decades, but difficulty has been observed in verifying the theories particularly at microstructural levels (Mihashi, 2007). Thus, it has been proposed that modelling must be postulated from macroscopic and empirical information.

The macrostructural models and associated numerical methods used for temperature prediction of importance to this research report are:

- Finite Difference Model (Ballim and Graham, 2005)
- Finite Element Model (Wang and Dilger, 1994)

The Finite Difference Model (Ballim and Graham, 2005) is a two-dimensional model as it assumes that the structure's length dimension is much larger than the width or thickness. This simplifies a three-dimensional structure to two dimensions by assuming that the heat does not vary over the length of the structure. The model utilizes the finite difference method of analysis. The distribution or flow of heat through a two-dimensional structure can be described by the Fourier equation (Holman 1986):

$$\rho c \frac{\partial T}{\partial t} = k\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right) + Q' \qquad (2.1)$$

where, T = temperature; t = time; k = thermal conductivity; Q' = rate of internal heat evolution; ρ = density of the concrete; c = specific heat capacity; x, y = coordinates at a particular point in the structure.

The finite difference model resolves much of the complexity of heat models by using as input, the results of a heat rate determination using an adiabatic calorimeter together with the Arrhenius maturity function. This indicates the rate and degree of hydration at any position and time in the concrete element, based on the time-temperature history at that point. This is executed by expressing the heat evolved as measured in the adiabatic test in terms of "maturity heat rate" as a function of the cumulative maturity, rather than a time rate. The maturity heat rate is expressed as:

$$Q'_M = \frac{dQ}{dM} \qquad (2.2)$$

The time based heat rate as required by the Fourier equation (equation 2.1) is then determined as follows:

$$Q' = \frac{dQ}{dt} \tag{2.3}$$

Applying the chain rule from differential calculus:

$$Q' = Q'_M \frac{dM}{dt} \tag{2.4}$$

The finite difference model maintains a measure of both the development of maturity and the time based rate of change of maturity at each point under consideration in the concrete element. An appropriate maturity heat rate is selected during each calculation step based on the cumulative maturity at the position under consideration. The time based rate of change of maturity is multiplied by the appropriate maturity heat rate to yield the time based heat rate required by equation 2.1. This approach allows the rate of heat evolution determined from the adiabatic calorimeter test to be expressed in a form that is independent of the starting temperature of the test. It has been shown that the starting temperature of adiabatic calorimeter tests greatly influences both the magnitude and time distribution of the heat rate (Ballim and Graham, 2003).

The Arrhenius maturity function was used to normalise the results obtained from a adiabatic calorimeter test for a specific concrete sample, and takes the following form:

$$t_{20} = \sum_{i=1}^{i=n} exp\left[\left(\frac{E}{R_g}\right)\left(\frac{1}{293} - \left(\frac{1}{273 + 0.5(T_i + T_{i-1})}\right)\right)\right] \cdot (t_i - t_{i-1}) \tag{2.5}$$

where, t_{20} = equivalent maturity time (in hours); E = activation energy; R_g = universal gas constant; T_i = temperature (°C) at the end of the i^{th} time interval, t_i

Previous research into the model shows that it is fairly accurate but that it has some limitations. It showed good correlation with the temperature as measured in a laboratory test. A rectangular concrete block which had a length dimension significantly larger than the other two dimensions, instrumented with thermal probes was cast in the laboratory. The results obtained were satisfactory with a maximum temperature difference of only 2°C between the measured and the predicted results.

Ballim and Graham (2005) identify the following limitations with the model:

- The model cannot deal with structures that are not rectangular.
- The problem of early drying on the exposed surfaces and in general the definition of the boundary conditions needs to be overcome and more precisely defined to include the effects of sunlight, humidity, wind and cloud cover.
- The model cannot allow for:
    1. Sequential construction (i.e. casting fresh concrete onto concrete that has not dissipated all of its hydration heat);
    2. The analysis of the concrete temperature where conduits, which have chilled water flowing through them, are cast into the concrete in order to reduce the overall temperature within the concrete element;

The finite element method of analysis has previously been implemented to calculate the heat liberated and distributed through a concrete element (Wang et al., 1994). However, the appropriateness of their approach for determining the rate of heat evolution is questionable. The following approach was used:

$$\partial Q/\partial t = 0.5 + 0.54M^{0.5} \quad for \quad M \leq 10 \text{ hours} \tag{2.6}$$

$$\partial Q/\partial t = 2.2 \exp[-0.0286(M - 10)] \quad for \quad M \geq 10 \text{ hours} \tag{2.7}$$

where M is the maturity of the concrete in hours with reference to concrete cured at 20°C.

When the temperature of the test concrete is constantly equal to 20°C, the maturity time is equal to the clock time. The appropriateness of equations 2.6 and 2.7 for a temperature prediction model is problematic since:

- The maximum heat rate is fixed at 2.2W/kg. This is not appropriate since measured heat rates have been shown to vary significantly from this value (Ballim and Graham, 2004, 2009), particularly when cement extenders are used in concrete.
- The equations are not true if the environmental conditions cause the temperature within the concrete to drop to -10°C. At this temperature hydration is deemed to stop, whereas the above equations indicate a finite rate of heat evolution.
- The approach ignores the temperature at which the adiabatic test was conducted. The temperature at which an adiabatic test is started greatly influences the rate of hydration.

The two-dimensional model that was undertaken in this research project uses the finite element method of calculation rather than the finite difference method due to its number of shortcomings. The finite difference method is unsuitable for systems with irregular geometry, unusual boundary conditions, or heterogeneous compositions. The finite element method provides an alternative that is better suited to such systems as it breaks down a structure's cross section into small elements (either rectangular or triangular shaped elements) to create a better approximation of irregularly shaped sections. Further, values of the unknown variable can be generated continuously across the entire solution domain rather than at isolated points.

2.4 Evolution of the Finite Element Method

The principles of finite element theory, as it is known today, can be dated back as far as 250 B.C. The Greek mathematician Archimedes calculated the ratio of a circle's circumference to its diameter by the restructuring of a circle as a limit of inscribed regular polygons. This piece-wise solution is refined to an acceptable

approximation as the number of regular polygons is increased. Figure 2.4 shows the principle of this piece-wise solution and Table 2.1 indicates how the solution is refined towards the exact value of π as the number of regular polygons is increased.

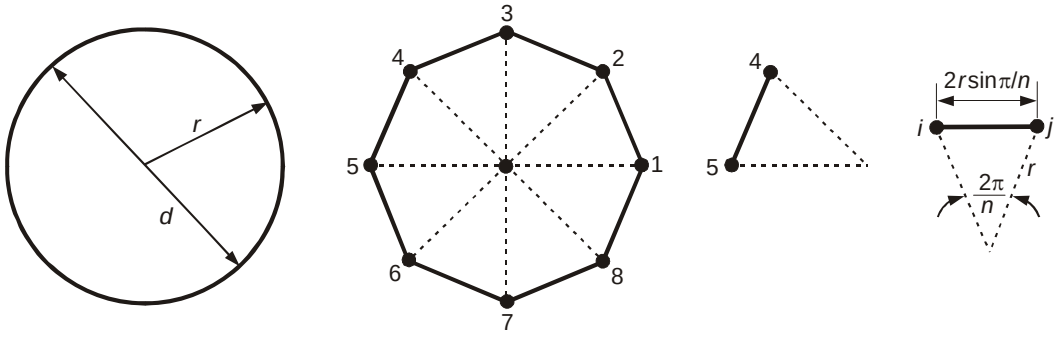


Figure 2.4 Archimedes piece-wise solution to calculate π

Table 2.1 Archimedes piece-wise solution to calculate π

n	$\pi_n = 2rn\sin(\pi/n)/d$
1	0,0000
2	2,0000
4	2,8284
8	3,0615
16	3,1214
32	3,1365
64	3,1403
128	3,1413
256	3,1415
512	3,1416
n = 512 to 9 decimal places	3.141572940
Exact π to 9 decimal places	3,141592654

It is difficult to document the exact origin of the finite element method, because the basic concepts have evolved over a period of 150 years or more.
Courant (1943) was the first to develop the Finite Element Analysis by utilizing the Ritz method of numerical analysis and minimization of variational calculus to obtain approximate solutions to vibration systems. Soon thereafter, a paper

published by Turner (1956) determined a broader definition of numerical analysis. The paper centred on the "stiffness and deflection of complex structures".

The term finite element was first created by Clough (1960). In the early 1960's, scientists used the method for the approximate solution of problems in stress analysis, fluid flow, heat transfer, and many other areas. The first book published on the finite element method was by Zienkiewicz (1967).

In 1970's, Finite Element Analysis was limited to expensive mainframe computers generally owned by the aeronautics, automotive, defence, and nuclear industries due to the sizeable processing power required by the analysis. Following the decline in the cost of computers and the increase in computing power, the finite element method has developed into a useful numerical analysis technique. Present day supercomputers are now able to produce accurate results for all kinds of parameters. Thus continued research into different types of elements and convergence studies are presently under development.

2.5 THERMAL CRACKING OF CONCRETE

In attempting to control the risk of thermal cracking in concrete elements, knowledge of the expected temperature rise during hydration of cement is desirable (Morabito, 1998). Due to increased accessibility of aggressive agents like chloride ions, CO_2 and other harmful agents into concrete as well as increased potential leakage of liquid retaining structures, the effect of thermal cracking can lead to reduction of the durability and serviceability of concrete. The long term effect of thermal cracking can therefore be of significant consequence and control is essential.

Harrison (1981) discusses practical solutions to prevent early age thermal cracking in concrete. Section thickness, cement type, concrete mixture proportions, formwork, insulation, restraint and ambient conditions were

discussed. Data is provided with respect to concrete tensile strain capacity and anticipated heat evolution of different binder types.

Emborg (1992) computes the risk of cracking with respect to different structural and environmental scenarios. The proposed procedures have regularly been used in Sweden for producing concrete in massive structures intended to be free of cracks. The emphasis on the theory is placed on calculations pertaining to the cracking risk. The critical maximum temperature differential is limited to 20 degrees Celsius. A useful finding by Emborg is the relationship between ambient air temperature and initial concrete temperature in relation to surface cracking in very early age concrete. He notes that favourable conditions for placing of concrete exist when the ambient temperature is much lower than the placing temperature of the concrete. This condition decreases the risk of early age surface cracking.

Taylor and Addis (1994) also confirm that the maximum allowable temperature differential should not exceed 20 degrees Celsius for ordinary concrete structures. An important fact to note is that during the temperature rising phase of a hydrating concrete mixture, the reinforcement does little to prevent cracking because of the weak bond between the concrete and steel.

Recently an analytical system labelled JCMAC3 (Suzuki et al., 1990) has been developed to not only deal with thermal strain but also other initial strains within concrete at an early age (i.e. heat of hydration, autogeneous shrinkage and drying shrinkage). The model computes drying shrinkage from the moisture transport analysis. A non-linear finite element analysis is the global solver for the strain model. JCMAC3 is expected to be a useful tool for simulation of cracking in concrete, since it reproduces recent research results and has an essential practical aspect.

2.6 CONCLUSION

Portland Cement Concrete has been shown to be a complex material that has been researched extensively with numerous methods for understanding this material being proposed. The sophistication of proposed simulation models indicates this complexity of the material and requires introduction of model assumptions which in turn limit the functionality of the models.

The introduction of adiabatic calorimeter results into a temperature prediction model therefore has benefits of reducing theoretical and scientific inaccuracies. The adiabatic test combined with the Arrhenius maturity function expresses the rate of heat evolution from the hydrating cement in a rational and normalised form required by numerical models.

The finite element temperature prediction model presented in this research report will therefore, with the use of adiabatic calorimeter results, be restricted to a macrostructural model controlled by factors related to the chemistry of cement. The model is unique in that the powerful finite element theory has been combined with a reliable method of describing the amount of heat liberated from a specific concrete mixture.

The model however, is limited with respect to cracking potential as it is not able to predict the consumption of water by the hydration reaction which results in a change of concrete stiffness. The problem regarding stiffness gain with respect to time can however be dealt with statically at a specific point in time with empirical Young's modulus values. This relationship will not be dealt with in this research report, but the finite element model allows for a future possibility of this development.

3 DEVELOPMENT OF THE FINITE ELEMENT MODEL FOR HEAT FLOW IN CONCRETE

3.1 INTRODUCTION

The finite element method is a numerical analysis technique for obtaining approximate solutions to a wide variety of engineering problems. In engineering situations today it is necessary to obtain approximate numerical solutions to problems due to the unavailability of exact closed-form solutions. Such examples are: finding the load carrying capacity of a steel plate that has several stiffeners and odd-shaped holes, evaluating the concentration of pollutants during non-uniform atmospheric conditions, determining the rate of fluid flow through a passage of arbitrary space configuration. With relative ease, governing equations and boundary conditions for these situations can be resolved, although no simple analytical solution may be found. The difficulty in these three examples lies in the fact that either the geometry or some other feature of the problem may be irregular or poorly quantified. Analytical solutions to problems of this type seldom exist even though these situations are commonly encountered by engineers. One possible way to overcome this problem is to make simplifying assumptions, which ignore inherent complexities. Occasionally this procedure yields adequate results. However, more often than not, serious inaccuracies and incorrect solutions are encountered. A more viable alternative is to retain the complexities of the problem and find an approximate numerical solution.

Many numerical analysis techniques have been developed, but the most easily accessible technique is the finite difference scheme. This analysis tool gives a point-wise solution to the governing equations of the problem. The continuum in which the solution is sought is replaced by a finite set of points and an attempt is made to find approximate values for these points by the use of difference equations. This model improves in accuracy as more points are used within the continuum. The finite difference technique can solve fairly difficult problems and,

with the ease of implementation, the method is commonly employed. However, when irregular geometries or unusual boundary conditions are encountered the finite difference technique becomes difficult to implement and the results are unreliable in their accuracy.

In recognition of the shortcomings of the finite difference method other numerical analysis techniques have been developed. The most powerful technique emerging in recent years is the finite element method. Unlike the finite difference method, which envisions the continuum as a region of grid points, the finite element method envisions the continuum as a region built up of small interconnected sub-regions or elements. Therefore the finite element method gives a piece-wise approximation to the governing equations. The basic principle of the finite element method is that the continuum can be analytically modelled or approximated by replacing it with an assemblage of discrete elements. These elements can be assembled in a variety of ways and can thus be used to represent extremely complex geometric shapes.

As an example, Figure 3.1 shows a part for an industrial machine. This complex geometric shape is modelled using both the finite difference and finite element methods. The uniform finite difference mesh approximates the irregular shape well except at the boundaries where the series of steps give a rough estimate. On the other hand the finite element method (using a combination of triangular and quadrilateral elements) improves the approximation of the continuum and requires fewer nodes. This example illustrates the fact that the finite element method is better suited as a numerical analysis technique for problems with irregular geometries.

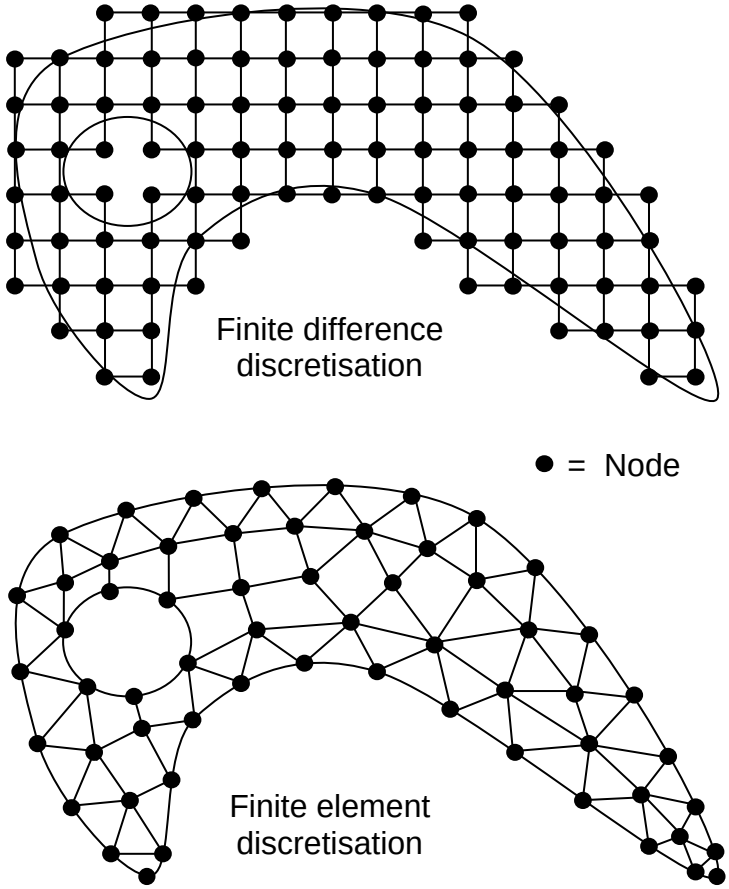


Figure 3.1 Finite difference and finite element discretisations of a machine part

3.2 PRINCIPLE OF THE FINITE ELEMENT METHOD

In a continuum problem (body of matter or simply a region of space in which a particular phenomenon is occurring) of any dimension, the field variable (temperature in the case of a heat transfer problem) possesses infinitely many values because it is a function of each generic point in the body or solution region. Therefore the problem is one with an infinite number of unknowns. The discretisation procedure reduces the problem to a finite number of unknowns by dividing the continuum into elements and by expressing the unknown field variable in terms of assumed approximating functions within each element. The approximating functions or interpolation functions are defined at specified points called nodes. Nodes are generally positioned on the element boundaries where adjacent elements are connected. For the finite element representation of a

problem the nodal values of the field variable are the unknowns. Once the unknowns are found the interpolation functions define the distribution of the field variable throughout the assemblage of elements. Thus, the nature of the solution and the degree of accuracy depend not only on the size and number of elements used but also on the interpolation functions selected. Interpolation functions cannot be chosen arbitrarily due to certain compatibility conditions that need to be satisfied.

The finite element method differs from other numerical analysis techniques due to the ability of formulating solutions for individual elements before putting them together to represent the entire problem. In essence, a complex problem reduces to considering a series of greatly simplified problems.

The formulation of the properties of individual elements can be done in a variety of ways. There are three different approaches, namely:

- Direct approach

  This approach is used for relatively simple problems and is traceable to the direct stiffness method of structural analysis.
- Variational approach

  This approach relies on the calculus of variations and involves extremizing the functional. For problems in solid mechanics the functional turns out to be the potential energy.
- Weighted residuals approach

  This approach is the most versatile approach to deriving element properties. It starts with the governing equations of the problem and proceeds without relying on a variational statement. This method is widely used to derive element properties for non-structural applications such as heat transfer and fluid mechanics.

Whichever of the above approaches is used to formulate the properties of an individual element, the process of finding the solution to a continuum problem is a step-by-step procedure shown below.

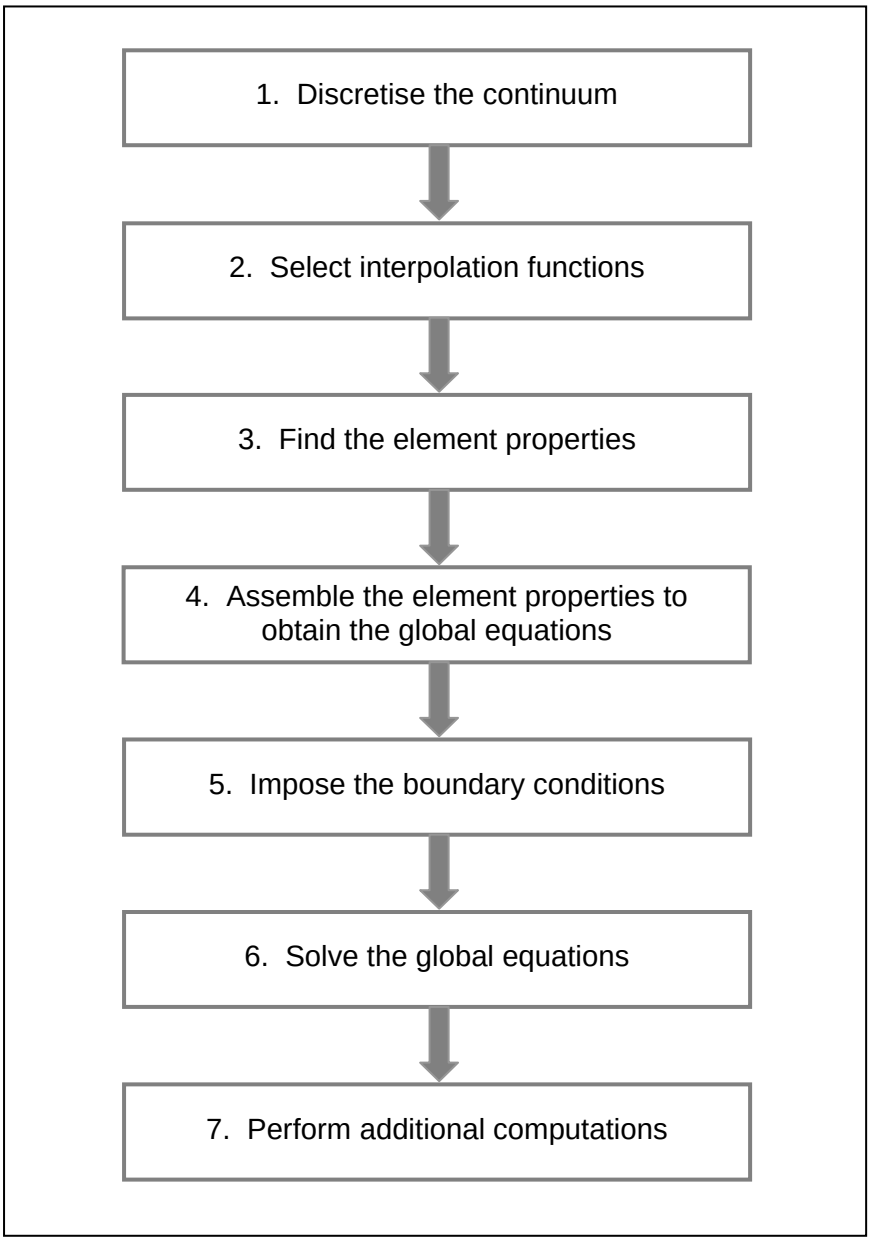


Figure 3.2 Stepwise finite element process

1 Discretising the continuum:

This initial step involves subdividing the continuum into a suitable number of small bodies, called finite elements, as shown in Figure 3.1. Different element shapes can be used in order to obtain a better approximate of the continuum. For many cases an irregular boundary can be approximated by a number of straight lines. Alternatively, it may be necessary to use mathematical functions of sufficient order to approximate the boundary. If the boundary is a parabolic shape, a second order quadratic function can be used to approximate that boundary. This method is used in the concept of isoparametric elements, and will be discussed in further detail in the application of the finite element method.

2 Selecting interpolation functions:

In this step, we allocate nodes to each element and then choose an interpolation function to represent the variation of the field variable over the element. Polynomials are often selected as interpolation functions because of the ease with which they can be integrated or differentiated. The degree of the polynomial depends on the number of nodes assigned to the element (i.e. the greater the number of nodes the higher the degree of the polynomial). The nodal points of the element provide strategic points for writing mathematical functions to describe the shape of the distribution of the field variable over the domain of the element. If we denote u as the field variable, the polynomial interpolation function can be expressed as:

$$u = N_1u_1 + N_2u_2 + N_3u_3 + ... + N_mu_m \qquad (3.1)$$

In the above expression u_1, u_2, u_3,..., u_m are the values of the field variable at the nodal points 1, 2, 3,..., m and N_1, N_2, N_3,..., N_m are the interpolation functions. It should be noted here that after all the steps of the finite element method are accomplished, the values of the field variable at all of the nodes is established. However, to initiate action towards obtaining the solution, an interpolation or shape function needs to be assumed in advance. Figure 3.3 shows that the final solution is a combination of solutions in each element patched together at the

common boundaries. This is further illustrated by sketching a cross-section along section P-P. From this figure, it can be observed that the computed solution is not necessarily the same as the exact solution shown by the solid curve. The statement that the finite element discretisation yields approximate solutions can be visualized from this schematic representation.

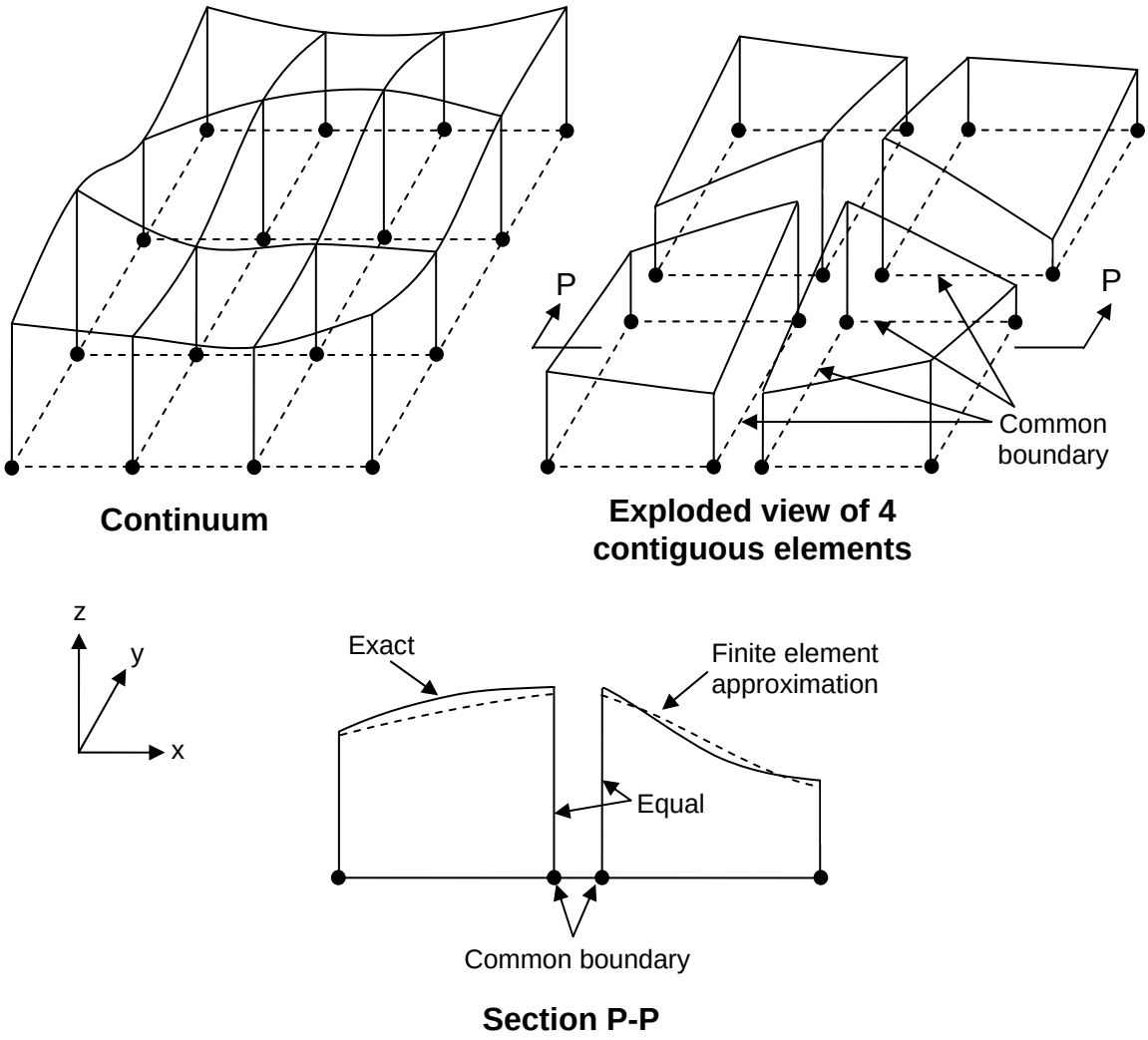


Figure 3.3 Approximate solution as a patchwork of solutions over the elements

3 Finding the element properties:

Once the type of elements and their interpolation functions have been selected, the matrix equations expressing the properties of the individual elements may be formulated. The formulation of the properties of individual elements can be done in any one of the three ways mentioned previously (viz. the direct approach, the variational approach, or the weighted residuals approach).

4 Assembling the element properties to obtain the global equations:

The final aim is to obtain equations for the overall system modelled by the network of elements. Fundamentally, the matrix equations expressing the behaviour of the elements are combined to form the matrix equations expressing the behaviour of the entire system of elements. In other words we are assembling the element equations into system equations, otherwise known as global equations, for the entire system. The global matrix equations have the same form as the individual element equations except that they contain many more terms since they include all the nodes. The principle of this assemblage process stems from the fact that at a common node, where elements are interconnected, the value of the field variable is identical for each element sharing the node.

5 Imposing the boundary conditions:

This step accounts for the boundary conditions which are imposed on the global matrix equations. The boundary conditions need to be applied before a solution to the problem can be found. Thus, known nodal values of the dependent variable are imposed on the boundary.

6 Solving the global equations:

The assemblage process of step four results in a set of simultaneous equations that can be solved to obtain the unknown nodal values of the field variable for the specific problem. If the problem describes steady behaviour, it will be necessary to solve linear or non-linear algebraic equations. However, if the problem is unsteady, the nodal values are a function of time and it is therefore necessary to solve linear or non-linear ordinary differential equations.

7 Performing additional computations if required:

The solution of the system equations can very often be used to calculate various other parameters. For example, in structural engineering application involving a load-deformation problem, the primary solution is deformation. These deformations can be used to calculate secondary solutions such as the element stresses and strains.

3.3 Application of the Finite Element Method

The application of the finite element method can be divided into three main categories, each dependent on the nature of the problem needing to be solved.

The first category deals with problems known as *equilibrium problems* or time-independent problems. Most applications of the finite element method fall into this category. The second category is concerned with problems known as *eigenvalue problems* of solid or fluid mechanics. These types of problems are steady state. The third category deals with problems which are time dependent and are known as *propagation problems* of continuum mechanics. This is a category of problems composed of the previous two problems with an added time dimension, thus it is defined as a transient problem. Recognizing that heat transfer finite element analysis is a transient problem the third category of problems is applicable to the problem of temperature development in concrete.

The application of the finite element method to a transient heat transfer problem has been simplified into the following stepwise method:

In general all the subsequent equations are taken or developed from Huebner et al. (1995) and Lewis et al. (1996). It is important to note that the parentheses [ ] refer to a matrix that contains more than one row and one column, { } refer to a column vector and ⌊ ⌋ refer to a row vector.

Step 1:

A subject of paramount importance in finite element analysis is the selection of the particular finite elements to be used to discretise the continuum. Fitting a curved boundary with straight sided elements generally leads to a satisfactory representation of the boundary, but improved fitting would be possible if curved sided elements could be formulated. With the use of curve sided elements it would be possible to use a smaller number of larger elements and still obtain a close boundary representation. Thus for large complex continuums, the amount of computer time required to obtain a solution to the problem can be considerably reduced.

The principle of curved sided elements centres on mapping or transforming simple geometric shapes (particularly quadrilaterals) in some local coordinate system into distorted shapes in the global Cartesian coordinate system and then evaluating the element equations for the curved sided elements that result. These types of elements are known as isoparametric elements.

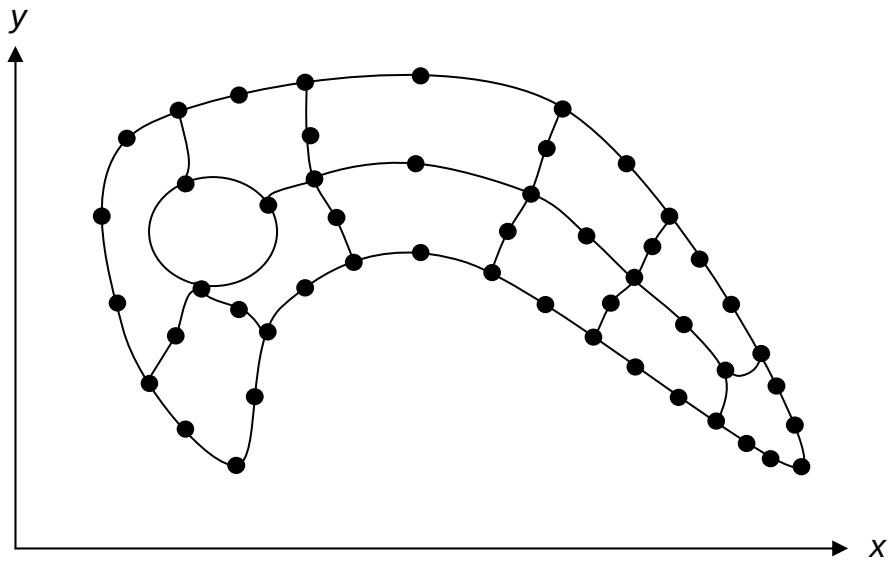


Figure 3.4 Curve sided elements – Isoparametric elements

Three nodes must be associated with each side of the quadrilateral element if it is required to represent a continuum in *x*-*y* Cartesian coordinates by a network of curved sided quadrilateral elements, and furthermore it is required that the field variable *T* (temperature in this case) has a quadratic variation within each element. The continuum and the desired finite element model might appear as shown in Figure 3.4. To construct one typical element of this assemblage it is required to relate this one element to a simpler "parent" element in the ξ-η local coordinate system shown in Figure 3.5.

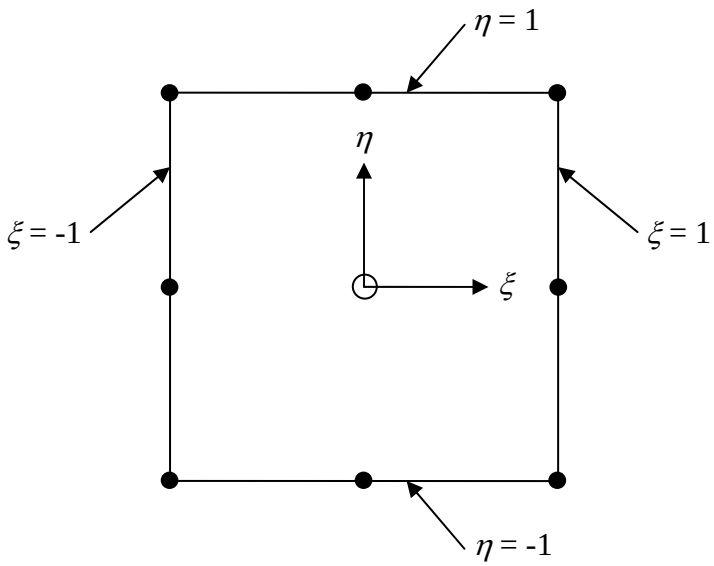


Figure 3.5 Isoparametric "parent" element

This element is part of the serendipity family of rectangular elements. The nodes in the ξ-η plane may be mapped into corresponding nodes in the *x*-*y* plane as follows:

$$x = \sum_{i=1}^{8} N_i(\xi,\eta)x_i \tag{3.2}$$

$$y = \sum_{i=1}^{8} N_i(\xi,\eta)y_i \tag{3.3}$$

In the above equations the mapping functions, N_i, are the element interpolation functions and these will be defined in Step 2. These interpolation functions need to be quadratic since the curved boundaries of the element in the *x*-*y* plane need three points for their unique specification, and the interpolation functions should take on the proper values of unity and zero when evaluated at the nodes in the ξ-η plane.

When writing equations such as equations 3.2 and 3.3, we assume that the transformation between the local ξ-η coordinates and the global *x*-*y* coordinates is unique (i.e. we assume that each point in one coordinate system relates to a unique corresponding point in the other coordinate system). It is also important that continuity between curved elements is upheld when the elements are assembled (i.e. the slope between adjacent elements is constant).

These mapping equations result in curved sided quadrilateral elements of the type shown in Figures 3.6 and 3.7. These figures were produced using equations 3.2 and 3.3 and the element interpolation functions.

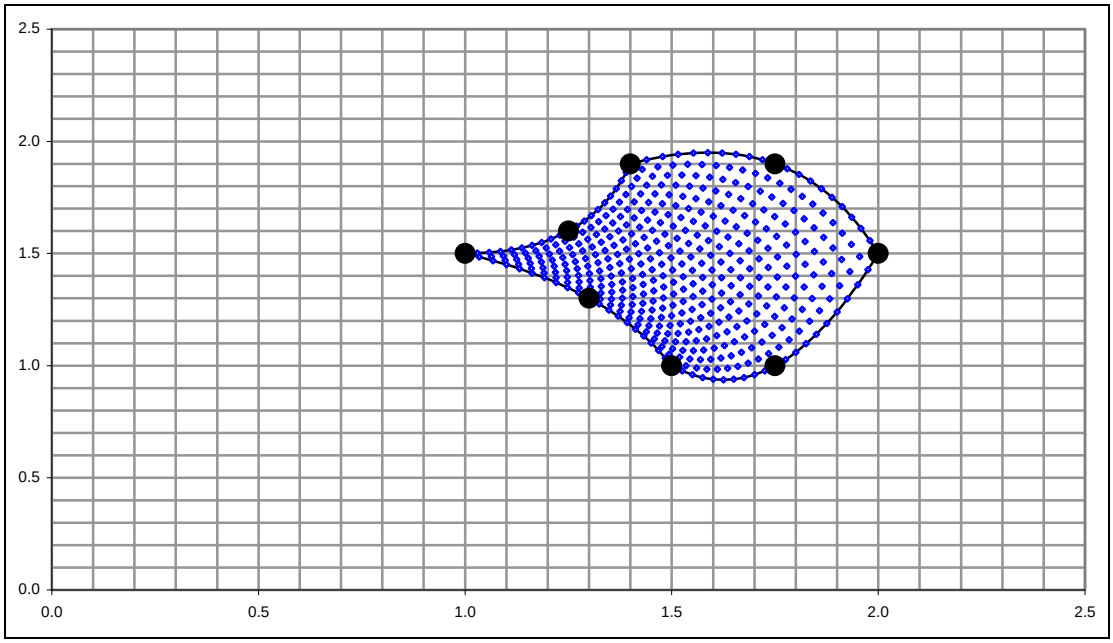


Figure 3.6 Curved sided eight noded quadrilateral isoparametric elements

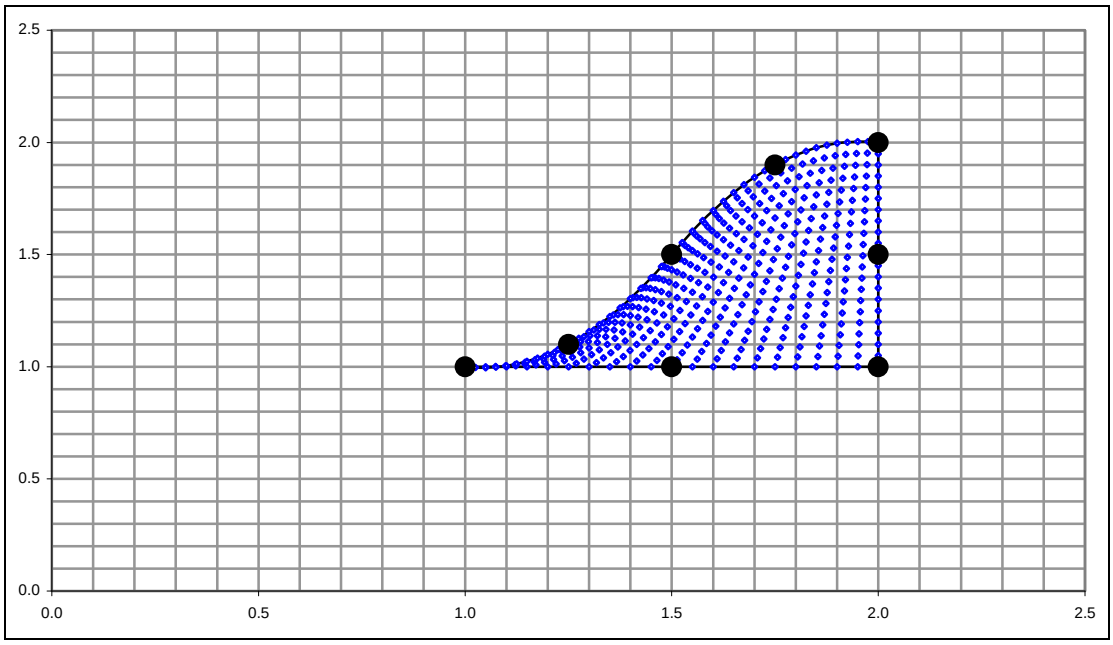


Figure 3.7 Curved sided eight noded quadrilateral isoparametric elements

Step 2:

In the finite element procedure, once the element mesh for the solution continuum has been decided, the behaviour of the unknown field variable over each element is approximated by continuous functions expressed in terms of the nodal values of the field variable. The functions defined over each finite element are called interpolation functions or shape functions.

Interpolation functions cannot be chosen arbitrarily as certain continuity requirements must be met to ensure that the convergence criteria are satisfied. Due to the fact that the field variable is continuous at the element faces, it can be said that the current problem has C^0 continuity.

The type of elements to be used in the proposed heat transfer analysis is eight noded quadrilateral isoparametric elements with the nodes located as shown in Figure 3.8.

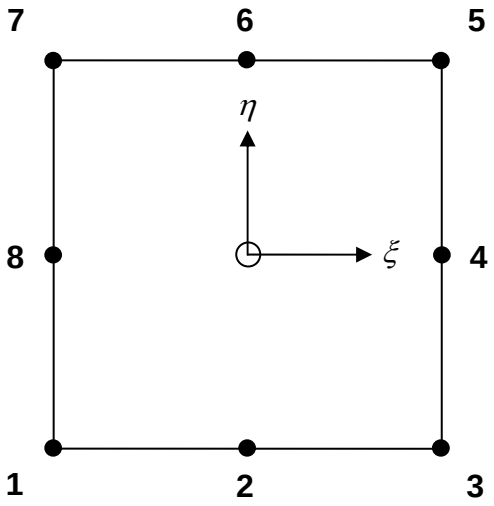


Figure 3.8 Eight noded quadrilateral isoparametric element

These elements possess interpolation functions associated with any arbitrary node *j* and can be determined directly from the requirement:

$$N_j = \begin{cases} 1 \;\; at \;\; Node \;\; j \\ 0 \;\; at \;\; Node \; k; \; k \neq j \end{cases}$$

For this type of element the corresponding nodal interpolation functions are:

$$\left.\begin{aligned} N_1 &= (1-\xi)(1-\eta)(-1-\xi-\eta)/4 & \qquad N_2 &= (1-\xi^2)(1-\eta)/2 \\ N_3 &= (1+\xi)(1-\eta)(-1+\xi-\eta)/4 & \qquad N_4 &= (1+\xi)(1-\eta^2)/2 \\ N_5 &= (1+\xi)(1+\eta)(-1+\xi+\eta)/4 & \qquad N_6 &= (1-\xi^2)(1+\eta)/2 \\ N_7 &= (1-\xi)(1+\eta)(-1-\xi+\eta)/4 & \qquad N_8 &= (1-\xi)(1-\eta^2)/2 \end{aligned}\right\} \quad (3.4)$$

The form of the interpolation functions is illustrated in Figure 3.9, specifically for node 1. The form of the interpolation functions, N_1 to N_8, produce similar plots.

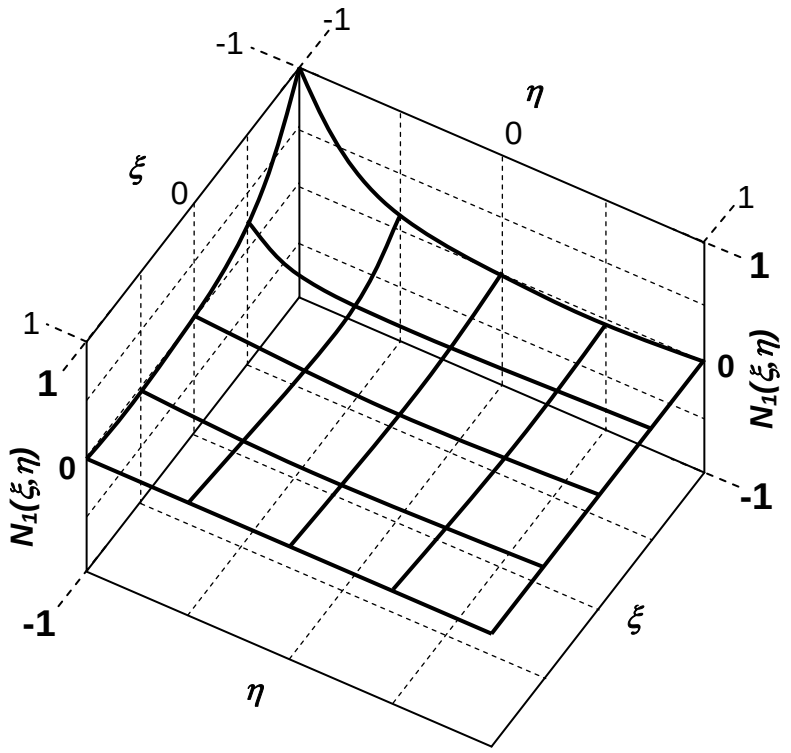


Figure 3.9 Form of interpolation function N_1

Figure 3.10 shows the temperature distribution due to the interpolation functions within an individual element when the nodal values of temperature T_1, T_2,…, T_8 are prescribed.

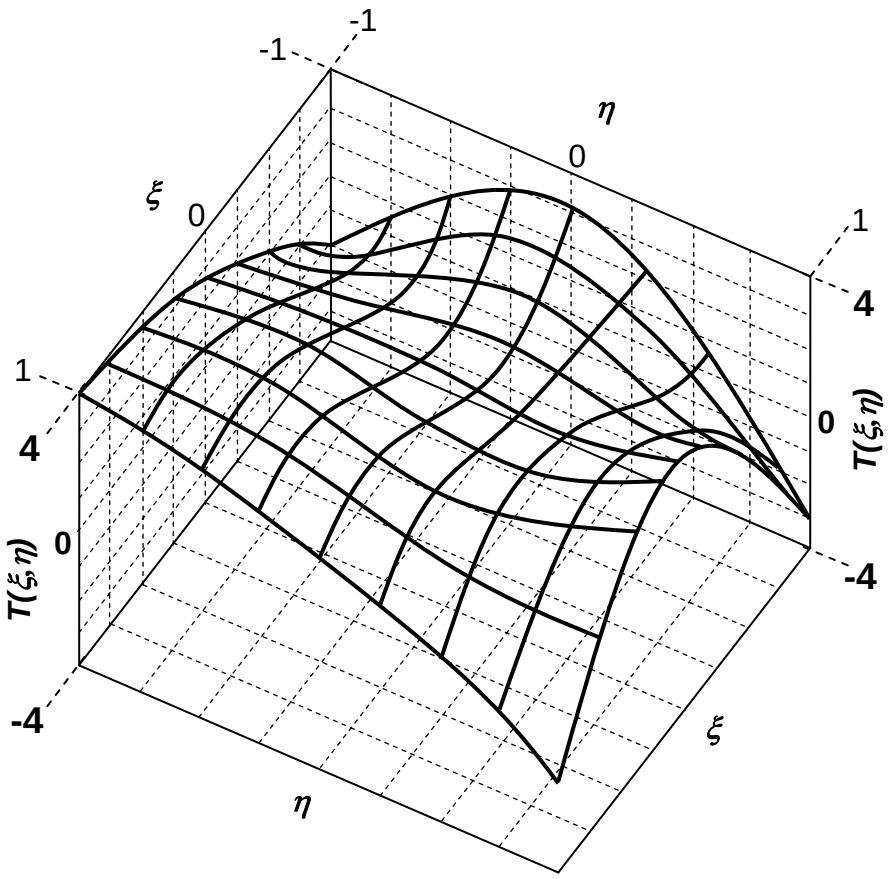


Figure 3.10 Temperature distribution when nodal values of temperature are prescribed

If the edge of an eight noded quadrilateral isoparametric element coincides with the boundary of a region with surface heat transfer, an additional conductance matrix and heat load vector are required. The heat load vectors are the essential boundary conditions to which the surface of the continuum is subjected. The curved edge of a typical eight noded quadrilateral isoparametric element resembles the quadratic rod element and therefore we can use the corresponding matrices for the rod element, provided the proper surface is employed.

The three noded quadratic rod element has interpolation functions of the form:

$$\left.\begin{aligned} N_1(\xi) &= -\xi(1-\xi)/2 \\ N_2(\xi) &= \left(1-\xi^2\right) \\ N_3(\xi) &= \xi(1+\xi)/2 \end{aligned}\right\} \tag{3.5}$$

These interpolation functions are shown in Figure 3.11, as well as the form of the field variable T when the nodal values T_1, T_2, and T_3 are prescribed.

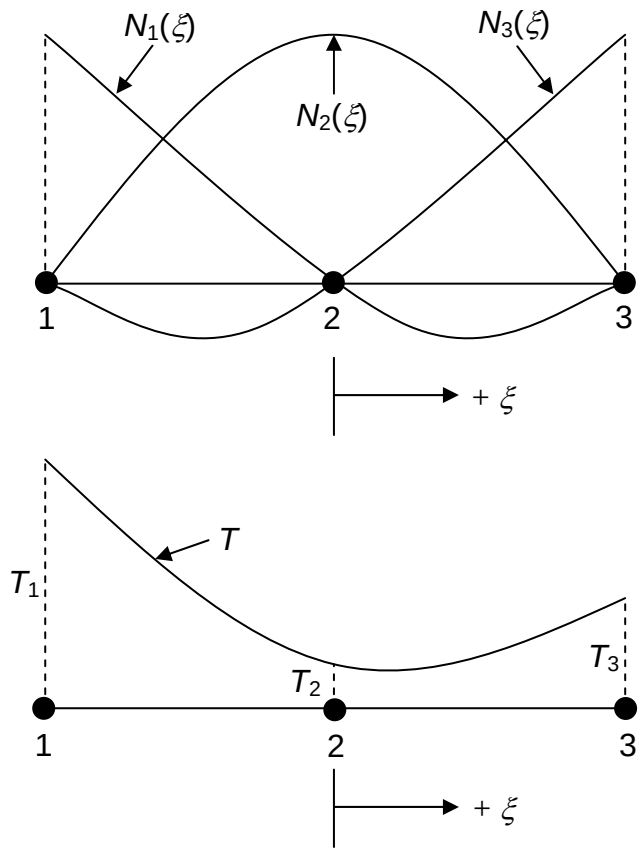


Figure 3.11 Form of the interpolation functions and temperature distribution when the nodal values of the temperature are prescribed

The 3 noded quadratic rod element is also part of the serendipity family of elements. Therefore the nodes in the ξ plane are mapped onto corresponding nodes in the *x*-*y* plane. This is accomplished by again mapping the region $-1 \leq \xi \leq 1$ into the region of interest by the isoparametric mapping condition:

$$x = \sum_{i=1}^{3} N_i(\xi)\, x_i \tag{3.6}$$

$$y = \sum_{i=1}^{3} N_i(\xi)\, y_i \tag{3.7}$$

These elements are assembled with the eight noded quadrilateral isoparametric elements as shown in step 4.

Step 3:

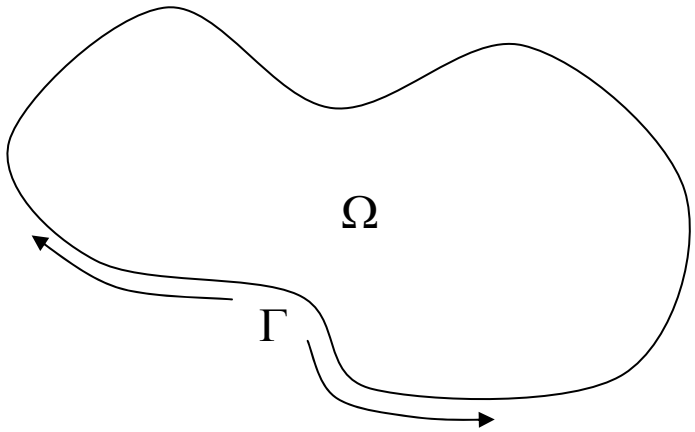


Figure 3.12 General continuum (Ω) and boundary (Γ)

Considering a transient heat transfer problem in a two-dimensional solid, Ω, bounded by a surface, Γ, (Figure 3.12). The problem is governed by the energy equation:

$$-\left(\frac{\partial q_x}{\partial x}+\frac{\partial q_y}{\partial y}\right)+Q'=\rho\ c\frac{\partial T}{\partial t} \tag{3.8}$$

Where Q' is the internal heat generation rate per unit volume, ρ is the density, c is the specific heat, t is the time and q_x and q_y are components of the heat flow rate vector per unit area in Cartesian coordinates. For an anisotropic continuum Fourier's law is:

$$\left.\begin{aligned} q_x &= -\left(k_{11}\frac{\partial T}{\partial x}+k_{12}\frac{\partial T}{\partial y}\right) \\ q_y &= -\left(k_{21}\frac{\partial T}{\partial x}+k_{22}\frac{\partial T}{\partial y}\right) \end{aligned}\right\} \tag{3.9}$$

With k being the thermal conductivity, the heat conduction equation is derived for non-linear material properties. Note: Thermal conductivity k can be considered constant for concrete, while assuming it to be an isotropic medium.

The heat conduction equation is solved subject to initial and boundary conditions. The initial condition stipulates the temperature within the continuum when time is zero (i.e. the placing temperature of the concrete).

$$T(x,y,0)=T_0(x,y) \tag{3.10}$$

The boundary conditions may incorporate specified surface temperature, specified surface heat flow, convective heat exchange and radiation.

The latter two boundary conditions were selected as being the most appropriate conditions for the problem; thus the boundary conditions are:

$$q_x n_x + q_y n_y = h\left(T_s - T_e\right) \qquad \text{Convection on all surfaces} \qquad (3.11)$$

$$q_x n_x + q_y n_y = \varepsilon\sigma\left(T_s^4 - T_e^4\right) \qquad \text{Radiation on all surfaces except bottom surface} \qquad (3.12)$$

Where n_x and n_y are the direction cosines of the outward normal to the surface, h is the convective heat transfer coefficient, T_s is the unknown surface temperature, T_e is the known atmospheric temperature, σ is the Stefan-Boltzmann constant and ε is the surface emissivity.

The convection heat transfer coefficient, h, is dependent on whether the concrete is still contained by the formwork. The formwork generally is constructed of timber, and the heat transfer coefficient is taken as approximately 5 W/m^{2}K compared with the approximate value of 30 W/m^{2}K for concrete without formwork (Holman, 1986). The heat transfer coefficient is a measure of how much heat can be transmitted between the surface and the environment.

The surface temperature, T_s, is the temperature at the nodes on each of the surfaces and is unknown. The atmospheric temperature, T_e, is the temperature of the surrounding environment and is approximated by the following sinusoidal function (Ballim, 2004a):

$$T_e = -\sin\left(\frac{2\pi\left(t_d + t_w\right)}{24}\right)\left(\frac{T_{\max} - T_{\min}}{2}\right) + \left(\frac{T_{\max} + T_{\min}}{2}\right) \qquad (3.13)$$

Where t_d is the clock time of day at which the approximation is being made (0 to 24 hours), t_w is the time at which the minimum overnight temperature occurs, T_{max} is the maximum temperature for the day under consideration and T_{min} is the minimum temperature for the day under consideration.

An important benefit of this equation is that it can approximate the ambient temperature at any time using only the daily maximum and minimum temperatures which can be obtained from the local meteorological office. This ensures that at the design stage of a construction project, the ambient temperatures can be predicted on an hourly basis as required for the numerical heat transfer analysis. However, the model does have a disadvantage. It has been simplified and does not incorporate factors such as cloud cover, wind or direct sunlight. It should be noted that direct sunlight could be included in the radiation boundary condition as solar radiation if required. The presence of wind could also be included within the convection heat transfer coefficient. The greater the wind speed the higher the coefficient value and vice versa. Thus, with the above arguments the ambient temperature model can be said to be fairly accurate and has been shown to give acceptable results (Ballim, 2004a). Figure 3.13 shows the sinusoidal function for prescribed maximum and minimum temperatures (Ballim, 2004a).

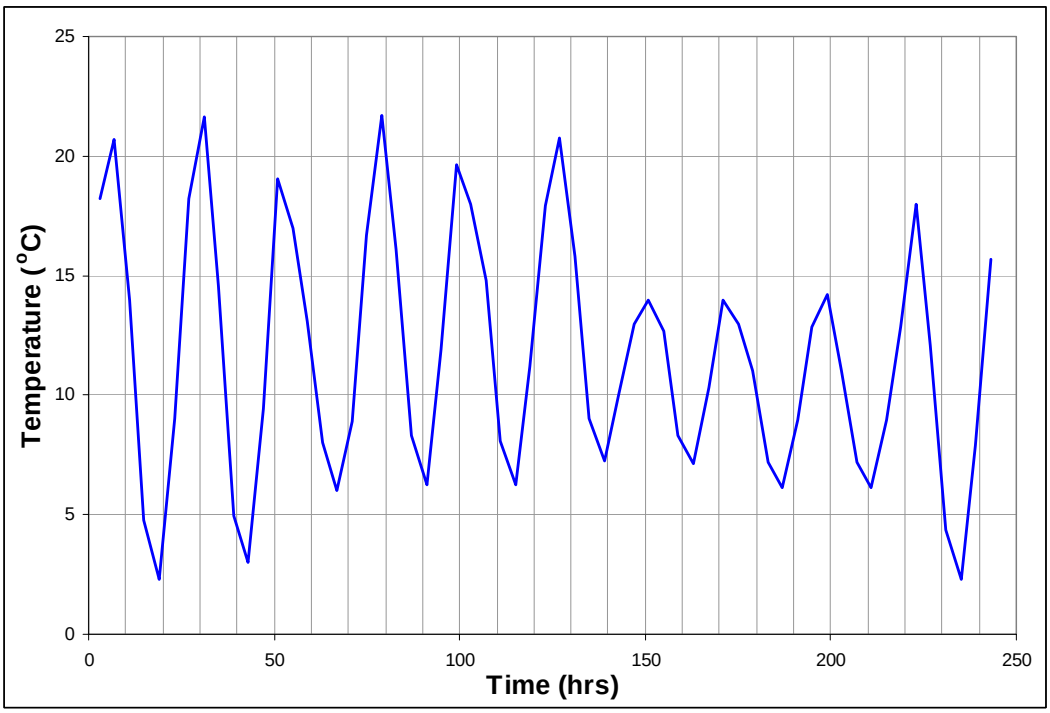


Figure 3.13 Modelled atmospheric temperatures using equation 3.13

The radiation boundary condition shows that energy will be emitted at a rate proportional to the fourth power of the absolute temperature of the body. The emissivity of the grey concrete surface is equal to the ratio of emission from a grey surface to that from a perfect radiator at the same temperature and can be taken as 0.9 for most cases (Isgor, 2004). The Stefan-Boltzmann constant, σ, is the proportionality constant and has a value of 5.669×10^{-8} W/m^{2}K^4.

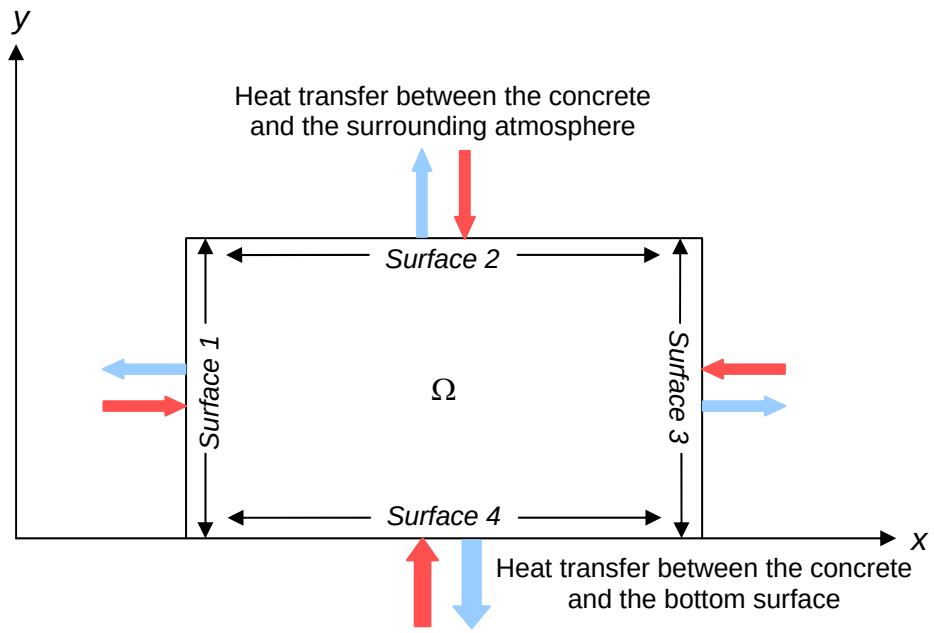


Figure 3.14 Heat transfer between Surfaces 1 to 4 and the environment

For a typical cross-section as shown in Figure 3.14, the boundaries are specified as Surface 1 to Surface 4. Each of the surfaces undergoes convection into the surface or convection out of the surface from the surrounding environment. It is assumed that the bottom surface (Surface 4) will lose or gain heat from the casting surface (generally rock or concrete) through convection. Conceptually this is incorrect but this assumption will not have a large effect on the results (refer to Chapter 4 of this report). A proposed solution to this problem would be to model the casting surface as a separate mesh. This would then lead to more accurate results although the computation time required to obtain a solution would increase significantly. It is also assumed that radiation into or out of the cross-section occurs on surfaces 1, 2 and 3 only.

The material properties, specifically thermal conductivity, and boundary conditions are non-linear as they change with incremental time. It can therefore be stated that the problem is inherently non-linear. Although the derivation of the finite element equation is for non-linear problems it can be assumed at a later stage that these material properties are actually linear. This is done for simplicity since consideration of non-linear material properties would have high demand on the computation time required to obtain a solution to a particular problem. The internal heat generation rate per unit volume (*Q*') is inherently non-linear as a result of the exothermic hydration reaction. This non-linearity is resolved through the use of an iterative solution as it cannot be regarded as linear.

Following the derivation of the initial and boundary conditions, the formulation of the finite element equation for a single element is required. Knowing that the continuum has been divided into *M* number of elements each containing eight nodes, the temperature and temperature gradients within each element can be expressed as:

$$T^{(e)}(x,y,t) = \sum_{i=1}^{8} N_i(x,y) T_i(t) \tag{3.14}$$

$$\partial T^{(e)} / \partial x\,(x,y,t) = \sum_{i=1}^{8} \partial N_i / \partial x(x,y) T_i(t) \tag{3.15}$$

$$\partial T^{(e)} / \partial y\,(x,y,t) = \sum_{i=1}^{8} \partial N_i / \partial y(x,y) T_i(t) \tag{3.16}$$

or in matrix notation

$$T^{(e)}(x,y,t) = \lfloor N(x,y) \rfloor \{T(t)\} \tag{3.17}$$

$$\begin{Bmatrix} \partial T/\partial x(x,y,t) \\ \partial T/\partial y(x,y,t) \end{Bmatrix} = [B(x,y,t)]\{T(t)\} \tag{3.18}$$

Where the superscript *e* denotes a single element, [*N*] is the temperature interpolation matrix, [*B*] is the temperature gradient interpolation matrix, $T_i(t)$ is the value of the temperature at each node and $\{T(t)\}$ is a single column vector with eight rows of nodal temperatures.

$$\lfloor N(x,y) \rfloor = \lfloor N_1 \quad N_2 \quad N_3 \quad N_4 \quad N_5 \quad N_6 \quad N_7 \quad N_8 \rfloor \tag{3.19}$$

$$[B(x,y)] = \begin{bmatrix} \partial N_1/\partial x & \partial N_2/\partial x & \cdots & \partial N_8/\partial x \\ \partial N_1/\partial y & \partial N_2/\partial y & \cdots & \partial N_8/\partial y \end{bmatrix} \tag{3.20}$$

The method of weighted residuals is now employed to formulate the individual element equations. This method is a global technique for obtaining approximate solutions to linear and non-linear partial differential equations. It allows the finite element equations to be derived directly from the governing differential equations. The method employs two steps, the first of which is to assume the general functional behaviour of the dependent field variable (temperature), in such a way as to satisfy the governing differential equation and boundary conditions. This assumption will result in an error when it is substituted into the governing differential equation. This error is known as a residual. In order to cancel out this residual, the error is averaged over the entire continuum ensuring an insignificant error.

If it is required to find an approximate functional representation for the field variable, *T*, governed by the differential equation (equation 3.21) in a continuum, Ω, bounded by the surface Γ, step one can be illustrated.

$$G(T) - b = 0 \tag{3.21}$$

Where *b* is a known function of the independent variable (i.e. *b* specifies the boundary conditions) and *G* is the differential operator. The first step in the

method of weighted residuals requires that the unknown exact solution T is approximated by T' and is shown in equation 3.22.

$$T \approx T' = \sum_{i=1}^{m} N_i C_i \tag{3.22}$$

Where N_i are the assumed interpolation functions, C_i are the unknown functions of the independent variable (function of time for transient problems) and m is the number of unknown C_i values. Substituting T' (equation 3.22) into the governing differential equation (equation 3.21) results in the following:

$$G(T') - b = R \neq 0 \tag{3.23}$$

Where R is the error or residual that is obtained with the approximation of T with T'. As mentioned previously the method of weighted residuals attempts to determine the m unknown C_i values such that the residual over the entire continuum is small. A weighted average of the residual is then created which fades away over the entire continuum. Therefore, m linearly independent weighting functions, W_i, are chosen and the following weighting is then performed:

$$\int_T [G(T') - b] W_i \, d\Omega = \int_\Omega R W_i \, d\Omega = 0 \qquad i = 1, 2, 3, \ldots, m \tag{3.24}$$

Therefore $R \approx 0$. With the specification of the weighting functions, the above equation represents a set of m algebraic or ordinary differential equations.

The second step involves solving the equation resulting from the first step and to find a general functional form for the governing differential equation which will then become the approximate solution to the problem. This is done by solving equation 3.24 for C_i and hence obtaining an approximate solution for the field

variable T from equation 3.22. It has been shown that as m increases towards infinity the approximate temperature T' approaches the actual temperature T. The selection of the weighting functions now becomes important due to the wide variety of functions that can be used. The Galerkin method is the most common method used to derive the weighting functions which are required to obtain the finite element equations. In particular the Bubnov-Galerkin method is employed and thus the chosen weighting functions are the same as the interpolation functions given in Step 2. Therefore $W_i = N_i$ for $i = 1, 2, 3,\ldots, 8$. There are many other choices for the weighting functions such as the Pertov-Galerkin method (e.g. least-squares method and collocation method), however these are far more complex than the Bubnov-Galerkin method.

The Bubnov-Galerkin method is now implemented on the governing differential equation (equation 3.21).

$$\int_{\Omega}\left[G(T') - b\right]N_i\, d\Omega = 0 \tag{3.25}$$

This equation deals with the entire continuum, although it also holds for an arbitrary subdomain or element within the continuum. Therefore the above equation will be formulated for a single element as Step 3 of the application of the finite element method is only concerned with single element equations. The weighting functions N_i are now the interpolation functions for the type of element concerned (eight noded quadrilateral isoparametric element) and the C_i unknown values are the nodal values of the field variable (i.e. temperature). Equation 3.25 now becomes:

$$\int_{\Omega^{(e)}}\left[G\left(T'^{(e)}\right) - b^{(e)}\right]N_i^{(e)}\, d\Omega = 0 \qquad i = 1, 2, 3,..., r \tag{3.26}$$

Where the superscript e indicates that equation 3.26 is for a single element and r is the number of nodes assigned to the element.

With equation 3.27 the temperature at each node within the element can then be solved.

$$T^{(e)} = \lfloor N^{(e)} \rfloor \{T\}^{(e)} \tag{3.27}$$

Now that the method of weighted residuals has been briefly introduced, the differential equation used to illustrate the method (equation 3.21) can now be replaced in equation 3.26 with the governing heat transfer differential equation (equation 3.8).

$$\int_{\Omega^{(e)}} \left(\partial q_x / \partial x + \partial q_y / \partial y - Q + \rho c \, \partial T / \partial t\right) N_i d\Omega = 0 \tag{3.28}$$

For simplicity the superscript *e* has been omitted. The above equation is now expanded in order to obtain a solution.

$$\underbrace{\int_{\Omega^{(e)}} \left(\partial q_x / \partial x + \partial q_y / \partial y\right) N_i d\Omega}_{\text{A}} - \underbrace{\int_{\Omega^{(e)}} Q N_i d\Omega}_{\text{B}} + \underbrace{\int_{\Omega^{(e)}} \rho c \, \partial T / \partial t \, N_i d\Omega}_{\text{C}} = 0 \tag{3.29}$$

It is now required to simplify equation 3.29 with part A of the equation simplified by applying integration of parts. When integration of parts is used to find the element equations it offers a convenient way to introduce the natural boundary conditions that must be satisfied on some portion of the boundary. Although the boundary terms containing the natural boundary conditions appear in the equations for each element, in the assembly of the element equations only the boundary elements give contributions that do not disappear.

Integration by parts for a two-dimensional continuum Ω with boundary Γ is:

$$\int_{\Omega} u(\nabla \cdot v)d\Omega = \int_{\Gamma} u(v \cdot \hat{n})d\Gamma - \int_{\Omega} v \cdot \nabla u d\Omega$$
$$Let \quad u = N_i \quad and \quad v = q_x \hat{i} + q_y \hat{j} \tag{3.30}$$

Applying integration of parts on part A of equation 3.29 yields:

$$\begin{aligned} A &= \int_{\Omega^{(e)}} (\partial q_x / \partial x + \partial q_y / \partial y) N_i d\Omega \\ &= \int_{\Gamma^{(e)}} N_i (q_x n_x + q_y n_y) d\Gamma - \int_{\Omega^{(e)}} (q_x + q_y)(\partial N_i / \partial x + \partial N_i / \partial y) d\Omega \\ &= \int_{\Gamma^{(e)}} (\bar{q} \cdot \hat{n}) d\Gamma - \int_{\Omega^{(e)}} [\partial N_i / \partial x \quad \partial N_i / \partial y] \begin{Bmatrix} q_x \\ q_y \end{Bmatrix} d\Omega \end{aligned} \tag{3.31}$$

Now equation 3.29 is re-written in a simplified form:

$$\int_{\Omega^{(e)}} \rho c \, \partial T / \partial t \, N_i d\Omega -$$
$$\int_{\Omega^{(e)}} [\partial N_i / \partial x \quad \partial N_i / \partial y] \begin{Bmatrix} q_x \\ q_y \end{Bmatrix} d\Omega = \int_{\Omega^{(e)}} Q N_i d\Omega \underbrace{- \int_{\Gamma^{(e)}} (\bar{q} \cdot \hat{n}) N_i d\Gamma}_{\text{Boundary Conditions}} \tag{3.32}$$

The surface integral is now expressed in terms of the relevant surface and the boundary conditions are introduced. As previously mentioned, each surface experiences convection (i.e. Surface 1 to Surface 4) and surfaces 1 to 3 experience radiation.

Therefore equation 3.32 becomes:

$$\int_{\Omega^{(e)}} \rho c\, \partial T/\partial t\, N_i d\Omega - \int_{\Omega^{(e)}} \begin{bmatrix} \partial N_i/\partial x & \partial N_i/\partial y \end{bmatrix} \begin{Bmatrix} q_x \\ q_y \end{Bmatrix} d\Omega =$$

$$\begin{aligned} &\int_{\Omega^{(e)}} Q N_i d\Omega - \int_{S1} h(T_s - T_e) N_i d\Gamma - \int_{S2} h(T_s - T_e) N_i d\Gamma - \\ &\int_{S3} h(T_s - T_e) N_i d\Gamma - \int_{S4} h(T_s - T_e) N_i d\Gamma - \int_{S1} \varepsilon\sigma\left(T_s^4 - T_e^4\right) N_i d\Gamma - \\ &\int_{S2} \varepsilon\sigma\left(T_s^4 - T_e^4\right) N_i d\Gamma - \int_{S3} \varepsilon\sigma\left(T_s^4 - T_e^4\right) N_i d\Gamma \end{aligned} \qquad (3.33)$$

Where S1 to S4 refer to Surfaces 1 to 4 respectively.

From equation 3.9, knowing that the material properties are assumed to be non-linear:

$$\begin{Bmatrix} q_x \\ q_y \end{Bmatrix} = -\begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{bmatrix} \begin{Bmatrix} \partial T/\partial x \\ \partial T/\partial y \end{Bmatrix}$$
$$\{q\} = -[k]\{\dot{T}\} \qquad (3.34)$$

From equations 3.15 and 3.16:

$$\{\dot{T}\} = [B]\{T(t)\} \qquad (3.35)$$

Therefore:

$$\begin{Bmatrix} q_x \\ q_y \end{Bmatrix} = -\begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{bmatrix} \begin{bmatrix} \partial N_1/\partial x & \partial N_2/\partial x & \dots & \partial N_8/\partial x \\ \partial N_1/\partial y & \partial N_2/\partial y & \dots & \partial N_8/\partial y \end{bmatrix} \begin{Bmatrix} T_1 \\ T_2 \\ \vdots \\ T_8 \end{Bmatrix} \qquad (3.36)$$

Now we will assume that the material properties are linear, therefore:

$$[k] = \begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{bmatrix} \qquad (3.37)$$

Where *k* is the constant thermal conductivity for the concrete.

$$\int_{\Omega^{(e)}} \rho c N_i \left(\partial \lfloor N \rfloor \{T(t)\}/\partial t\right) d\Omega +$$

$$\int_{\Omega^{(e)}} \begin{bmatrix} \partial N_1/\partial x & \partial N_1/\partial y \\ \partial N_2/\partial x & \partial N_2/\partial y \\ \vdots & \vdots \\ \partial N_8/\partial x & \partial N_8/\partial y \end{bmatrix} [k] \begin{bmatrix} \partial N_1/\partial x & \partial N_2/\partial x & \dots & \partial N_8/\partial x \\ \partial N_1/\partial y & \partial N_2/\partial y & \dots & \partial N_8/\partial y \end{bmatrix} \{T\} d\Omega =$$

$$\begin{aligned} &\int_{\Omega^{(e)}} Q N_i d\Omega - \int_{S1} h T_s N_i d\Gamma + \int_{S1} h T_e N_i d\Gamma - \int_{S2} h T_s N_i d\Gamma + \int_{S2} h T_e N_i d\Gamma - \\ &\int_{S3} h T_s N_i d\Gamma + \int_{S3} h T_e N_i d\Gamma - \int_{S4} h T_s N_i d\Gamma + \int_{S4} h T_e N_i d\Gamma - \\ &\int_{S1} \varepsilon\sigma T_s^4 N_i d\Gamma + \int_{S1} \varepsilon\sigma T_e^4 N_i d\Gamma - \int_{S2} \varepsilon\sigma T_s^4 N_i d\Gamma + \int_{S2} \varepsilon\sigma T_e^4 N_i d\Gamma - \\ &\int_{S3} \varepsilon\sigma T_s^4 N_i d\Gamma + \int_{S3} \varepsilon\sigma T_e^4 N_i d\Gamma \end{aligned} \tag{3.38}$$

Where:

$$N_i = \begin{Bmatrix} N_1 \\ N_2 \\ \vdots \\ N_8 \end{Bmatrix} \tag{3.39}$$

$$T = \lfloor N \rfloor \{T(t)\} \tag{3.40}$$

Simplifying:

$$\begin{aligned} &\int_{\Omega^{(e)}} \rho c \{N\} \lfloor N \rfloor \{\partial T/\partial t\} d\Omega + \int_{\Omega^{(e)}} [B]^T [k][B] \{T\} d\Omega = \int_{\Omega^{(e)}} Q \{N\} d\Omega - \\ &\int_{S1} h \{N\} \lfloor N \rfloor \{T\} d\Gamma + \int_{S1} h T_e \{N\} d\Gamma - \int_{S2} h \{N\} \lfloor N \rfloor \{T\} d\Gamma + \int_{S2} h T_e \{N\} d\Gamma - \\ &\int_{S3} h \{N\} \lfloor N \rfloor \{T\} d\Gamma + \int_{S3} h T_e \{N\} d\Gamma - \int_{S4} h \{N\} \lfloor N \rfloor \{T\} d\Gamma + \int_{S4} h T_e \{N\} d\Gamma - \\ &\int_{S1} \varepsilon\sigma \{N\} (\lfloor N \rfloor \{T\})^4 d\Gamma + \int_{S1} \varepsilon\sigma T_e^4 \{N\} d\Gamma - \int_{S2} \varepsilon\sigma \{N\} (\lfloor N \rfloor \{T\})^4 d\Gamma + \\ &\int_{S2} \varepsilon\sigma T_e^4 \{N\} d\Gamma - \int_{S3} \varepsilon\sigma \{N\} (\lfloor N \rfloor \{T\})^4 d\Gamma + \int_{S3} \varepsilon\sigma T_e^4 \{N\} d\Gamma \end{aligned} \tag{3.41}$$

Equation 3.41 reduces to:

$$\begin{aligned}[C]\{\partial T/\partial t\}+[k_c]\{T\} \;=\; & \{R_Q\}-[k_h]_{S1}\{T\}-[k_h]_{S2}\{T\}-[k_h]_{S3}\{T\}- \\ & [k_h]_{S4}\{T\}+\{R_h\}_{S1}+\{R_h\}_{S2}+\{R_h\}_{S3}+ \\ & \{R_h\}_{S4}-\{R_\sigma\}_{S1}-\{R_\sigma\}_{S2}-\{R_\sigma\}_{S3}+ \\ & \{R_r\}_{S1}+\{R_r\}_{S2}+\{R_r\}_{S3}\end{aligned} \tag{3.42}$$

Where:

$$[C]=\int_{\Omega^{(e)}}\rho c\{N\}\lfloor N\rfloor d\Omega$$

$$[k_c]=\int_{\Omega^{(e)}}[B]^T[k][B]d\Omega$$

$$\{R_Q\}=\int_{\Omega^{(e)}}Q\{N\}d\Omega$$

$$[k_h]_{S1}=\int_{S1}h\{N\}\lfloor N\rfloor d\Gamma \qquad [k_h]_{S2}=\int_{S2}h\{N\}\lfloor N\rfloor d\Gamma$$

$$[k_h]_{S3}=\int_{S3}h\{N\}\lfloor N\rfloor d\Gamma \qquad [k_h]_{S4}=\int_{S4}h\{N\}\lfloor N\rfloor d\Gamma$$

$$\{R_h\}_{S1}=\int_{S1}hT_e\{N\}d\Gamma \qquad \{R_h\}_{S2}=\int_{S2}hT_e\{N\}d\Gamma$$

$$\{R_h\}_{S3}=\int_{S3}hT_e\{N\}d\Gamma \qquad \{R_h\}_{S4}=\int_{S4}hT_e\{N\}d\Gamma$$

$$\{R_\sigma\}_{S1}=\int_{S1}\varepsilon\sigma\{N\}(\lfloor N\rfloor\{T\})^4\,d\Gamma \qquad \{R_\sigma\}_{S2}=\int_{S2}\varepsilon\sigma\{N\}(\lfloor N\rfloor\{T\})^4\,d\Gamma$$

$$\{R_\sigma\}_{S3}=\int_{S3}\varepsilon\sigma\{N\}(\lfloor N\rfloor\{T\})^4\,d\Gamma$$

$$\{R_r\}_{S1}=\int_{S1}\varepsilon\sigma T_e^4\{N\}d\Gamma \qquad \{R_r\}_{S2}=\int_{S2}\varepsilon\sigma T_e^4\{N\}d\Gamma$$

$$\{R_r\}_{S3}=\int_{S3}\varepsilon\sigma T_e^4\{N\}d\Gamma$$

Where [C] is the element capacitance matrix and relates to conduction and [k_c] and [k_h] are the element conductance matrices which relate to convection. The convection matrix [k_h] is only formulated for elements on the boundaries where convection takes place. The matrices R_Q, R_h, R_σ and R_r are the matrices due to the internal heat generation, surface convection and surface radiation respectively (the latter two matrices are both for surface radiation).

The formulation of the finite element equation for a single element has now been completed. This equation (equation 3.42) however, is formulated for elements with interpolation functions that are defined in Cartesian coordinates. These elements have relatively low accuracy because of the low order interpolation functions. In step 2 of the application of the finite element method eight noded quadrilateral isoparametric elements were chosen to discretise the continuum. These elements have higher order interpolation functions thus they provide higher accuracy and can better approximate curved boundaries. The disadvantage of utilizing isoparametric elements is that the elements are required to be mapped into the Cartesian coordinate system. This principle requires the matrices in equation 3.42 to be modified.

The element matrices involving volume integrals (i.e. [C], [k_c] and $\{R_Q\}$) are integrated with respect to the volume of an element in Cartesian coordinates.

$$d\Omega = t\ dx\ dy$$

Where t is the thickness of the element.

These integrals now need to be modified such that the integration is with respect to the ξ-η coordinate system. This is done such that numerical integration is possible.

The introduction of the Jacobian matrix is required to carry out this transformation. From equations 3.15 and 3.16 it is required to express $\partial N_i/\partial x$ and $\partial N_i/\partial y$ in terms of ξ and η. Due to the inverse form of equations 3.2 and 3.3 it is possible to utilize the chain rule of differentiation such that:

$$\left.\begin{aligned}\partial N_i/\partial\xi &= (\partial N_i/\partial x)(\partial x/\partial\xi)+(\partial N_i/\partial y)(\partial y/\partial\xi)\\ \partial N_i/\partial\eta &= (\partial N_i/\partial x)(\partial x/\partial\eta)+(\partial N_i/\partial y)(\partial y/\partial\eta)\end{aligned}\right\} \tag{3.43}$$

These equations can now be written in matrix notation:

$$\begin{Bmatrix}\partial N_i/\partial\xi\\ \partial N_i/\partial\eta\end{Bmatrix}=\begin{bmatrix}\partial x/\partial\xi & \partial y/\partial\xi\\ \partial x/\partial\eta & \partial y/\partial\eta\end{bmatrix}\begin{Bmatrix}\partial N_i/\partial x\\ \partial N_i/\partial y\end{Bmatrix} \tag{3.44}$$

The Jacobian [J] is now defined as:

$$[J]=\begin{bmatrix}\partial x/\partial\xi & \partial y/\partial\xi\\ \partial x/\partial\eta & \partial y/\partial\eta\end{bmatrix} \tag{3.45}$$

Equations 3.2 and 3.3 are now rewritten and differentiated accordingly:

$$x=\sum_{i=1}^{8}N_i(\xi,\eta)x_i \tag{3.2}$$

$$y=\sum_{i=1}^{8}N_i(\xi,\eta)y_i \tag{3.3}$$

The Jacobian therefore becomes:

$$[J(\xi,\eta)] = \begin{bmatrix} \sum_{i=1}^{8} (\partial N_i(\xi,\eta)/\partial\xi) x_i & \sum_{i=1}^{8} (\partial N_i(\xi,\eta)/\partial\xi) y_i \\ \sum_{i=1}^{8} (\partial N_i(\xi,\eta)/\partial\eta) x_i & \sum_{i=1}^{8} (\partial N_i(\xi,\eta)/\partial\eta) y_i \end{bmatrix} \tag{3.46}$$

Equation 3.44 can be rearranged in the following form:

$$\begin{Bmatrix} \partial N_i/\partial x \\ \partial N_i/\partial y \end{Bmatrix} = [J]^{-1} \begin{Bmatrix} \partial N_i/\partial\xi \\ \partial N_i/\partial\eta \end{Bmatrix} \tag{3.47}$$

Now this transformation process can be illustrated using equations 3.15 and 3.16:

$$\begin{Bmatrix} \partial T/\partial x \\ \partial T/\partial y \end{Bmatrix} = \begin{bmatrix} \partial N_1/\partial x & \partial N_2/\partial x & \dots & \partial N_8/\partial x \\ \partial N_1/\partial y & \partial N_2/\partial y & \dots & \partial N_8/\partial y \end{bmatrix} \begin{Bmatrix} T_1 \\ T_2 \\ \vdots \\ T_3 \end{Bmatrix} \tag{3.48}$$

The above equation can be transformed into:

$$\begin{Bmatrix} \partial T/\partial x \\ \partial T/\partial y \end{Bmatrix} = [J]^{-1} \begin{bmatrix} \partial N_1/\partial\xi & \partial N_2/\partial\xi & \dots & \partial N_8/\partial\xi \\ \partial N_1/\partial\eta & \partial N_2/\partial\eta & \dots & \partial N_8/\partial\eta \end{bmatrix} \begin{Bmatrix} T_1 \\ T_2 \\ \vdots \\ T_3 \end{Bmatrix} \tag{3.49}$$

The element matrices are integrated with respect to the volume of the element relative to the Cartesian coordinates. This can now also be transformed as follows:

$$d\Omega = t\, dx\, dy = t\, |J(\xi,\eta)|\, d\xi\, d\eta \tag{3.50}$$

The Jacobian matrix is also useful for checking whether the mapping is acceptable (i.e. the elements do not overlap). If $|J| \neq 0$ and the sign of $|J|$ does not change in the continuum, acceptable mapping can be assured ($|J|$ is the determinate of J).

This transformation allows the element integrals to be evaluated by integration over the unit square of the isoparametric parent element. The element matrices involving volume integrals ([C], [k_c] and {R_Q}) can now be transformed into:

Capacitance matrix:

$$[C] = \int_{-1}^{1} \int_{-1}^{1} \rho\, c\, t \{N(\xi,\eta)\} \lfloor N(\xi,\eta) \rfloor |J(\xi,\eta)|\, d\xi\, d\eta \tag{3.51}$$

This double integral can now be evaluated using a numerical integration technique. The commonly used method is Gauss-Legendre quadrature. Using this technique the capacitance matrix becomes:

$$[C] = \sum_{i=1}^{NG} \sum_{j=1}^{NG} W_i W_j \rho\, c\, t \left\{N(\xi_i,\eta_j)\right\} \left\lfloor N(\xi_i,\eta_j) \right\rfloor \left|J(\xi_i,\eta_j)\right| \tag{3.52}$$

Where *NG* is the number of Gauss points in each integration direction and can range between 1 and 8, W_i and W_j are the Gauss weights and ξ_i and η_j are the coordinates of the Gauss points.

This numerical integration technique (Gauss-Legendre quadrature) was however not used in the Matlab programme code, due to the vast number of built-in integration functions in Matlab.

Conductance matrix:

The element conductance matrix is formulated in a similar manner, except the temperature gradient matrix [B] is evaluated in terms of ξ and η. Using equations 3.20 and 3.47 the equation for [B] in terms of ξ and η is derived:

$$[B(\xi,\eta)] = [J(\xi,\eta)]^{-1}\begin{bmatrix} \partial N_1/\partial\xi & \partial N_2/\partial\xi & \dots & \partial N_8/\partial\xi \\ \partial N_1/\partial\eta & \partial N_2/\partial\eta & \dots & \partial N_8/\partial\eta \end{bmatrix} \tag{3.53}$$

The element conductance matrix therefore is:

$$[k_c] = \int_{-1}^{1}\int_{-1}^{1} t\,[B(\xi,\eta)]^T[k][B(\xi,\eta)]\,|J(\xi,\eta)|\,d\xi\,d\eta \tag{3.54}$$

The conductance matrix can also be evaluated by Gauss-Legendre quadrature:

$$[k_c] = \sum_{i=1}^{NG}\sum_{j=1}^{NG} W_i W_j\, t\,[B(\xi_i,\eta_j)]^T[k][B(\xi_i,\eta_j)]\,|J(\xi_i,\eta_j)| \tag{3.55}$$

Heat liberated matrix:

$$\{R_Q\} = \int_{-1}^{1}\int_{-1}^{1} Q\,t\,\{N(\xi,\eta)\}\,|J(\xi,\eta)|\,d\xi\,d\eta \tag{3.56}$$

The heat liberated matrix can also be evaluated by Gauss-Legendre quadrature:

$$\{R_Q\} = \sum_{i=1}^{NG}\sum_{j=1}^{NG} W_i W_j\, t\, Q\,\{N(\xi_i,\eta_j)\}\,|J(\xi_i,\eta_j)| \tag{3.57}$$

The remaining element matrices ($[k_h]_{S1\text{ to }S4}$, $\{R_h\}_{S1\text{ to }S4}$, $\{R_\sigma\}_{S1\text{ to }S4}$ and $\{R_r\}_{S1\text{ to }S4}$) can be evaluated similarly by Gauss-Legendre quadrature in the ξ plane. These element matrices are due to the existence of the three noded quadratic rod elements that are used to incorporate the boundary conditions. Integrals along an element edge (integration with respect to Γ) are evaluated using a local coordinate system *s*. The distance $d\Gamma$ along an element edge can be expressed in terms of *ds*:

$$ds = \sqrt{(dx/d\xi)^2 + (dy/d\xi)^2}\; d\xi = d\Gamma \tag{3.58}$$

Considering a side *s* of an element that connects three nodes, as shown in Figure 3.15, the Pythagoras theorem is used for this derivation.

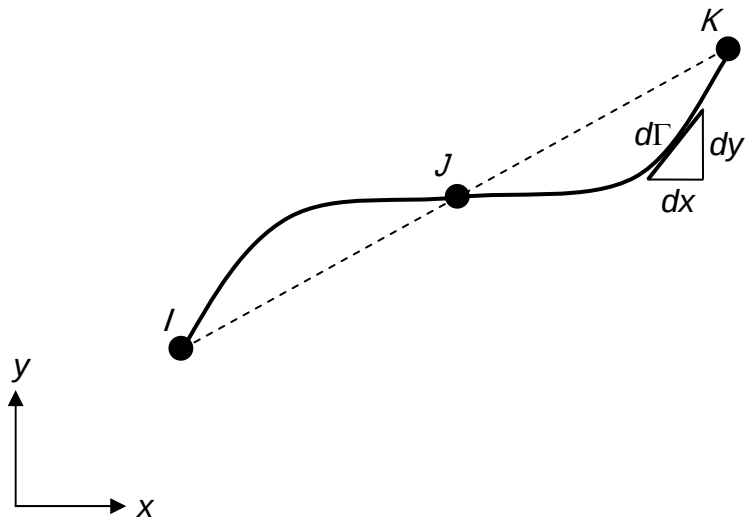


Figure 3.15 Segment *s* of a boundary which forms one side of an eight noded quadrilateral element

This transformation allows the element integrals to be evaluated by integration over the isoparametric parent element. The element matrices involving surface integrals ($[k_h]_{S1\text{ to }S4}$, $\{R_h\}_{S1\text{ to }S4}$, $\{R_\sigma\}_{S1\text{ to }S4}$ and $\{R_r\}_{S1\text{ to }S4}$) can now be transformed into:

Additional conductance matrix:

$$[k_h]_{S1\,to\,S4} = \int_{-1}^{1} h\,\{N(\xi)\}\lfloor N(\xi)\rfloor t\sqrt{(dx/d\xi)^2 + (dy/d\xi)^2}\,d\xi \tag{3.59}$$

Using Gauss-Legendre quadrature:

$$[k_h]_{S1\,to\,S4} = \sum_{i=1}^{NG} W_i\; h\,t\,\{N(\xi_i)\}\lfloor N(\xi_i)\rfloor\sqrt{(dx/d\xi)^2 + (dy/d\xi)^2} \tag{3.60}$$

Additional heat load matrices:

$$\{R_h\}_{S1\,to\,S4} = \int_{-1}^{1} h\,T_e\;\{N(\xi)\}\,t\,\sqrt{(dx/d\xi)^2 + (dy/d\xi)^2}\,d\xi \tag{3.61}$$

$$\{R_\sigma\}_{S1\,to\,S4} = \int_{-1}^{1} \sigma\,\varepsilon\,\{N(\xi)\}\,(\lfloor N(\xi)\rfloor\{T\})^4\;t\,\sqrt{(dx/d\xi)^2 + (dy/d\xi)^2}\,d\xi \tag{3.62}$$

$$\{R_r\}_{S1\,to\,S4} = \int_{-1}^{1} \sigma\,\varepsilon\,T_e^4\;\{N(\xi)\}\,t\,\sqrt{(dx/d\xi)^2 + (dy/d\xi)^2}\,d\xi \tag{3.63}$$

Gauss-Legendre quadrature can also be applied to the above three matrices.

Now that the finite element equation has been defined and modified due to the use of isoparametric elements, the assemblage of the elements is required.

Step 4:

The assembly procedure is in principle a general procedure that applies to all finite element systems. The general procedure can be summarized in the following steps:

1 Create n x n and n x 1 all zero matrices, where n is the total number of nodes required to discretise a continuum. In Figure 3.16, n = 21. Due to temperature being the only unknown (degree of freedom) at each node the number of nodes is equal to the number of unknowns required to be solved throughout the continuum. Therefore the number of unknown temperature values is equal to n.

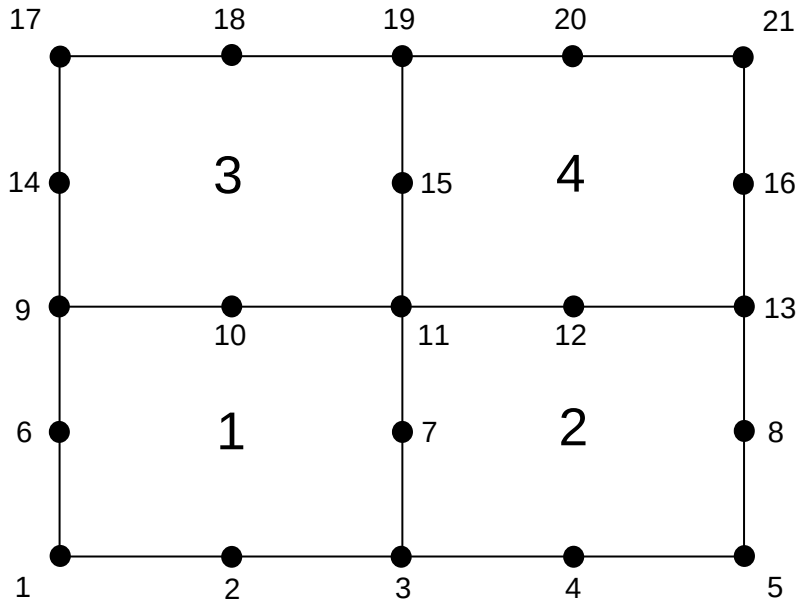


Figure 3.16 Four elements connected producing 21 nodes

2 Transform the element equations from the local to the global numbering system if the two numbering systems are not coincident. In Figure 3.17 the local and global numbering systems are shown for element 4 from Figure 3.16. The process of assemblage is a stepwise process, thus only one element is considered presently.

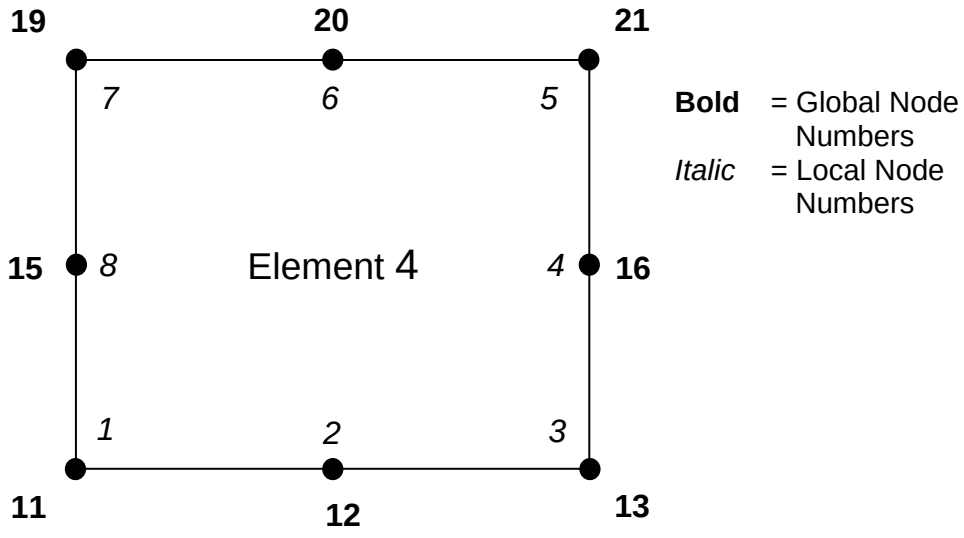


Figure 3.17 Element 4, global and local numbering scheme

3 Using this established correspondence between local and global numbering systems, change from the local to the global indices. Further, it requires that a subscript referencing the position of each and every term in the element matrices be created. The square element matrices will have a double subscript to indicate the row and column in which the term is located. The column matrices require only a single subscript. It must be emphasized that these subscripts are global node numbers not local node numbers.

4 These terms are now inserted into the corresponding $n \times n$ and $n \times 1$ all zero matrices in the locations designated by their indices. If for instance there is already a term in a certain position where another term is required to be placed, the terms are added.

5 It is now required to return to step 2 of the assemblage process and repeat this procedure until all the elements have been considered. The final result will be $n \times n$ global capacitance, conductance and additional conductance matrices and $n \times 1$ global column heat liberated and heat load matrices. This then results in a global finite element equation for the entire

continuum containing *M* number of elements. This equation is then solved to obtain the temperature distribution throughout the continuum (refer to step 6).

This assembling process is based on the law of compatibility or continuity as it requires that the body remain continuous. Due to the fact that temperature is a scalar quantity, it must be assured that continuity will be upheld.

There is however the issue of combining the three noded quadratic rod elements with the eight noded quadrilateral isoparametric elements. Figure 3.18 shows four quadrilateral elements and eight rod elements plus the relative global node numbering of the quadrilateral elements.

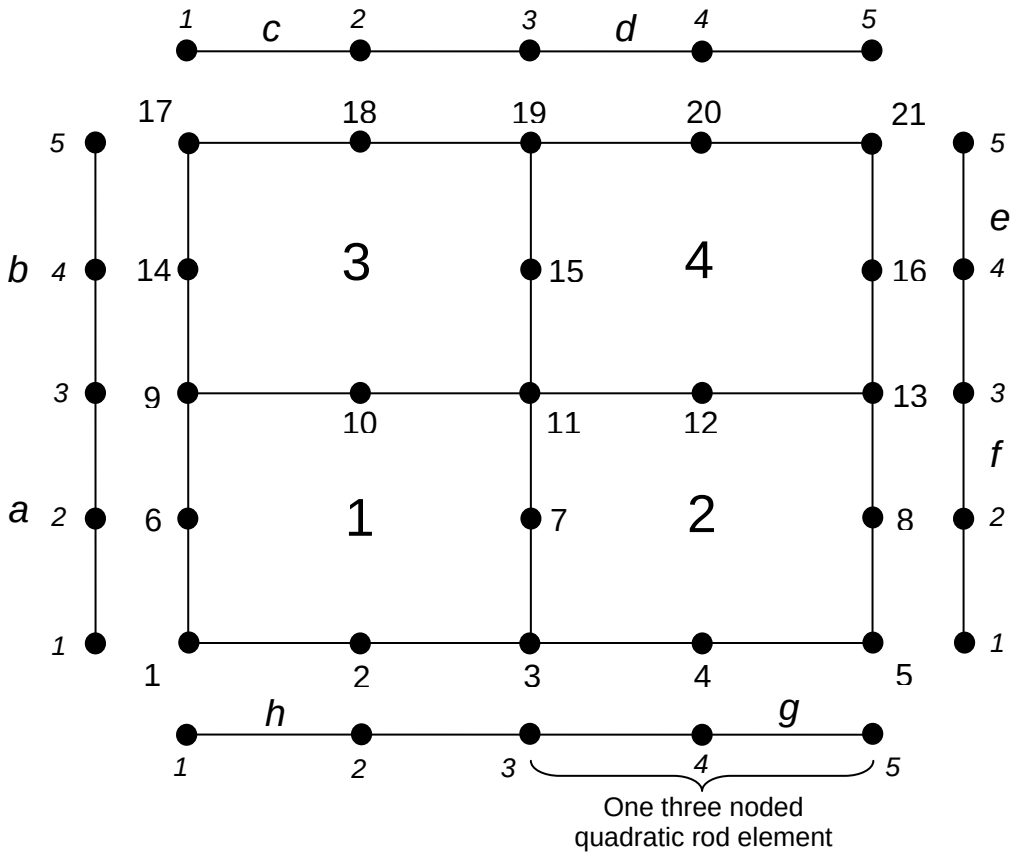


Figure 3.18 Combining three noded quadratic elements with eight noded quadrilateral isoparametric elements

Element *a* in Figure 3.18 shares common nodes with element 1 (i.e. node 2 of element *a* shares a node with node 6 of element 1). This sharing of nodes is fundamental to the assemblage process. The node numbering of the three-noded rod elements is in terms of the local numbering system, and therefore these elements will adopt the global node numbering from the eight noded quadrilaterals in the assemblage process due to node sharing. The heat load vectors will therefore only have entries in the rows relating to the common global node numbers.

The surface integrals can now be combined for the appropriate like terms and equation 3.42 becomes the global finite element heat transfer equation:

$$[C]\{\partial T/\partial t\} + [\,[k_c] + [k_h]\,]\{T\} = \{R_Q\} + \{R_h\} - \{R_\sigma\} + \{R_r\} \qquad (3.64)$$

Step 5:

The general stepwise procedure of finding a solution to a continuum problem with the use of the finite element method includes a separate step for the inclusion of the boundary conditions. This step however, has already been taken into account in Step 3. The process of solving for an individual element equation involves the method of weighted residuals and related integration by parts. As mentioned previously, the integration by parts incorporates the boundary conditions automatically and thus they have already been considered.

Step 6:

The transient global finite element heat transfer equation (equation 3.64) is solved in this step. This is an ordinary differential equation that is required to be solved iteratively. This numerical integration procedure relies on recursion formulas which permit the solution to be “marched out” in time, starting from an initial temperature distribution. The algorithm to be used is the implicit one-parameter

"θ" scheme where -1 ≤ θ ≤ 1. This scheme computes temperatures at time t_{n+1} from a set of coupled algebraic equations.

$$[C]\{\partial T/\partial t\} + [\,[k_c] + [k_h]\,]\{T\} = \{R_Q\} + \{R_h\} - \{R_\sigma\} + \{R_r\} \tag{3.64}$$

$$[C]\{\dot{T}\}_\theta + [\,[k_c] + [k_h]\,]\{T\}_\theta = \underbrace{\{R_Q\} + \{R_h\} - \{R_\sigma\} + \{R_r\}}_{H} \tag{3.65}$$

The following approximations can be formulated:

$$\{\dot{T}\}_\theta = (\{T\}_{n+1} - \{T\}_n)/\Delta t \sim \partial T/\partial t \tag{3.66}$$

$$\{T\}_\theta = (1-\theta)\{T\}_n + \theta\{T\}_{n+1} \tag{3.67}$$

Where Δt is the time step through which the solution will be iterated, $\{T\}_{n+1}$ is the temperature at the next time step and $\{T\}_n$ is the temperature at the current time step.

Substituting equations 3.66 and 3.67 into equation 3.65 yields:

$$[C](\{T\}_{n+1} - \{T\}_n)/\Delta t + [\,[k_c] + [k_h]\,](1-\theta)\{T\}_n + \theta\{T\}_{n+1} = H \tag{3.68}$$

Equation 3.68 then simplifies to provide a time marching scheme to be used to obtain a solution to the transient global finite element heat transfer equation.

$$[\,(1/\Delta t)[C] + [\,[k_c] + [k_h]\,]\theta\,]\{T\}_{n+1} = [\,(1/\Delta t)[C] - [\,[k_c] + [k_h]\,](1-\theta)\,]\{T\}_n + \{R_Q\} + \{R_h\} - \{R_\sigma\} + \{R_r\} \tag{3.69}$$

The parameter θ was chosen to be ½, which represents the Crank-Nicolson algorithm. This algorithm is unconditionally stable, but a time step that is too large may introduce oscillations that could cause the solution to diverge from the exact solution. Equation 3.69 is programmed in Matlab such that a solution to the temperature after each time step (Δt) throughout the continuum is established.

Step 7:

No additional computations are required for the finite element heat transfer analysis, although a proposed additional computation would be to calculate the stresses and strains that are generated within a continuum due to the differential temperatures within the medium.

3.4 CONCLUSION

This chapter considered the fundamental steps required to obtain a transient solution to a heat transfer problem. A general formulation was presented based on the method of weighted residuals. This theory was used to create a programme written in Matlab to predict the heat liberated and distributed over a two-dimensional continuum. Chapter 4 presents the results obtained from the numerical model and compares this data to experimental results. Appendix A gives the Matlab code for the fundamental functions.

4 MODEL OUTPUT AND DISCUSSION

4.1 INTRODUCTION

This chapter deals with the results produced by the finite element numerical model. Comparisons between the predicted and measured results obtained from two different experiments are then drawn. The two available measured temperature profiles that were utilized for a comparison/verification exercise are:

1 Temperature measurement exercise that was conducted during the construction of the Katse Dam in Lesotho (Ballim, 2004a).
2 Temperature verification exercise that was conducted for the previously mentioned finite difference numerical model: An instrumented block of concrete was cast in the University of the Witwatersrand's laboratory from which temperature profiles were obtained (Ballim, 2004b).

Furthermore, this chapter presents a detailed comparison between the predicted two dimensional temperature profiles using finite element and finite difference analysis techniques. Comments and proposed solutions to errors within the modelling techniques are also discussed.

This section will commence with an explanation of the input variables required to obtain a solution from the finite element numerical model - accurate input variables are the fundamental basis for the prediction of valid solutions. The results of the finite element numerical model, including a comparison with the measured and finite difference model results, are then presented. A sensitivity analysis, demonstrating the effect of various element configurations on the predicted results, will bring the model output portion to a close.

4.2 FINITE ELEMENT NUMERICAL MODEL

4.2.1 Input Data

Microsoft Excel spreadsheets were used to generate the input data essential for the finite element numerical model. Various code written in Microsoft Visual Basic, were implemented to create input files that specify the geometry of the finite element discretisation. The Microsoft Excel file "Co-ordinates.xls" (refer to attached compact disk) generates the discretised continuum with the following input data:

Table 4.1 Input required for the mesh generator

x Dimension =	2 m	Number of elements in the *x* direction =	20	Element width in the *x* direction (m) =	0.1	Number of nodes in the *x* direction =	41
y Dimension =	1 m	Number of elements in the *y* direction =	10	Element width in the *y* direction (m) =	0.1	Number of nodes in the *y* direction =	21
Total number of elements =			200	Aspect ratio =	1	Total number of nodes =	661

The cells highlighted in green (i.e. the *x* and *y* dimensions of the cross-section and the number of elements in the *x* and *y* directions) are the only input values required by the user for this discretisation procedure. This procedure produces a finite element mesh as shown as an example in Figure 4.1.

The spreadsheet also creates the global node numbers and corresponding Cartesian co-ordinates of each node. However it must be emphasized, that this particular mesh generator can only discretise a continuum with regular 8 noded rectangular elements of equal size.

The cell highlighted in blue in Table 4.1 represents an individual element's aspect ratio. This is a relationship of the element width (m or mm) to its height (m or mm) (i.e. the proportion of the maximum dimension to the minimum dimension).

As an approximate guideline, elements with an aspect ratio exceeding three should be used with caution and those exceeding ten are viewed as an inappropriate approximation of a continuum (i.e. the elements become excessively elongated and thus the finite element approximation in the extended direction is a rough estimate) (Kumar, 1996). Therefore, in order to obtain comparatively reliable results an aspect ratio of one is maintained (i.e. the elements are square). A sensitivity analysis found in Section 4.5, was performed with a variation of the element's aspect ratio to illustrate the above phenomenon.

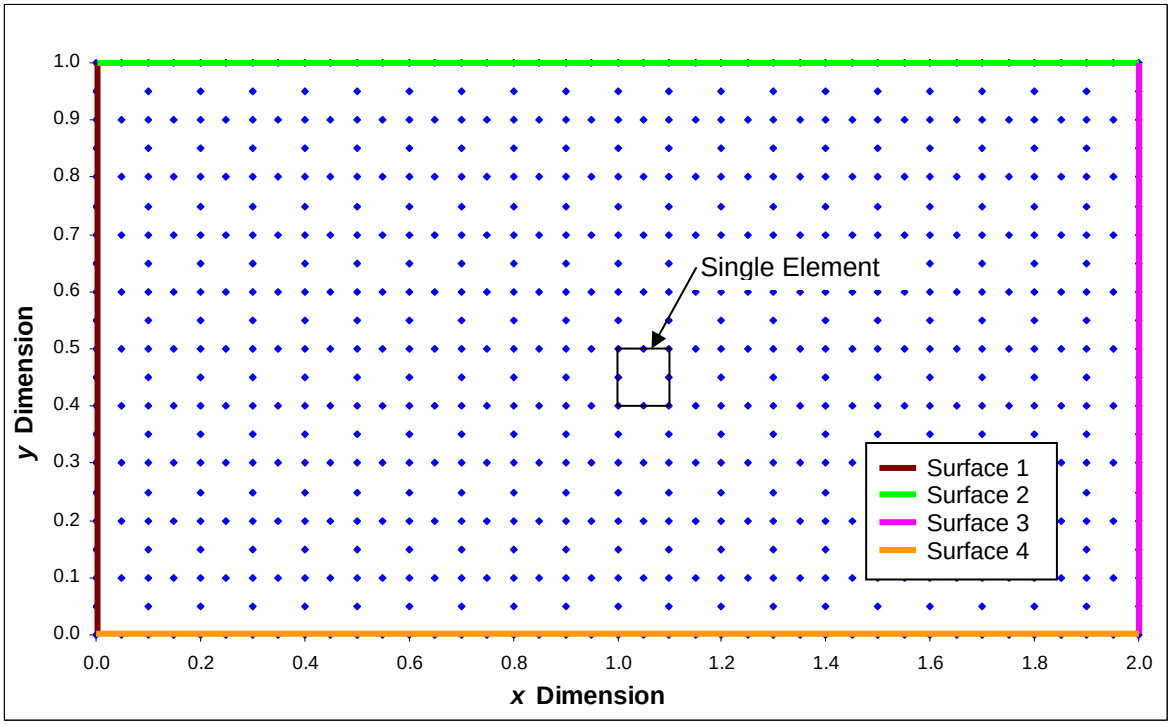


Figure 4.1 **An example of the Finite element discretisation over a specified cross-section (only element nodes are shown) showing all four peripheral surfaces**

This Microsoft Excel file (Co-ordinates.xls) also creates arrays of global node numbers and their relative *x* and *y* co-ordinates for the respective surfaces (i.e. Surface 1 to Surface 4 as shown in Figure 4.1).

A supplementary Microsoft Excel file titled "Elements.xls" (refer to attached compact disk) generates further input files with the same input data as the

previous file. The function of this spreadsheet is to specify how the elements are linked together, and to present the data from the previous spreadsheet in a format compatible with the finite element numerical model.

The spreadsheets within both of the Microsoft Excel files or parts thereof are then saved as data files, such that they can be imported into Matlab.

The general geometric input data produced by the Microsoft Excel files are:

- ElementCoord: The *x* and *y* co-ordinates for each node of every 8 noded quadrilateral element.
- NodesXElements: Global node numbers for each 8 noded quadrilateral element.
- NodesXElementsS1: Global node numbers for each 3 noded quadratic rod element on Surface 1.
- NodesXElementsS2: Global node numbers for each 3 noded quadratic rod element on Surface 2.
- NodesXElementsS3: Global node numbers for each 3 noded quadratic rod element on Surface 3.
- NodesXElementsS4: Global node numbers for each 3 noded quadratic rod element on Surface 4.
- S1: The *x* and *y* co-ordinates for each node of every 3 noded quadratic rod element on Surface 1.
- S2: The *x* and *y* co-ordinates for each node of every 3 noded quadratic rod element on Surface 2.
- S3: The *x* and *y* co-ordinates for each node of every 3 noded quadratic rod element on Surface 3.
- S4: The *x* and *y* co-ordinates for each node of every 3 noded quadratic rod element on Surface 4.

These arrays are also required to assemble the individual elements into a global system for the finite element numerical model assemblage process. Each array is formulated such that the element matrices are assembled relative to the discretisation of the continuum.

Furthermore, additional input arrays are required to define the ambient temperature and the amount of heat liberated within the concrete:

- AmbientTemp: Specifies daily maximum and minimum ambient temperatures (°C) for the use of the sinusoidal temperature variation function programmed in Matlab
- Maturity: Tabulated values of the heat rate curve as calculated using the experimental data obtained from the adiabatic calorimeter test. The values tabulated are: Time (t_{20} hours) with respect to the Maturity Heat Rate (W/kg)

Additional data such as; material properties, initial conditions, specific boundary conditions and time increment and duration are required as input for the finite element numerical model. These input variables must be entered at the commencement of the finite element programme:

- Total number of elements
- Total number of nodes
- Total number of elements in the y-direction
- Total number of elements in the x-direction
- Initial concrete temperature (°C)
- Time of day when concrete is cast (hrs)
- Thermal conductivity of concrete (W/m.K)
- Concrete density (kg/m^3)
- Concrete specific heat (J/kg.K)
- Formwork removal time (hrs)
- Convective heat transfer coefficient for the exposed concrete surface ($W/m^2.K$)

- Convective heat transfer coefficient for the surfaces covered with formwork ($W/m^2.K$)
- Thermal conductivity of founding rock (W/m.K)
- Stefan Boltzman constant ($W/m^2.K^4$)
- Emissivity of the grey concrete surface
- Time at which the minimum overnight temperature occurs (hrs)
- Binder content (kg/m^3)
- Apparent activation energy (kJ/mol)
- Universal gas constant (kJ/mol.K)
- Time increment (hrs)
- Time period (hrs)

4.2.2 Output data

The finite element numerical model produces graphical output as shown in Figure 4.2. Temperature profiles from individual nodes within the discretised continuum may be extracted from the analysis results, converted and then copied into a Microsoft Excel spreadsheet. These arrays can then be plotted with respect to measured temperatures with relative simplicity. Due to the limited range of the experimental results, only certain points within the cross-section are compared.

Figure 4.2(a) shows a plot of temperature (°C) versus time (hrs) for each individual node within the cross-section and Figure 4.2(b) illustrates a contour plot of the temperature over the cross-section at the specified time period. An envelope plot of the maximum and minimum temperatures, encompassing all the nodal temperatures is also depicted in Figure 4.2(c). This is a practical illustration of the maximum temperature gradient within the concrete element and the time at which it occurs. The envelope plot is the primary purpose of the temperature prediction model as it indicates, for a given concrete material and environmental conditions, the temperature gradients that are likely to be achieved for a given concrete structural configuration. This information is essential for determining the

thermally induced stresses that may develop and, the likelihood of such stresses inducing cracks in the concrete.

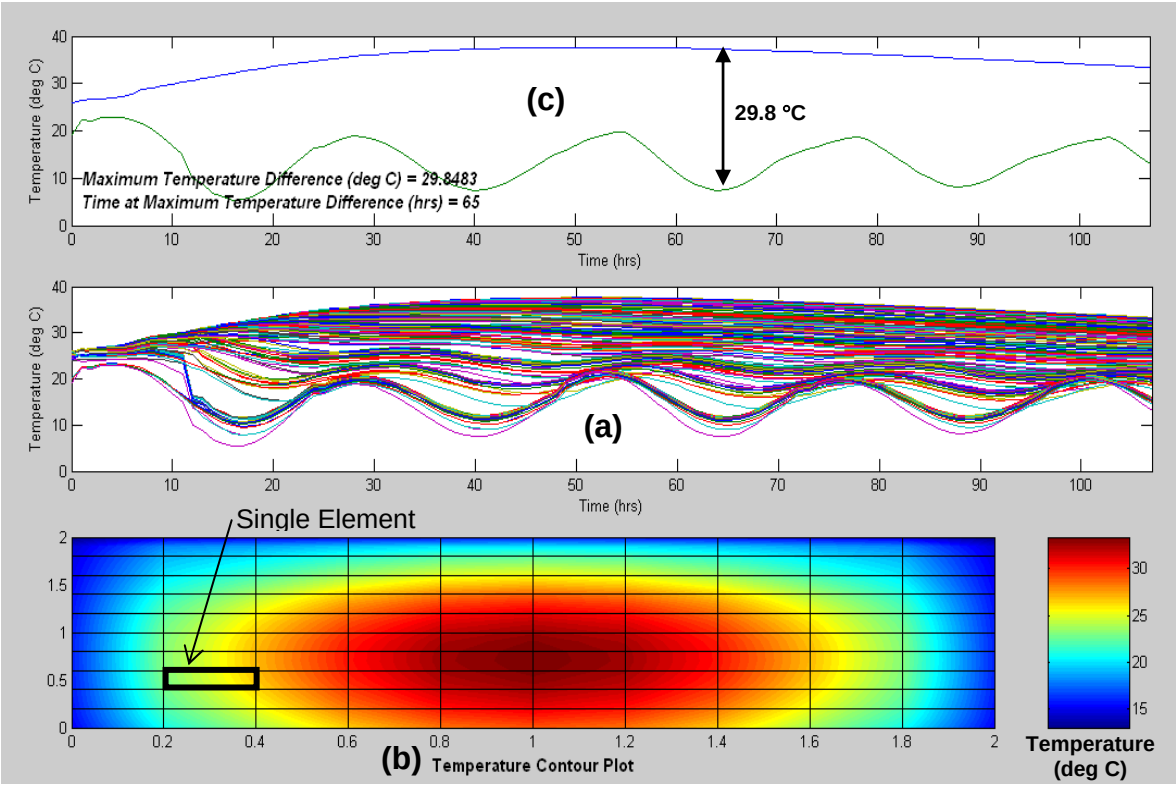


Figure 4.2 An example of the finite element numerical model graphical output

Appendix B should be consulted for a worked example that describes in detail the functionality and procedure required to implement and obtain results from the finite element numerical model.

4.3 EXPERIMENTAL AND NUMERICAL RESULTS COMPARISON

This section is divided into two parts; verification of the finite element model against data obtained from the Katse Dam measurements (Ballim, 2004b), followed by a further validation against data obtained from the laboratory experiment reported by Ballim (2004a).

4.3.1 Temperature – time profiles: Katse verification

Introduction

An unreinforced concrete cube, with dimensions 2m x 2m x 2m, instrumented with thermal probes was cast on the Katse Dam site. The concrete block was cast over a three hour time period with thermal probes numbered 1 to 6 held in position as shown in Figure 4.3.

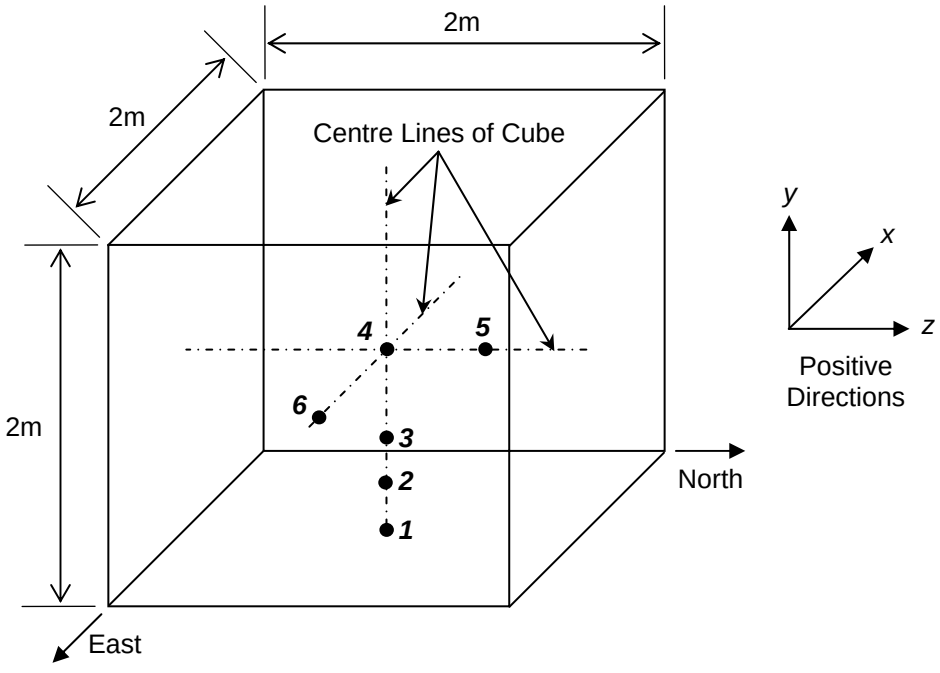


Figure 4.3 Positions of the thermal probes within the 8 m^3 concrete element cast on the Katse Dam site

The coordinates of the centre of the concrete element in millimetres (x, y, z) = (0, 0, 0).

Prior to the casting of concrete, thermal probes were laid out inside the cube. The relative positions were:

1: At the centre of the rock and concrete interface (bottom). (x, y, z) = (0, -1000, 0)
2: 250 mm above the bottom probe. (x, y, z) = (0, -750, 0)
3: 500 mm above the bottom probe. (x, y, z) = (0, -500, 0)
4: At the centre of the concrete cube. (x, y, z) = (0, 0, 0)
5: 500 mm north of the central probe. (x, y, z) = (0, 0, 500)
6: 50 mm from the east face level with the central probe. (x, y, z) = (-950, 0, 0)

Figure 4.4 illustrates the finite element discretisation through a central cross-section of the 8 m^3 cube. Relative numbering of thermal probe positions (numbers in italics) and finite element global node numbers are included.

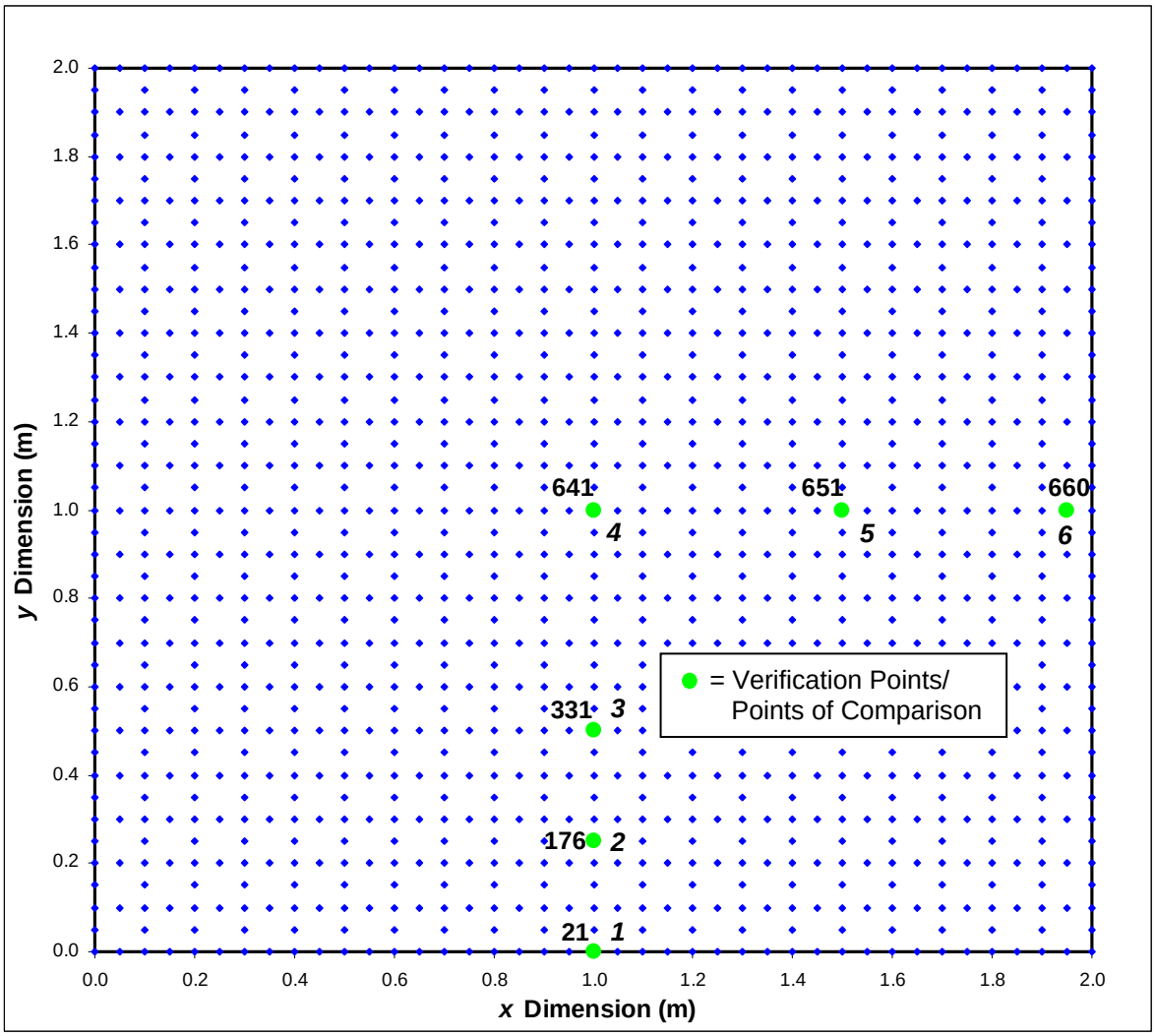


Figure 4.4 Finite element discretisation over the specified cross-section

Temperature versus time profiles are constructed and compared with the measured results by extracting relative data from the finite element numerical model. Each position (positions 1 to 6) within the cross-section has a specific temperature versus time profile that was plotted using the Microsoft Excel Chart Wizard. A plot of measured results and results obtained from the finite difference numerical model are included in every graph.

The finite element numerical model, evaluated over a time period of 107 hours, (this is the time at which the maximum temperature within the cross-section is produced relative to the finite element numerical model) produces the graphical output as shown in Figure 4.5.

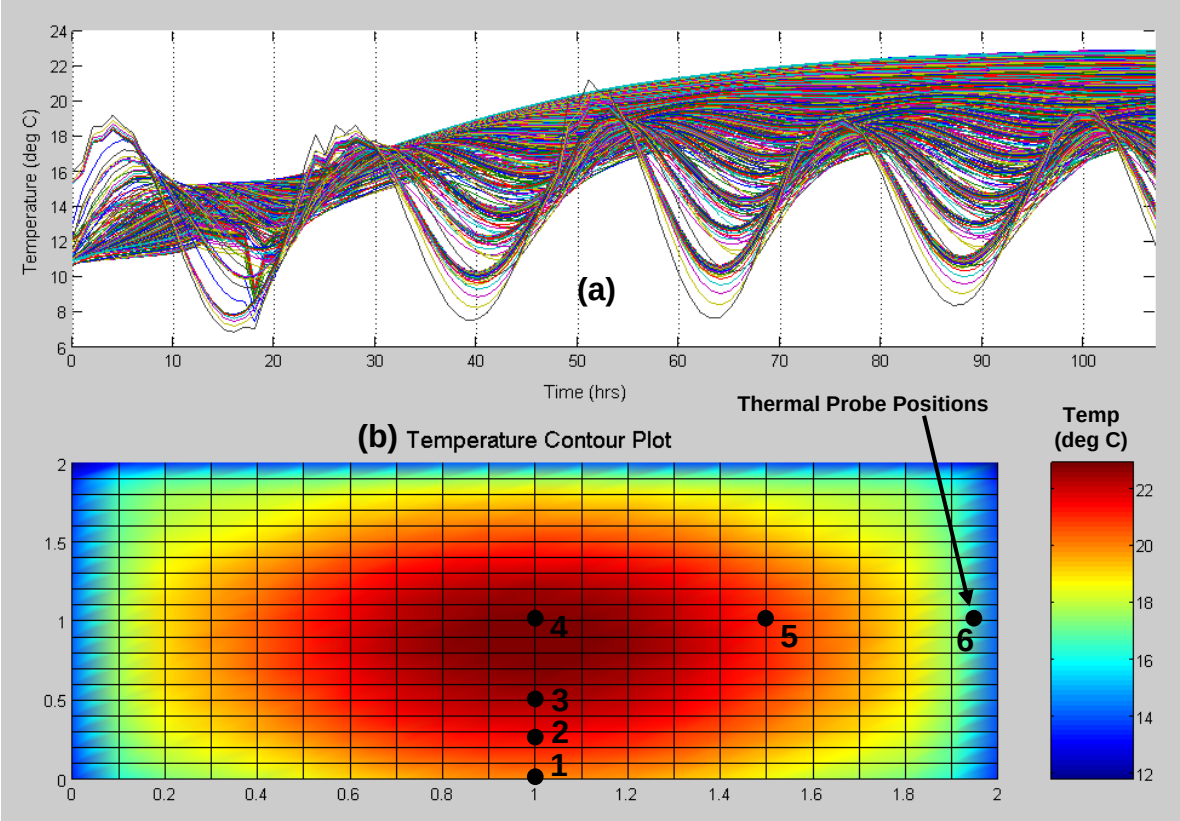


Figure 4.5 **Finite element numerical model graphical output for the Katse verification**

Predicted temperatures obtained from the finite element numerical model and finite difference numerical model (as determined by Ballim (2004b)) including the measured data are now presented at all six positions throughout the cross-section.

Position 1: At the centre of the rock and concrete interface

Measured results were recorded over a 210 hour time span. Thus, for this comparison, the same time period is utilized for the numerical models.

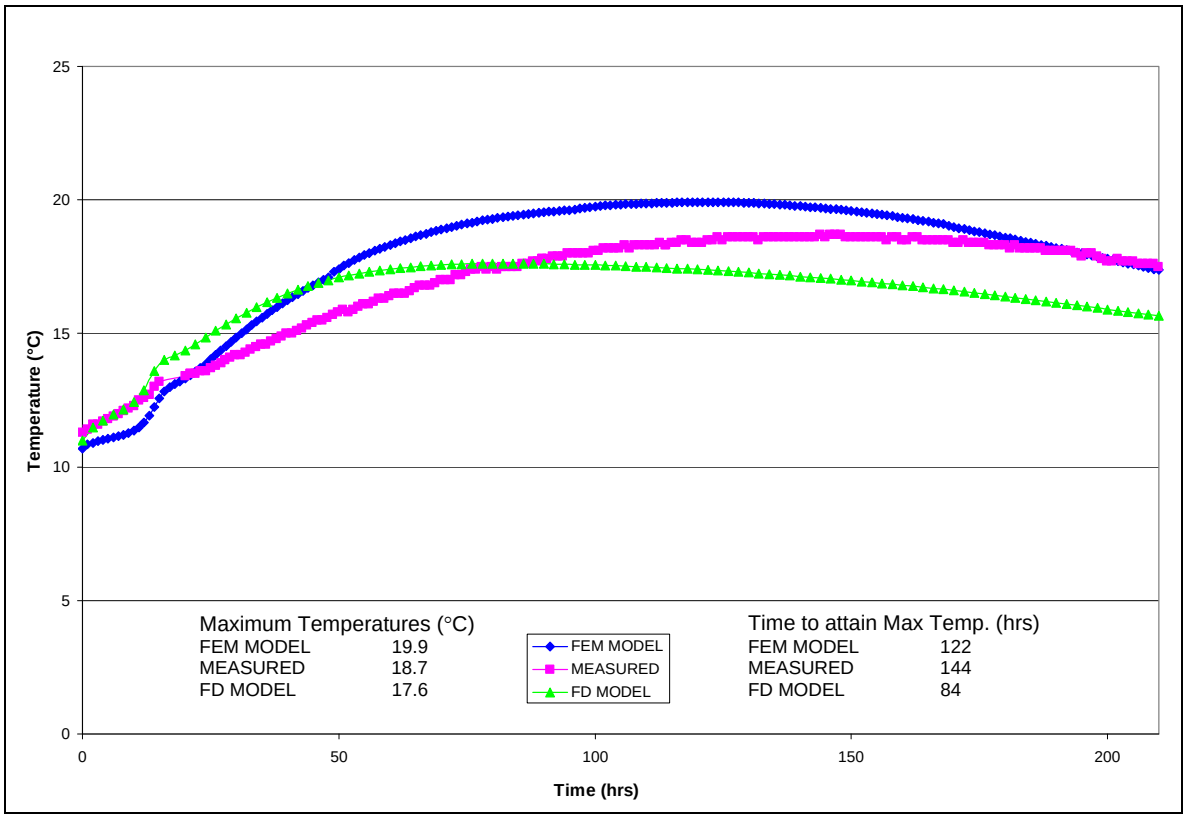


Figure 4.6 Temperature profile plot at position 1

Figure 4.6 reveals that the predicted and measured results are comparatively close. However, the finite element numerical model appears to overestimate the temperature for the greater part of the time period. This inaccuracy can be attributed to convection theory rather than conduction theory being employed to model the interface between the rock and the concrete. Discretising the rock and including it within the finite element analysis, thus treating the surface between the rock and concrete element as a thermal conduction problem rather than the assumed thermal convection problem would significantly improve the results of the finite element numerical model. It is important to note that the proposed technique will substantially increase the amount of computer time required to obtain a solution to the heat transfer problem. Nevertheless, it is argued that this

compromise in computer runtime will not greatly influence the accuracy of the results throughout the cross-section.

The maximum measured temperature was 18.7°C compared with 19.9°C for the finite element model and 17.6°C for the finite difference model. This translates to an absolute error of 1.2°C for the finite element model and 1.1°C for the finite difference model. Relatively small absolute errors indicate that the numerical modelling techniques predict the maximum temperatures that evolve at position 1 with reasonable accuracy. Furthermore, overall temperature predictions remained within an acceptable range throughout the time period.

Additional information such as the time taken for the maximum temperature to be reached can also be deduced from the plots in Figure 4.6. This is an important factor as it specifies the time frame at which maximum thermal stresses, owing to external restraints, are likely to occur within the concrete element. Furthermore it also provides another point of comparison required in the assessment of accuracy for the two numerical modelling techniques.

The measured time period for the maximum temperature to develop at position 1 is 144 hours, compared with the predicted durations of 122 hours for the finite element model and 84 hours for the finite difference model. This translates to an absolute error of 22 hours for the finite element model and 60 hours for the finite difference model. Consequently the finite element numerical model predicts the time period required for the maximum temperature to develop at position 1 to a greater level of accuracy than the finite difference model. It can therefore be deduced that the finite element method is an improved prediction technique at position 1 relative to the finite difference method.

Position 2: 250 mm above the bottom probe

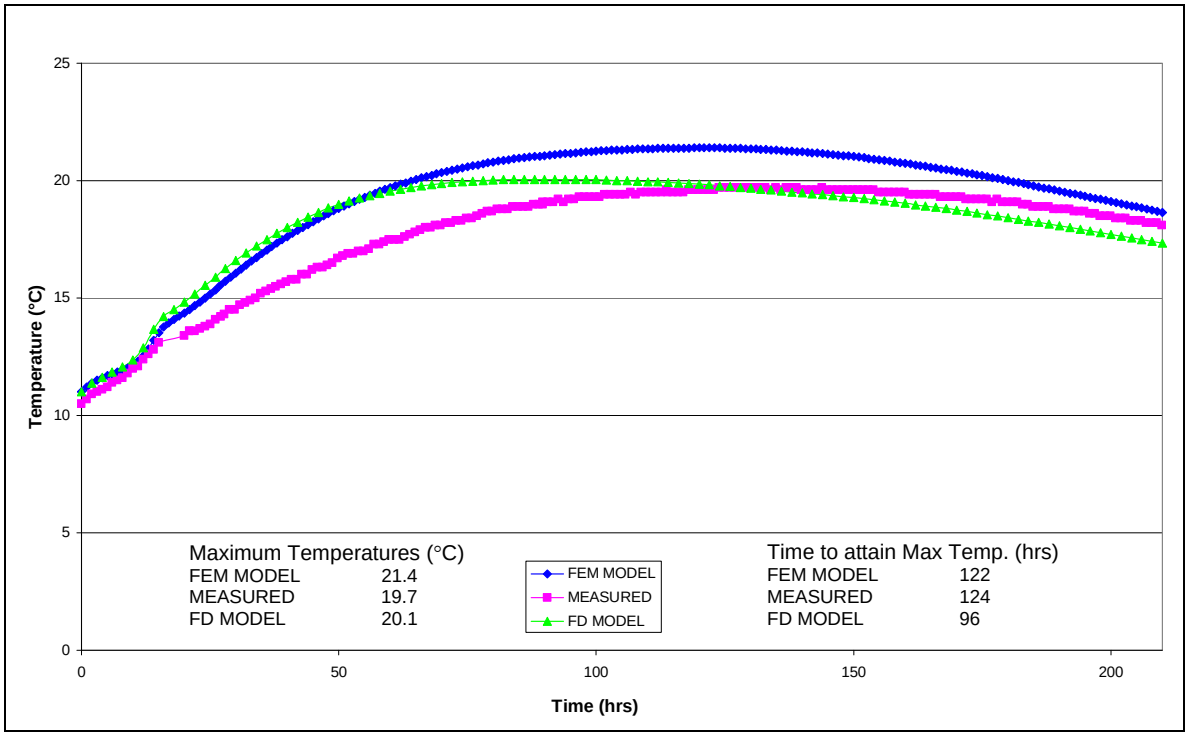


Figure 4.7 Temperature profile plot at position 2

Figure 4.7 shows that the modelled and measured results correlate well, although the finite element numerical model slightly overestimates the temperature for a greater portion of the time period, as with Position 1. This finding can be attributed to inaccuracies carried over from the modelling of the temperature profile at position 1. However, it can be expected that the margin of error will decrease with the proposed improvements to the modelling technique mentioned for position 1.

The absolute difference between the maximum measured temperature and the maximum predicted temperatures is 1.7°C for the finite element model and 0.4°C for the finite difference model. Although the finite difference model predicts the maximum temperature more accurately, the profile of the finite element curve better approximates that of the measured curve. Thus, with a decrease in temperature (i.e. the finite element numerical model plot shifts downwards) the finite element numerical model will yield an improved approximation of the

measured results, in addition to providing a better approximation of the time required to reach the maximum temperature. The time at which the maximum measured temperature occurs is 124 hours compared with the predicted durations of 122 hours for the finite element model and 96 hours for the finite difference model. This translates to an absolute error of 2 hours for the finite element model and 28 hours for the finite difference model.

Position 3: 500 mm above the bottom probe

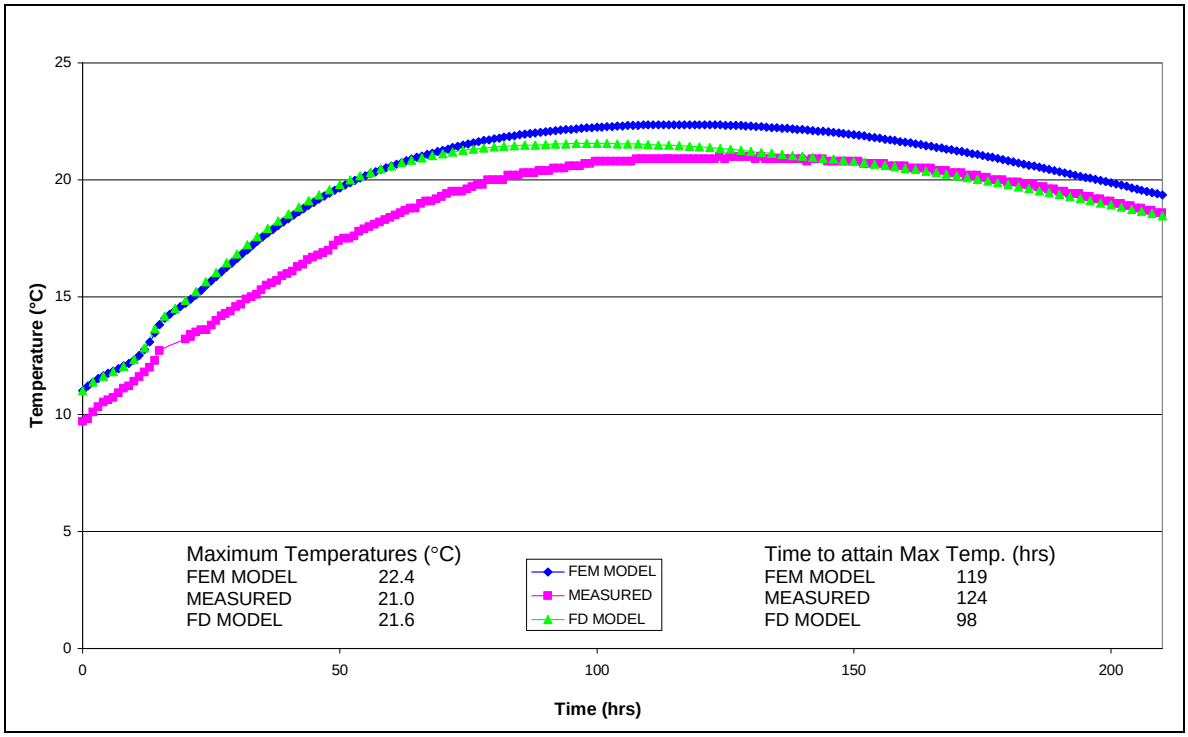


Figure 4.8 Temperature profile plot at position 3

Figure 4.8 shows that the modelled and measured results are similar to the temperature profiles at position 2. The maximum measured temperature was 21.0°C compared with 22.4°C for the finite element model and 21.6°C for the finite difference model. This translates to an absolute error of 1.4°C for the finite element model and 0.6°C for the finite difference model. The measured time period for the maximum temperature to develop at position 3 is 124 hours compared with 119 hours for the finite element model and 98 hours for the finite

difference model. This translates to an absolute error of 5 hours for the finite element model and 26 hours for the finite difference model. The evident inaccuracy and conclusions that can be drawn are rationalised in the same way as for position 2.

Position 4: At the centre of the concrete cube

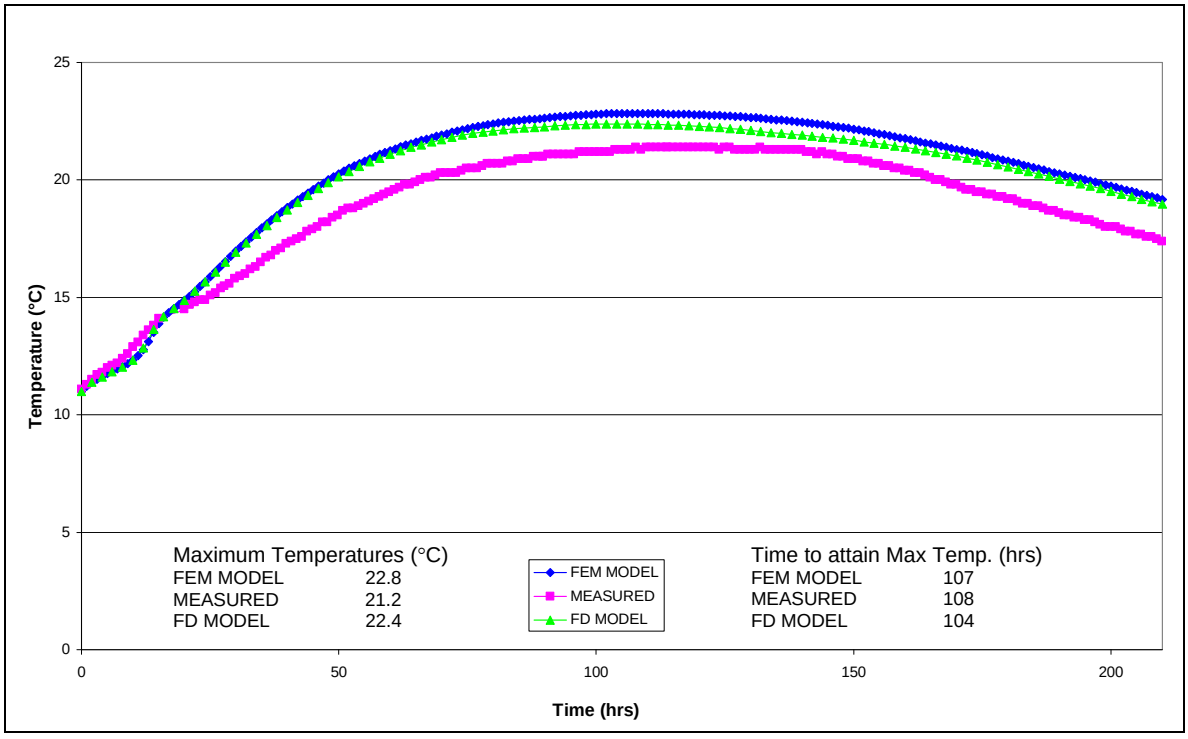


Figure 4.9 Temperature profile plot at position 4

Figure 4.9 reveals a good correlation between the predicted and measured temperature profiles. The maximum temperature should arise at the centre of the concrete block or close to the centre depending on the boundary conditions. From the above figure both the numerical models produce similar temperature profiles. The absolute difference between the maximum measured and predicted temperatures is 1.6°C for the finite element model and 1.2°C for the finite difference model. This error could be attributed to the 3 hour time period that was required to cast the 8 m^3 concrete block. If the concrete block could have been cast instantaneously or more specifically at the rate of placing the test sample in

the adiabatic calorimeter, the measured temperature curve would shift higher and thus yield an improved correlation with the predicted results. A decrease in the finite element numerical model plot (i.e. shift closer to the measured plot) with an increase in accuracy of the predicted temperature at position 1 is also expected.

The absolute difference between the measured and predicted time periods for the maximum temperature to develop at position 4, is 1 hour for the finite element model and 4 hours for the finite difference model. It can therefore be concluded that neither of the numerical models predicts the temperature profile at the centre of the concrete block to a higher accuracy than the other, but that both produce acceptable results.

Position 5: 500 mm north of the central probe

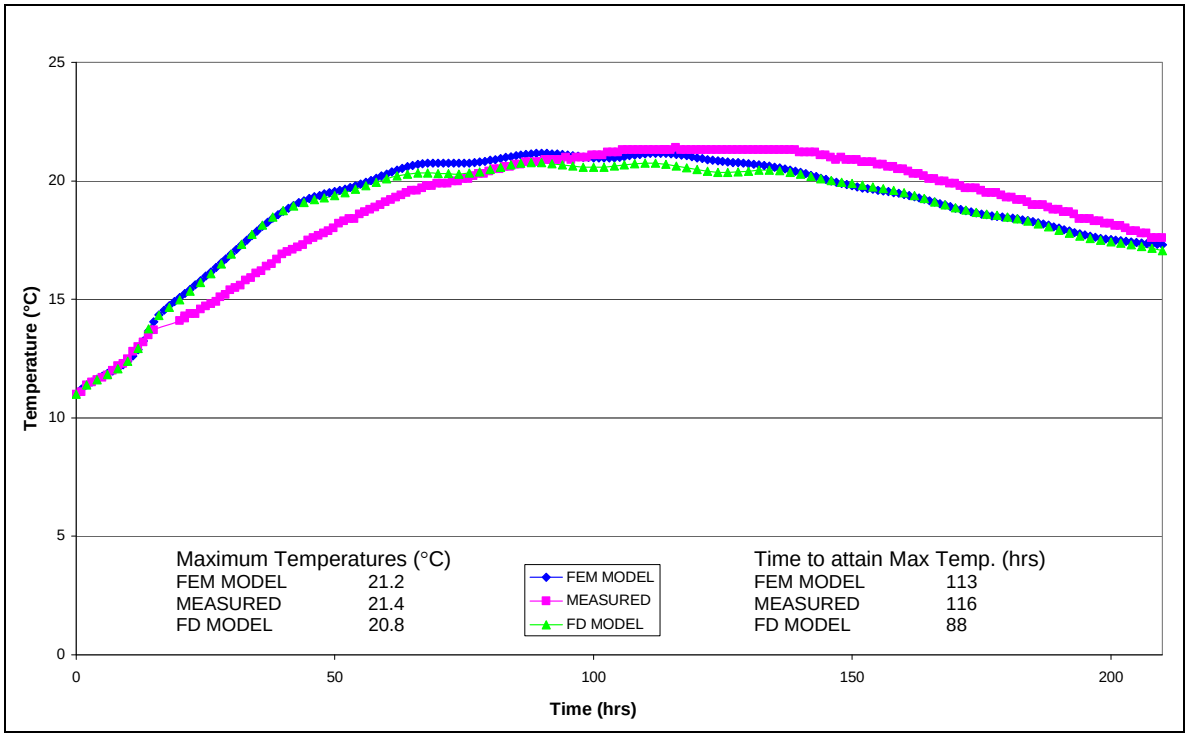


Figure 4.10 Temperature profile plot at position 5

Both the numerical model plots in Figure 4.10 overestimate the temperature for the time period below 100 hours and underestimate the temperature for the time range greater than 100 hours. The presence of small oscillations in the numerical

model plots is due to variations in atmospheric temperature. These oscillations become more prominent closer to the surface of the concrete exposed to the atmosphere.

The absolute difference between the maximum measured and predicted temperatures is 0.2°C for the finite element model and 0.6°C for the finite difference model. The absolute difference between the measured and predicted time periods for the maximum temperature to develop at position 5, is 3 hours for the finite element model and 28 hours for the finite difference model.

A phenomenon readily observed on the graph occurs when the concrete begins to dry and the moisture content decreases (i.e. water is consumed by the hydration reaction and lost by surface evaporation). Consequently the thermal conductivity decreases. This process of evaporation begins from the surface and gradually progresses into the concrete whereas the water consumed by hydration occurs over time throughout the cross-section. Due to the resulting decrease in thermal conductivity, the heat liberated within the concrete cannot dissipate to the surrounding environment as efficiently. This effect has not been taken into account in either of the numerical modelling techniques. Therefore the temperature prediction for the time period greater than 100 hours is lower than the measured temperatures. A reduced thermal conductivity will also dampen the oscillations induced by the atmospheric temperature. This is attributed to a decrease in heat transfer between the concrete and the surrounding environment. The finite element numerical model does not account for the relationship between moisture content and thermal conductivity due to the exponential increase in computer time required when solving a heat transfer problem. Van Breugle (1998) however, does give a linear relationship between moisture content and thermal conductivity relative to the initial concrete temperature and aggregate type. This relationship could be roughly implemented in order to refine the solutions obtained from the finite element numerical model. However, it was decided that the results obtained are currently acceptable as an optimization between computer run time and accuracy.

Position 6: 50 mm from the east face level with the central probe

The numerical modelling techniques estimate the temperature profile just below the surface of the concrete to a low degree of accuracy. As shown in Figure 4.11 the finite element numerical model gives a better approximation than that of the finite difference numerical model. This is revealed through the use of trend lines (i.e. the finite element numerical models' trend line approximates that of the measured values to a greater accuracy).

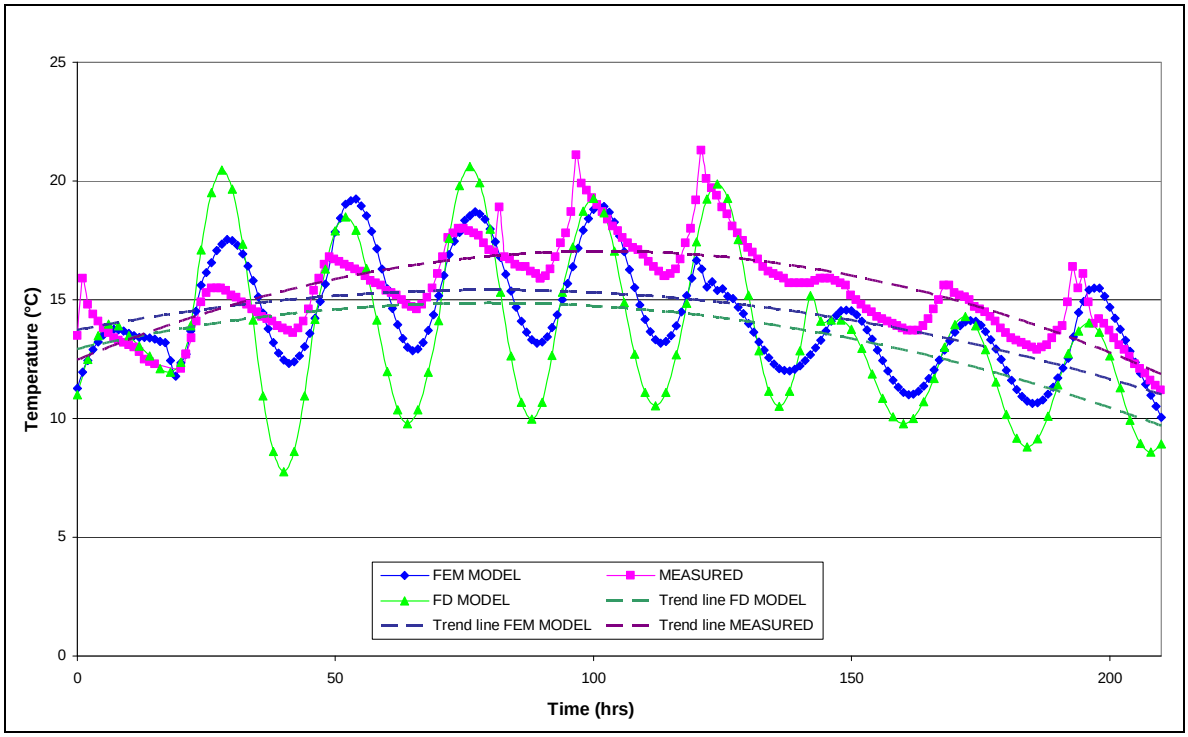


Figure 4.11 Temperature profile plot at position 6

As mentioned previously, the thermal conductivity of concrete should be taken relative to a varying moisture content. As the concrete loses water due to evaporation (occurs rapidly, as position 6 is near to the surface), the moisture content and thermal conductivity decrease, which in turn reduces the amount of heat dissipated to the surrounding environment. Consequently, the predicted temperatures would increase and the oscillations would dampen, resulting in an improved approximation of the measured values.

It should be noted that the major motivation for utilising the finite element approach for the heat transfer analysis was to obtain a more apt definition of the boundary conditions. Even though the varying thermal conductivity of the concrete was not included in the finite element numerical model, the predicted temperatures are closer than those of the finite difference numerical model to the measured temperatures at the boundaries. These boundary conditions can be significantly improved by introducing the moisture content versus thermal conductivity relationship into the finite element numerical model. Refinement of the element mesh would also improve the accuracy of the results.

A significant advantage of the finite element numerical model is that the boundary conditions can encompass a large range of variables and conditions that are difficult or impossible to model with the finite difference numerical model.

In the subsequent verification process, a concrete block was cast under laboratory conditions which ultimately reduces the combined effect of wind and cloud cover. Thus it was expected that the predicted results will correlate to a higher degree of accuracy than those obtained from the Katse verification process.

4.3.2 Temperature – time profiles: Laboratory verification

Introduction

An unreinforced concrete block, with dimensions 1m x 0.7m x 0.7m high, was cast in the University of the Witwatersrand's concrete laboratory with thermal probes held in position as shown in Figure 4.12. The concrete block was cast onto a thin plastic sheet directly on the concrete floor of the laboratory. Due to the controlled environment in the laboratory, the impact of ambient environmental conditions on the test block should be significantly reduced. Therefore wind, cloud cover, solar radiation and environmental temperatures had a negligible impact on the boundary conditions as opposed to the situation for the Katse

verification process. It was expected that the numerical modelling techniques would predict the temperatures to a greater degree of accuracy relative to the previous verification process due to the controlled boundary conditions.

Surface A and Surface B of the concrete block in Figure 4.12 were insulated with a 20 mm thick sheet of high density Styrofoam and a 15 mm layer of timber form-board to simulate a long dimension in the *z* direction. This was done to reproduce effectively a two dimensional heat transfer problem.

The coordinates of the centre of the concrete element in millimetres $(x, y, z) = (0, 0, 0)$

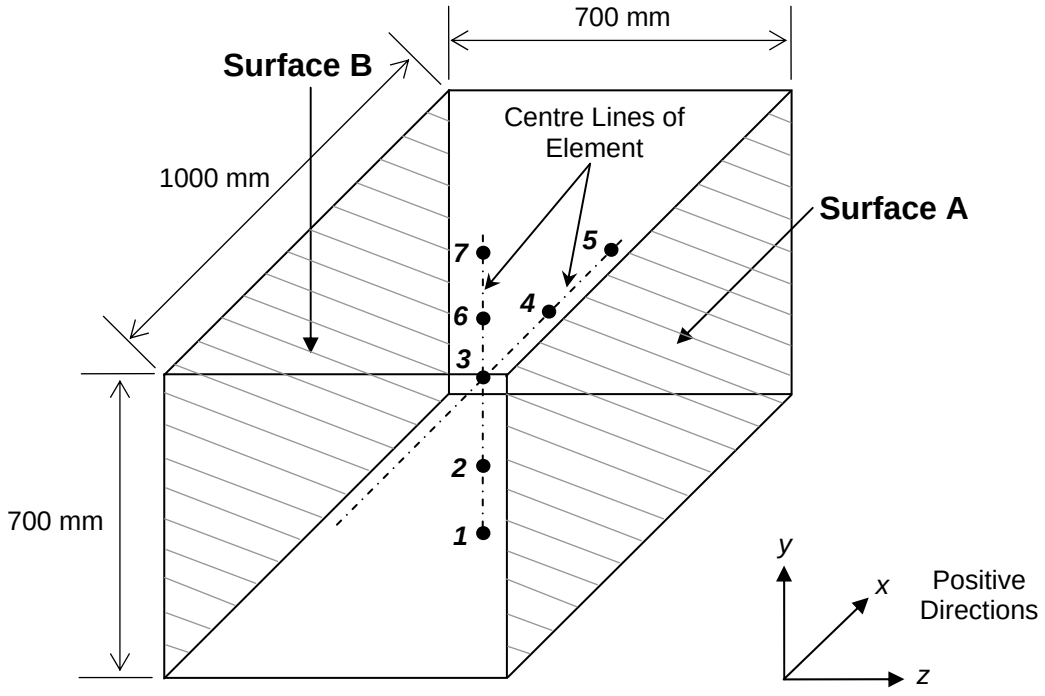


Figure 4.12 Positions of the thermal probes within the concrete element that was cast in the University of the Witwatersrand's concrete laboratory

Prior to casting of the concrete element, the thermal probes were fixed inside the formwork at the positions shown in Figure 4.12. The relative positions were:

1: At the centre of the floor and concrete interface (bottom). $(x, y, z) = (0, -350, 0)$

2: 150 mm above the bottom probe. $(x, y, z) = (0, -200, 0)$

3: At the centre of the concrete element. $(x, y, z) = (0, 0, 0)$

4: 250 mm in the *x* direction from the central probe. $(x, y, z) = (250, 0, 0)$

5: 450 mm in the *x* direction from the central probe. $(x, y, z) = (450, 0, 0)$

6: 150 mm above the central probe. $(x, y, z) = (0, 150, 0)$

7: 50 mm below the top surface above the central probe. $(x, y, z) = (0, 300, 0)$

Figure 4.13 illustrates the finite element discretisation through a central cross-section of the concrete block. The relative numbering of the thermal probe positions (numbers in italics) and finite element global node numbers are included.

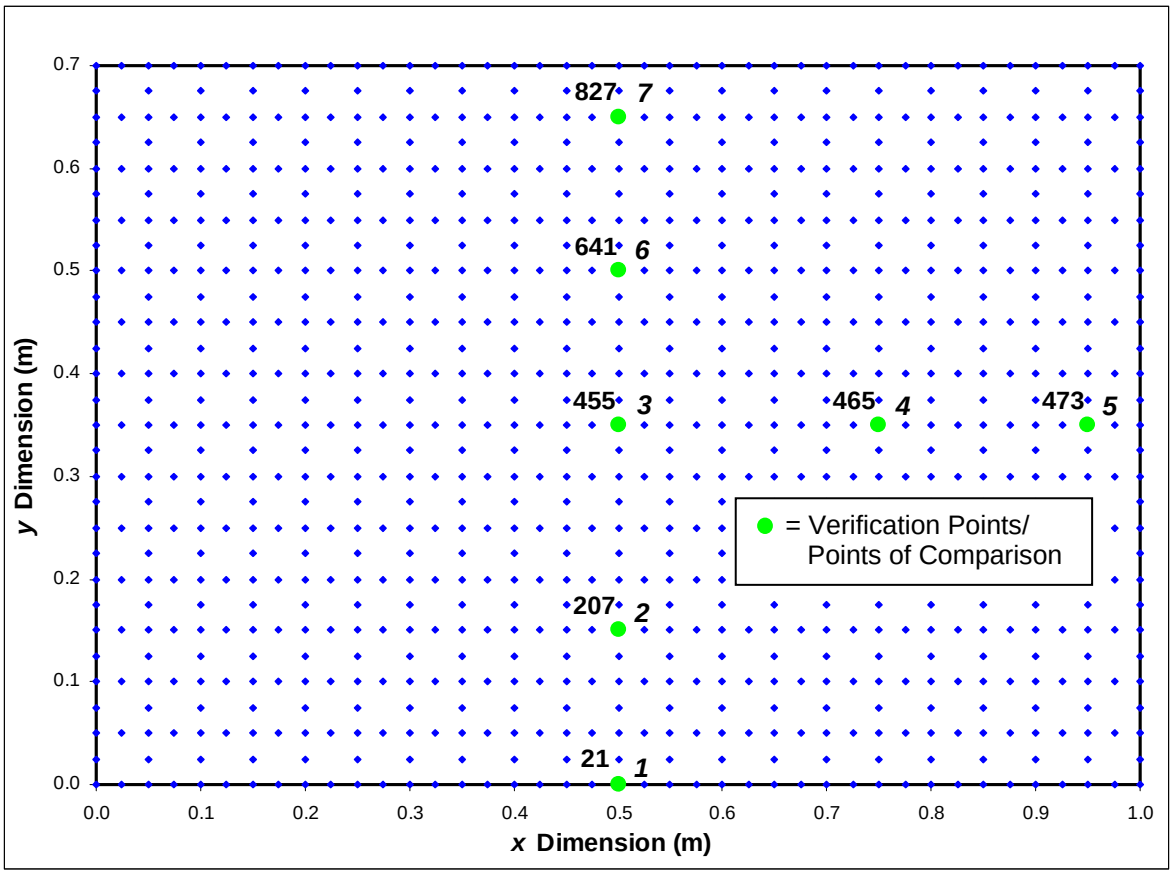


Figure 4.13 Finite element discretisation over the specified cross-section

Temperature versus time profiles can be produced by extracting the relative data from the finite element numerical model. Each position, within the finite element discretisation (positions 1 to 7), has a temperature versus time profile that was plotted using Microsoft Excel Chart Wizard. In addition, each graph includes a

plot of the measured results together with the results obtained from the finite difference numerical model.

The finite element numerical model, evaluated over a time period of 23 hours, (this is the time at which the maximum temperature within the cross-section is produced by the finite element numerical model) produces the graphical output as shown in Figure 4.14. Notably the nodal temperature profile plots (Figure 4.14 (a)) do not oscillate in the sinusoidal fashion as with the previous verification exercise. This is attributed to the stable surrounding environmental conditions.

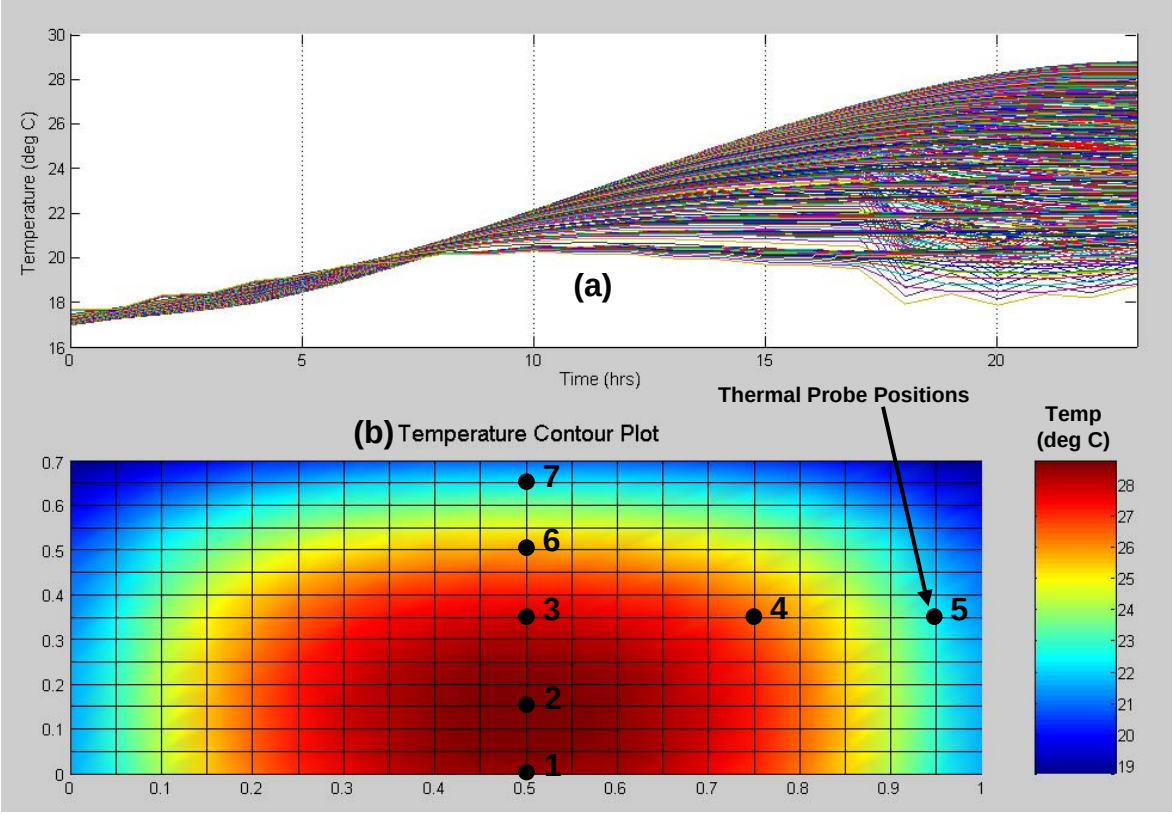


Figure 4.14 Finite element numerical model graphical output for the laboratory verification

Predicted temperatures obtained from the finite element numerical model and finite difference numerical model (as determined by Ballim (2004a)) including the measured data are now presented at all seven positions throughout the cross-section.

Position 1: At the centre of the floor and concrete interface

Measured results were recorded over a 100 hour time span. Thus, for this comparison, the same time period is utilized for the numerical models.

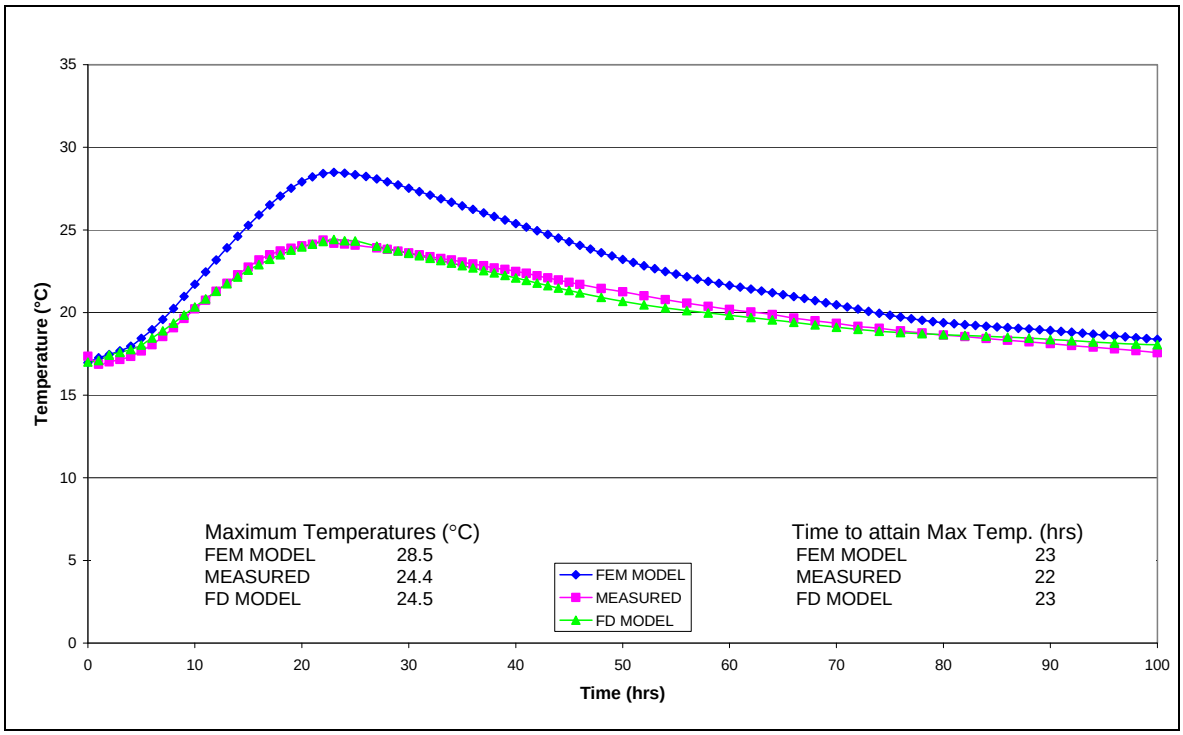


Figure 4.15 Temperature profile plot at position 1

The finite element numerical model in Figure 4.15 produces the same errors as the Katse verification process at position 1. The modelling of the interface between the concrete and the concrete floor is not well formulated in the finite element model. The absolute difference between the maximum measured temperature and the maximum predicted temperatures is 4.1°C for the finite element model and 0.1°C for the finite difference model. This error may seem significant but the temperature versus time profiles for the subsequent six positions is within an acceptable range, indicating that this error does not greatly influence the temperature distribution. It is expected that this modelling error will become more pronounced with a relative decrease in area of the cross-section (i.e. the zone of maximum temperature coincides with the surface that has been incorrectly modelled). This is observed when comparing the absolute differences between the

maximum temperatures of the Katse and laboratory predictions (i.e. the percentage error for the 4 m^2 Katse cross-section is 6.5% and for the 0.7 m^2 laboratory cross-section is 16.9%).

Position 2: 150 mm above the bottom probe

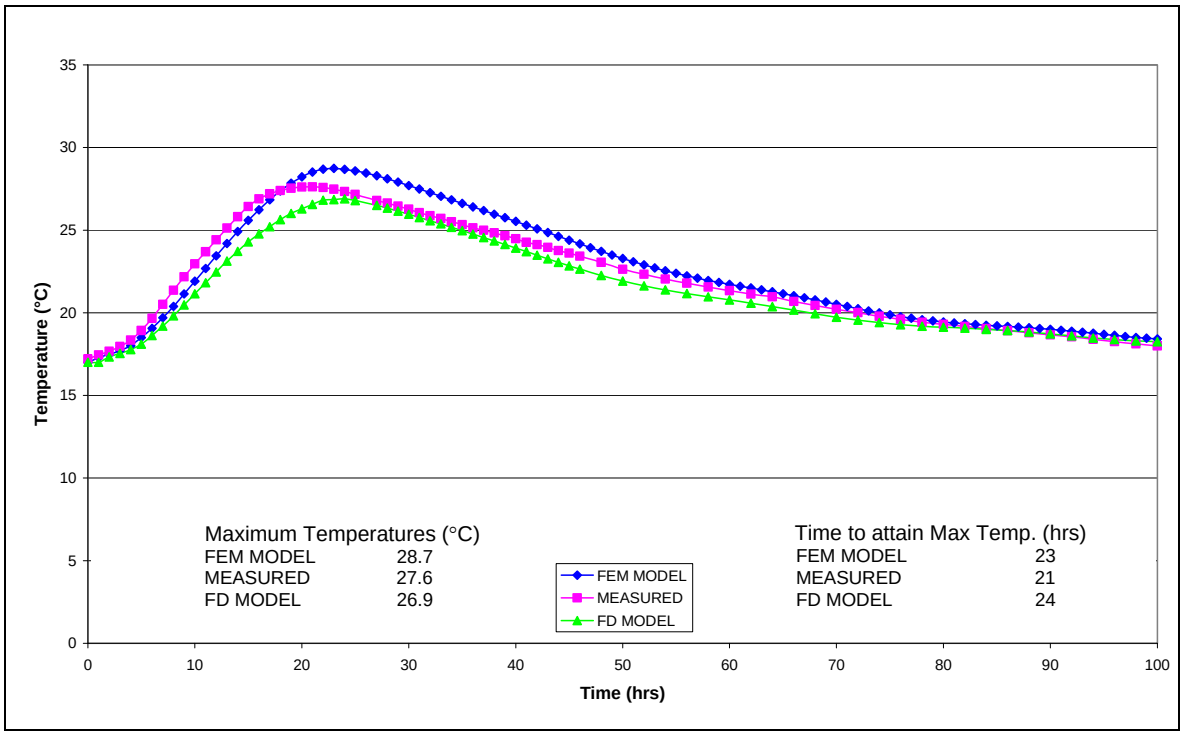


Figure 4.16 Temperature profile plot at position 2

Figure 4.16 shows a good correlation between the predicted and measured temperature profiles. The maximum measured temperature was 27.6°C compared with 28.7°C for the finite element model and 26.9°C for the finite difference model. This translates to an absolute error of 1.1°C for the finite element model and 0.7°C for the finite difference model. The measured time period for the maximum temperature to develop at position 2 is 21 hours, compared with 23 hours for the finite element model and 24 hours for the finite difference model. This translates to an absolute error of 2 hours for the finite element model and 3 hours for the finite difference model.

Position 3: At the centre of the concrete element

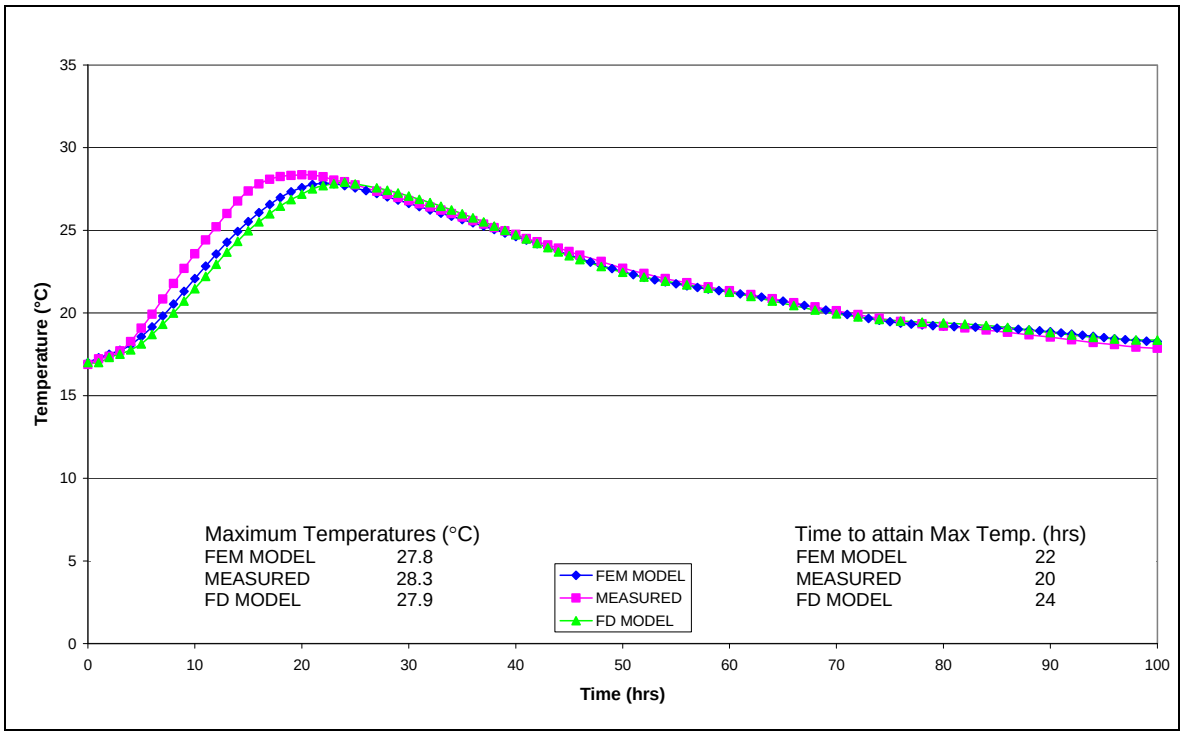


Figure 4.17 Temperature profile plot at position 3

Figure 4.17 shows that the modelled and measured temperature profiles are in good correlation. The absolute difference between the maximum measured and predicted temperatures is 0.5°C for the finite element model and 0.4°C for the finite difference model. The absolute difference between the measured and predicted time periods for the maximum temperature to develop at position 3, is 2 hours for the finite element model and 4 hours for the finite difference model. Small absolute differences indicate a precise solution for the temperature profile at the centre of the concrete element. One notable discrepancy occurring between the modelled and measured profiles is the rate of increase in temperature during the time range, 0 to 20 hours. Within this period of time the measured results increase at a greater rate than the predicted results. This error is most likely due to an inaccuracy associated with the rate of heat liberated existing in either the measured or predicted methods.

Position 4: 250 mm in the *x* direction from the central probe

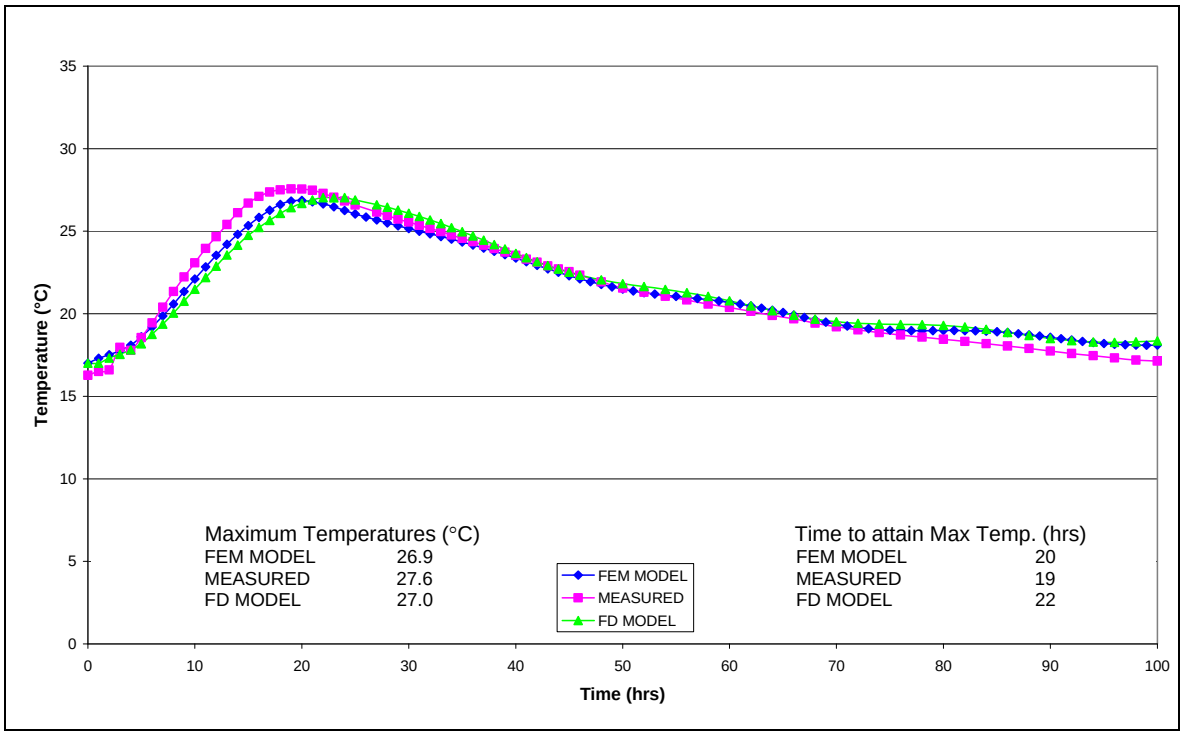


Figure 4.18 Temperature profile plot at position 4

Figure 4.18 reveals a strong correlation between the predicted and measured temperature profiles. The maximum measured temperature was 27.6°C compared with 26.9°C for the finite element model and 27.0°C for the finite difference model. This translates to an absolute error of 0.7°C for the finite element model and 0.6°C for the finite difference model. The measured time period for the maximum temperature to develop at position 4 is 19 hours, compared with 20 hours for the finite element model and 22 hours for the finite difference model. This translates to an absolute error of 1 hour for the finite element model and 3 hours for the finite difference model.

Position 5: 450 mm in the *x* direction from the central probe

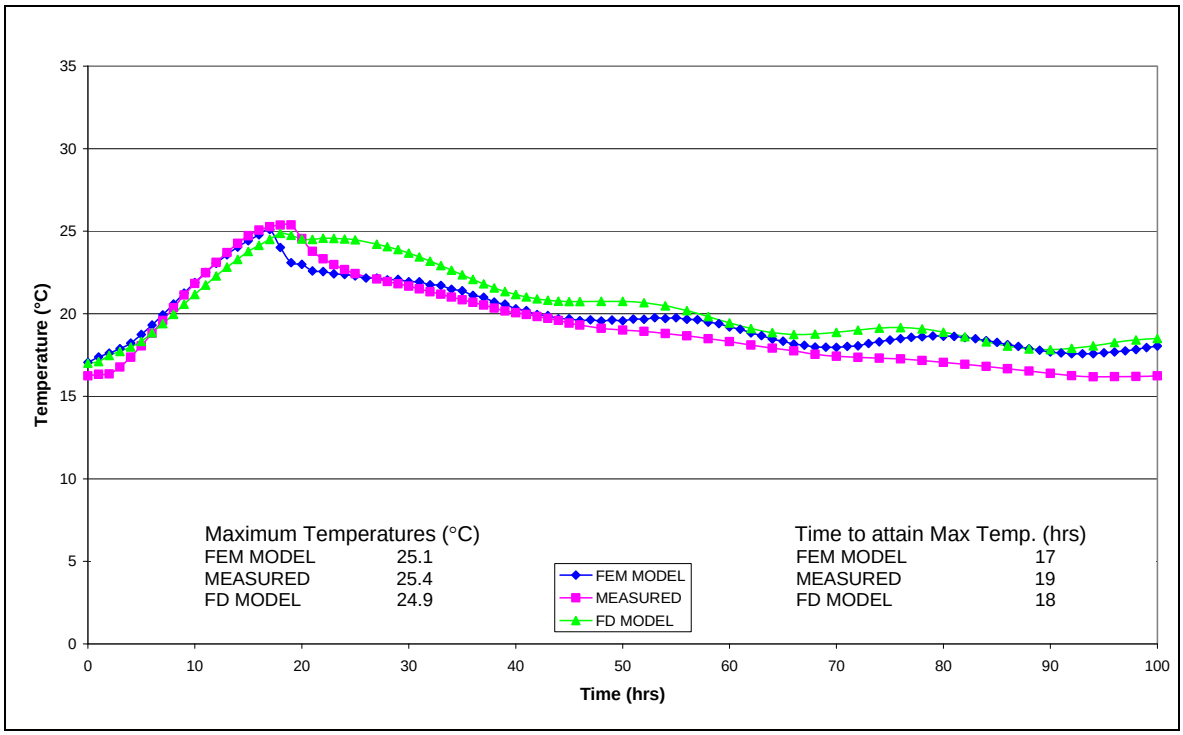


Figure 4.19 Temperature profile plot at position 5

In Figure 4.19, it is evident that the finite element numerical model predicts the temperature profile to a higher level of accuracy. The oscillations of the finite element temperature profile after approximately 40 hours can be attributed to the moisture content versus thermal conductivity relationship referred to previously. The absolute difference between the maximum measured and predicted temperatures is 0.3°C for the finite element model and 0.5°C for the finite difference model. The absolute difference between the measured and predicted time periods for the maximum temperature to develop at position 5, is 2 hours for the finite element model and 1 hour for the finite difference model.

Due to the removal of the formwork, inadvertently providing thermal insulation to the concrete element, a notable drop in temperature occurs from around 18 hours. Note that the removal of the formwork can be observed as a sudden decrease in temperature across all the temperature plots. The measured results show that the formwork removal time was 19 hours after casting, whereas the finite element

results reveal the drop-off from 17 hours (as stipulated with the input data obtained from the personnel who conducted the laboratory test). As expected, this difference in formwork removal time causes a divergence between the results obtained from the measured and predicted temperatures.

The results in Figure 4.19 demonstrate that the finite element numerical model is better able to describe and account for the boundary conditions in the heat flow analysis.

Position 6: 150 mm above the central probe

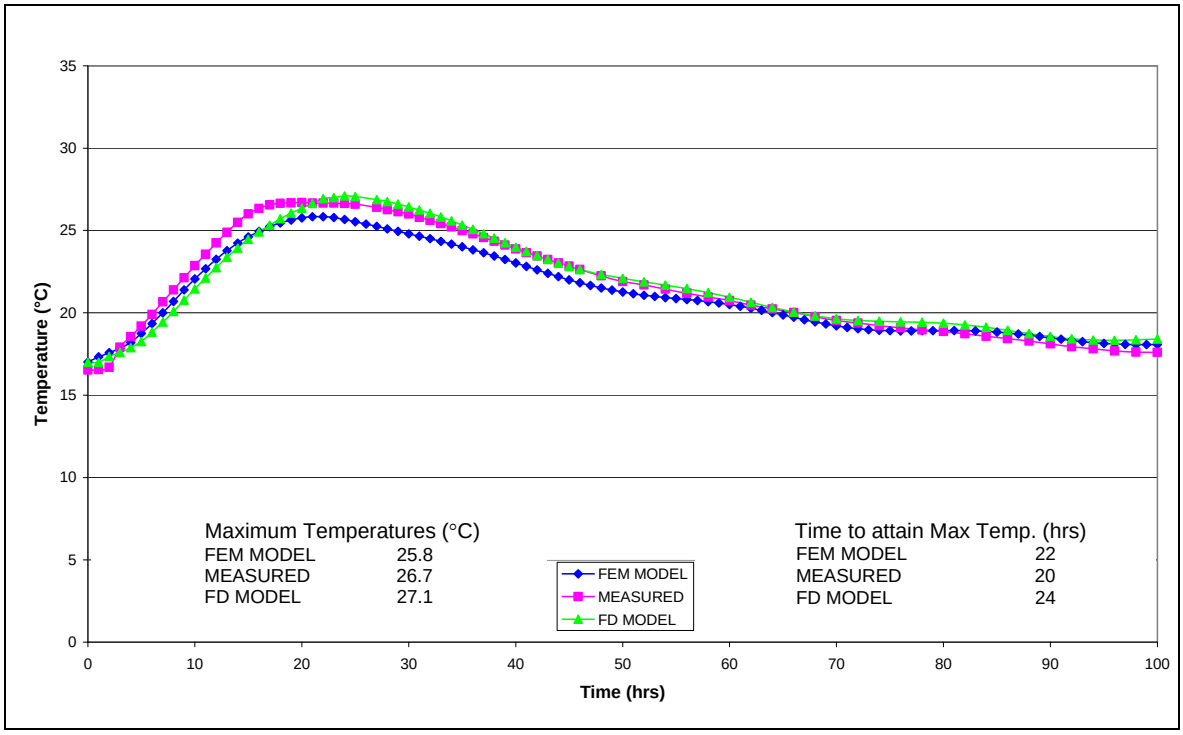


Figure 4.20 Temperature profile plot at position 6

Figure 4.20 shows a narrow discrepancy between the predicted and measured temperature profiles. The maximum measured temperature was 26.7°C compared with 25.8°C for the finite element model and 27.1°C for the finite difference model. This translates to an absolute error of 0.9°C for the finite element model and 0.4°C for the finite difference model. The measured time period for the maximum temperature to develop at position 6 is 20 hours, compared with

22 hours for the finite element model and 24 hours for the finite difference model. This translates to an absolute error of 2 hours for the finite element model and 4 hours for the finite difference model.

Position 7: 50 mm below the top surface above the central probe

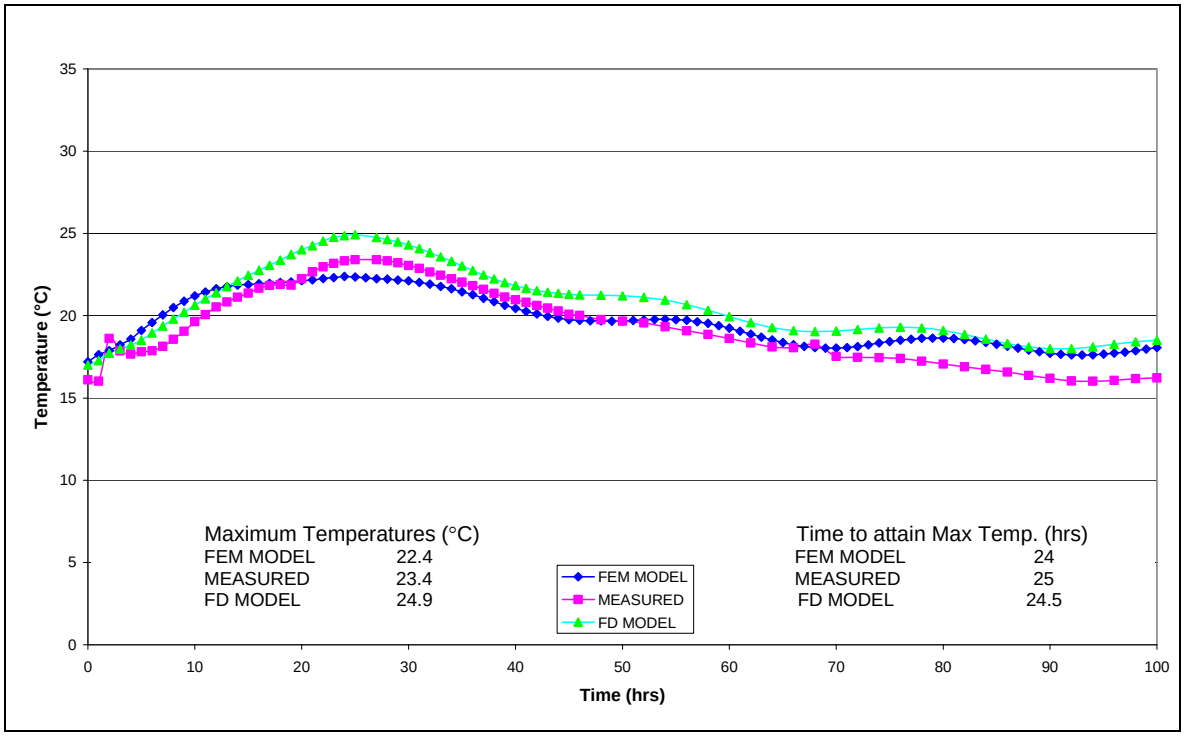


Figure 4.21 Temperature profile plot at position 7

A strong correlation between the modelled and measured temperature profiles is observed in Figure 4.21. The maximum measured temperature was 23.4°C compared with 22.4°C for the finite element model and 24.9°C for the finite difference model. This translates to an absolute error of 1.0°C for the finite element model and 1.5°C for the finite difference model. The measured time period for the maximum temperature to develop at position 7 is 25 hours, compared with 24 hours for the finite element model and 24.5 hours for the finite difference model. This translates to an absolute error of 1 hour for the finite element model and 0.5 hours for the finite difference model. Oscillations due to variations in ambient temperature appear after a time period of approximately 40 hours.

4.4 EXPERIMENTAL AND NUMERICAL RESULTS DISCUSSION

4.4.1 Katse verification

Figures 4.22 to 4.24 illustrate the temperature profiles for all the comparison positions with respect to the measured and predicted temperatures of the finite element and finite difference models. The purpose of these plots is to portray the maximum temperature gradient within the concrete element against the time at which it occurs. This is the primary purpose of the temperature prediction models, as it gives an indication of the extent of thermal stresses that could arise in a particular concrete element with a specified concrete mixture.

Figure 4.22 presents all measured results throughout the concrete element's cross-section, followed by the predicted results of the finite element (Figure 4.23) and finite difference (Figure 4.24) numerical models.

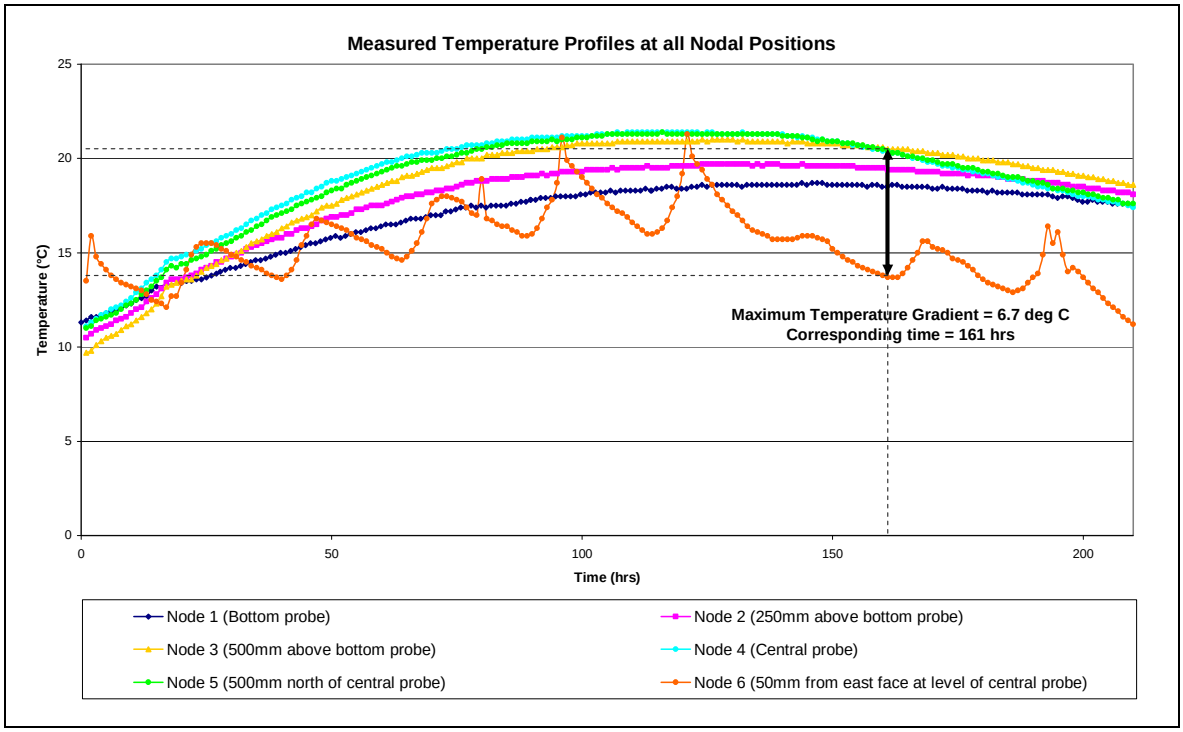


Figure 4.22 Measured temperature profiles – Katse verification

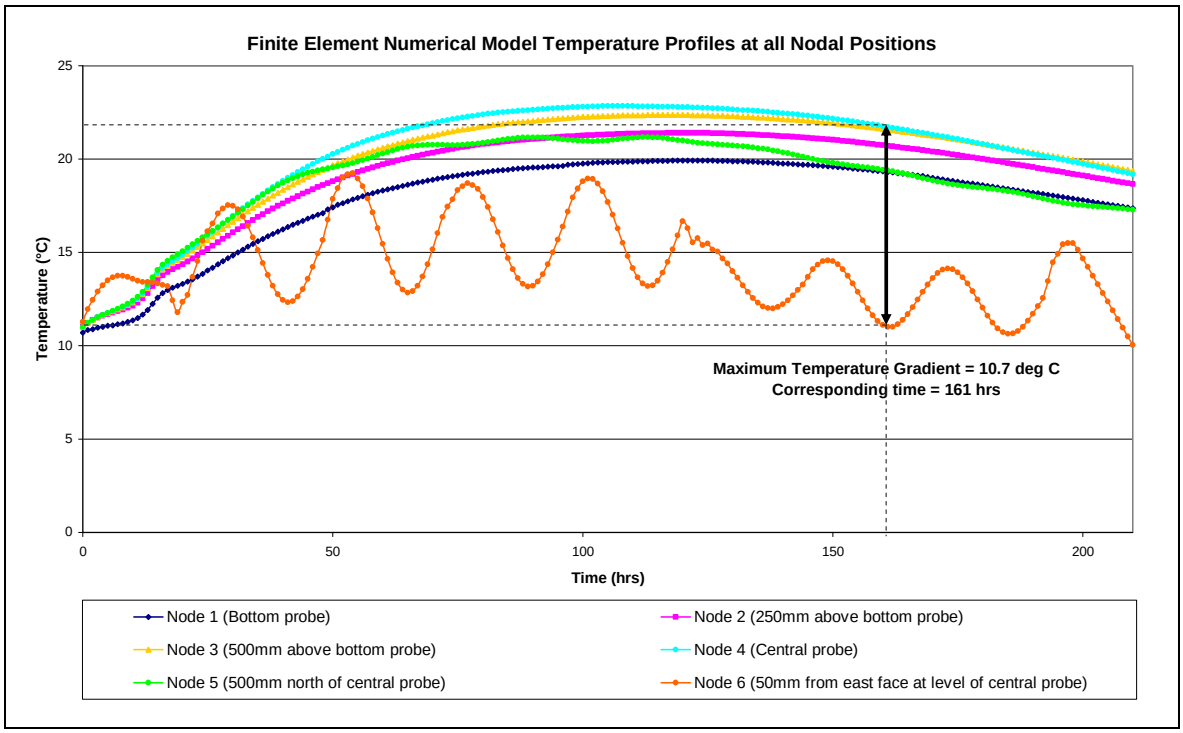


Figure 4.23 FEM temperature profiles – Katse verification

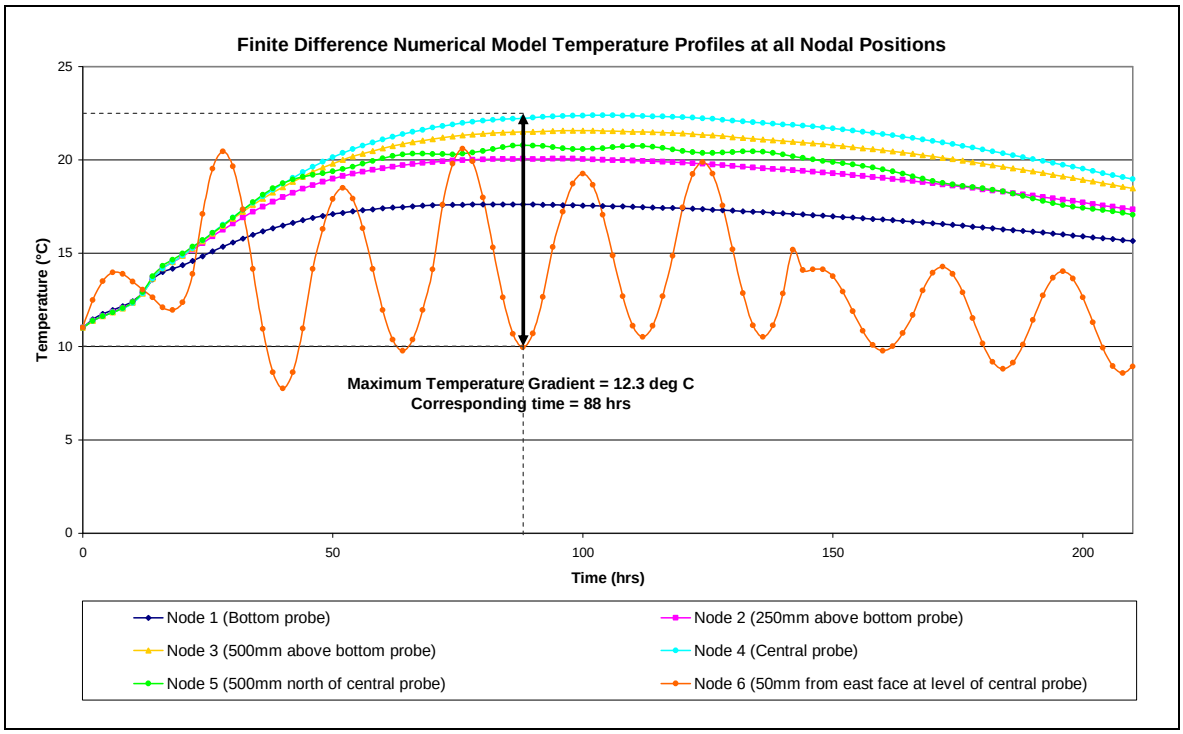


Figure 4.24 FD temperature profiles – Katse verification

The maximum measured temperature gradient was 6.7°C compared with 10.7°C for the finite element model and 12.3°C for the finite difference model. This

translates to an absolute error of 4.0°C for the finite element model and 5.6°C for the finite difference model. The measured time period for the maximum temperature gradient to develop is 161 hours, compared with 161 hours for the finite element model and 88 hours for the finite difference model. This translates to an absolute error of 73 hours for the finite difference model, while the finite element model is an exact prediction. The finite difference model generates a temperature gradient of 11.3°C at 161 hours which is marginally less accurate than that produced by the finite element model.

Evidently the finite element numerical model predicts the maximum temperature gradient and corresponding time to attain this gradient with a higher degree of accuracy than the finite difference numerical model. This comparison substantiates the previous nodal point temperature assessment and confirms that the finite element numerical model is an improved modeling technique.

4.4.2 Laboratory verification

Figure 4.25 represents all measured results throughout the concrete element's cross-section, followed by the predicted results of the finite element (Figure 4.26) and finite difference (Figure 4.27) numerical models.

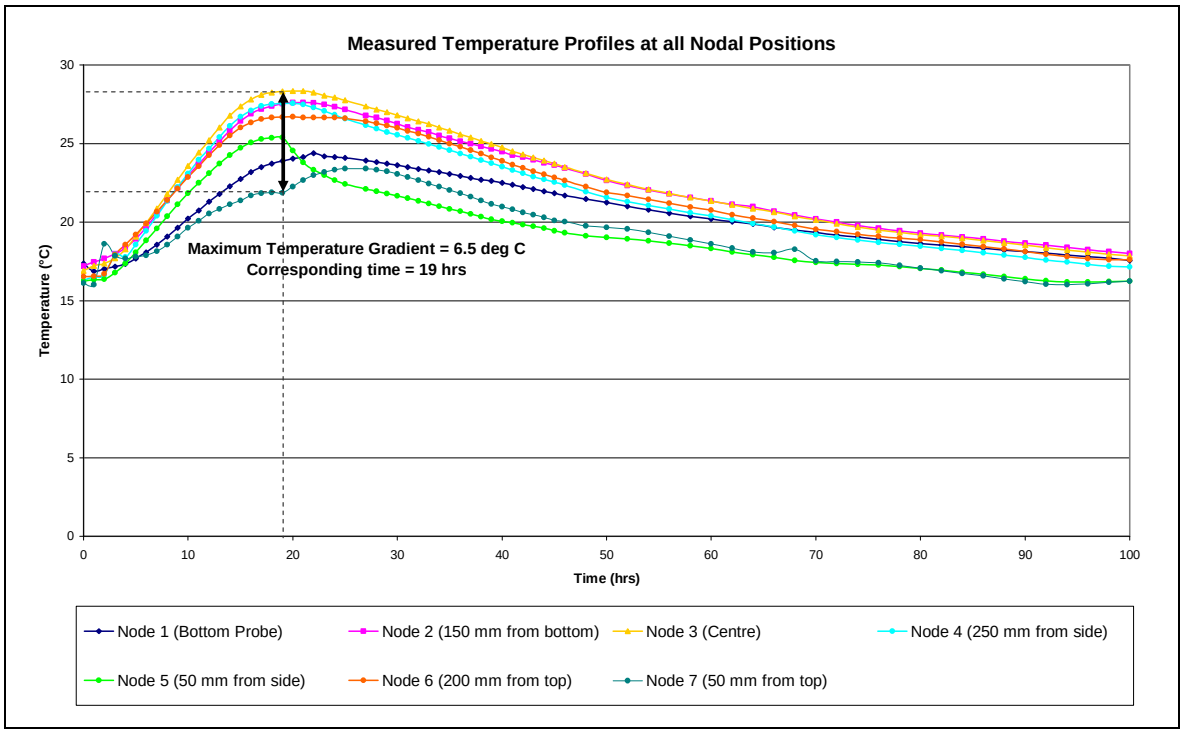


Figure 4.25 Measured temperature profiles – Laboratory verification

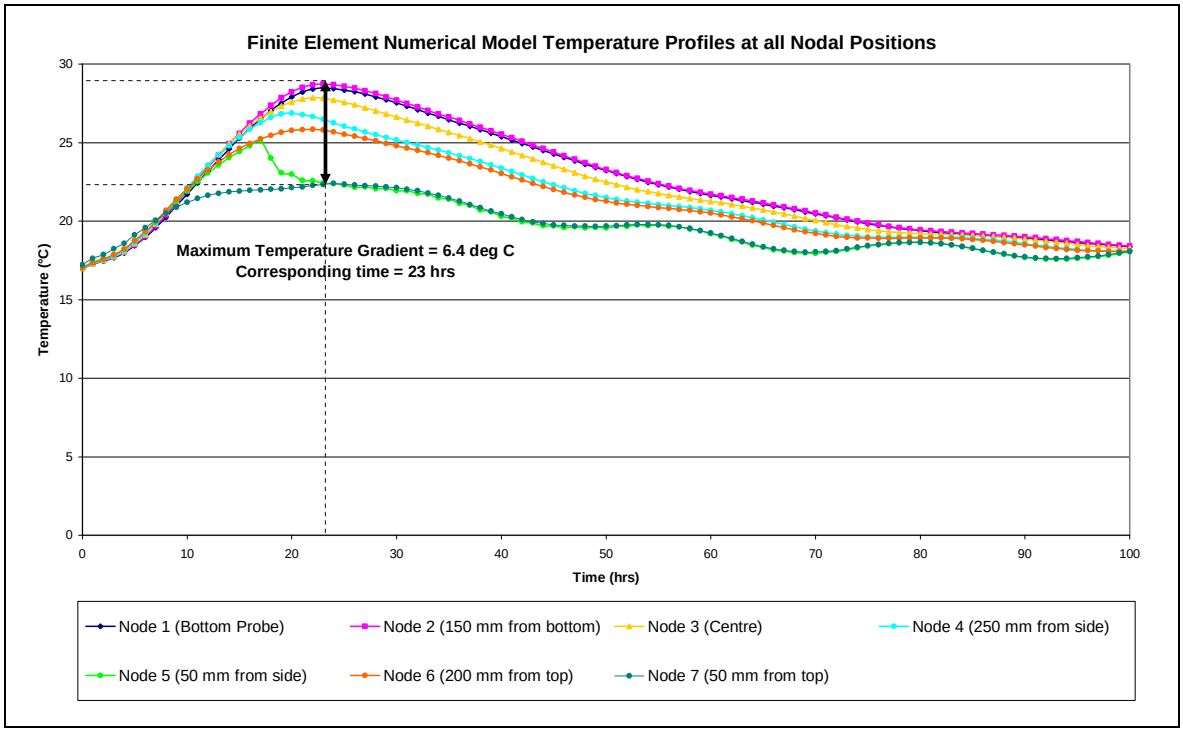


Figure 4.26 FEM temperature profiles – Laboratory verification

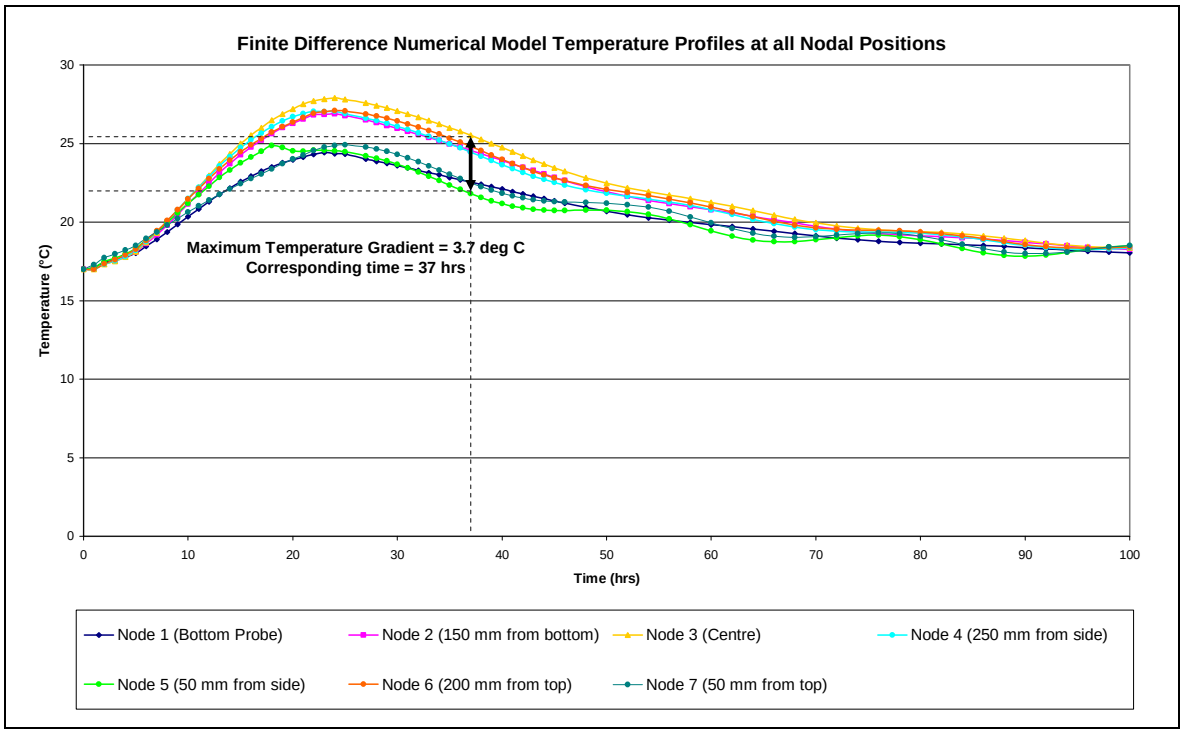


Figure 4.27 FD temperature profiles – Laboratory verification

The maximum measured temperature gradient was 6.5°C compared with 6.4°C for the finite element model and 3.7°C for the finite difference model. This translates to an absolute error of 0.1°C for the finite element model and 2.8°C for the finite difference model. The measured time period for the maximum temperature gradient to develop is 19 hours, compared with 23 hours for the finite element model and 37 hours for the finite difference model. This renders an absolute error of 4 hours for the finite element model and 18 hours for the finite difference model. The temperature gradient calculated by the finite difference model at 19 hours is 3.1°C which is less accurate than that of the finite element model.

Once again the finite element numerical model distinctly predicts the maximum temperature gradient and corresponding time with superior accuracy. Making use of the results obtained from the finite difference numerical model leads to an underestimation of the potential stress and its effect on the likelihood of cracking in the concrete.

4.4.3 Conclusion

The finite element numerical model predicts the temperature profiles within the concrete elements to an acceptable degree of accuracy. The maximum absolute error relative to the highest liberated temperature within the concrete elements is 1.7°C (excluding the temperature profile at position 6) and 4.1°C for the Katse and laboratory verifications respectively. Suggestions to reduce further the inaccuracies of the finite element numerical model are: to include a function that relates moisture content to thermal conductivity, and to implement a more accurate modelling technique at the interface of the concrete element and founding substrate.

A further concern pertaining to the validity of the predicted results is the accuracy of the measured results. As mentioned previously, casting of the instrumented concrete block at Katse occurred over a period of three hours. Therefore a certain amount of liberated heat was not accounted for. With the high rate of temperature increase arising during the early stages of hydration, the three hour time period could have had an impact on the measured results. Consequently, the experimental data is likely to be higher at the early stages and would thus correlate better with both numerical modelling techniques.

Furthermore, after reviewing the measured results for the Katse verification exercise, it is evident that the temperature versus time plots decrease from around the 15 hour time interval and deviate from the predicted results. No measured results are available for the 15 to 20 hour time period after casting. This gap in collected readings was most likely a consequence of the time required to remove the formwork.

An additional reason for the observed deviation between the measured and predicted results could be the incorrect specification of the heat transfer coefficient between the concrete and the atmosphere following formwork removal. The acquired value of 30W/m^{2}K could be regarded as an

underestimation resulting from factors such as wind speed. This factor is of great importance if better approximations of the temperatures are to be achieved. A means of solving this problem is the development of a statistical model which could predict boundary conditions such as wind speed, cloud cover and solar radiation from meteorological data, which could then operate in tandem with the finite element numerical model.

Moreover, the temperature measurements, attained in the Katse verification, originated from a three dimensional block of concrete that was not insulated in the third dimension. This results in the dissipation of thermal energy in the third dimension, leading to a possible deviation from the predicted results.

4.4.4 Thermal cracking propensity and control

Based on the output produced from a random example shown in Figure 4.28 (A), the application of the finite element numerical model is an iterative process if the temperature differentials predicted are greater than those allowed.

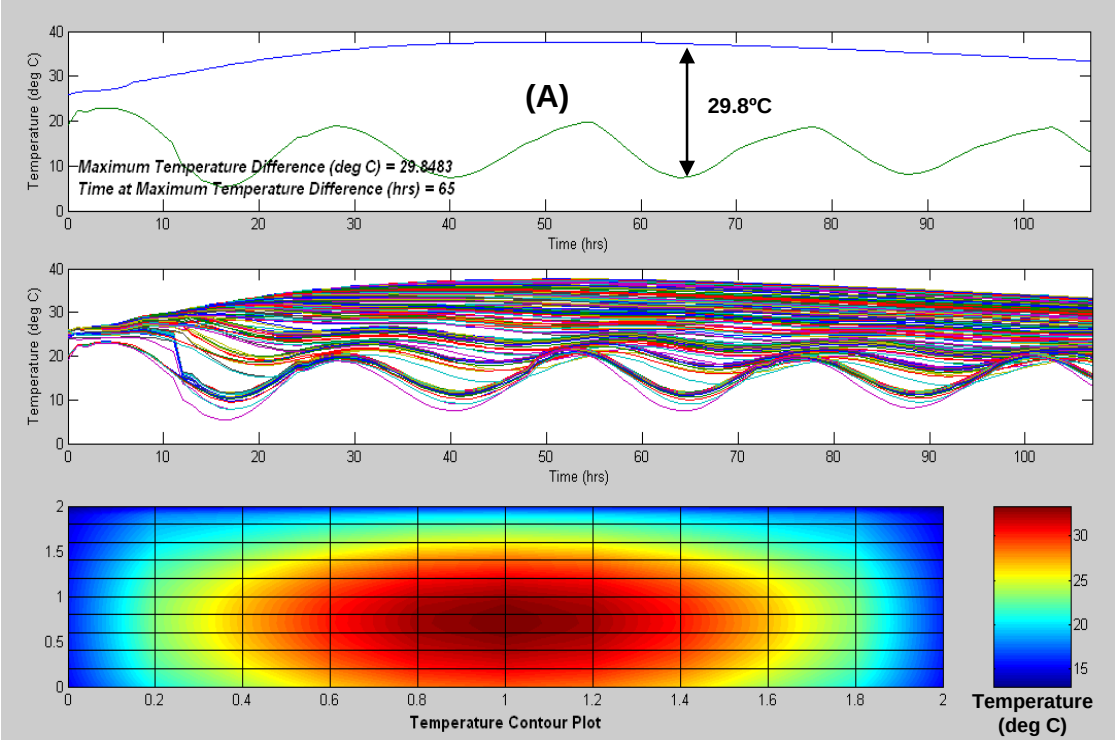


Figure 4.28 An example of the finite element numerical model graphical output

The figure shows a maximum temperature differential of 29.8 °C, which is greater than the “rule of thumb” value of 20 °C. Consequently, the likelihood of thermal cracking occurring in the concrete element is high.

Preventative measures must be implemented if the concrete mixture design cannot be altered. Options available to engineers include:

- Covering the exposed surfaces of the concrete element using polystyrene blocks or other insulating materials as soon as possible after casting. This would increase the surface temperature and reduce the temperature differential between the surface and the core. Care must be taken however, in assessing the appropriate time for the removal of the coverings to ensure that no thermal shock is experienced by the concrete. This would result in multiple hairline surface cracks. Thermal probes cast into the concrete element, linked to a data logger, are used to measure the temperature differentials in order to obtain insulation striking times. The finite element model can account for insulated surfaces through the adjustment of the convective heat transfer coefficient.
- Lowering the casting temperature of the concrete through the use of chilled water reduces the cooling effect of the surrounding environment, resulting in lower temperature differentials. The use of liquid nitrogen for cooling fresh concrete prior to casting is practiced although only in very limited cases.
- Introducing conduits within mass concrete elements that transport chilled water is commonly used in dam structures where extremely large concrete blocks are cast. The principle of this process is to withdraw heat from the centre of the block to minimise the temperature differential. The temperature of the chilled water should be controlled as large temperature differentials potentially occur around these conduits, resulting in extensive cracking.
- The use of plywood as insulating formwork assists in lowering the temperature differentials. The current simulation allows various types of thermal conductivities for formwork material to be modelled.

- Formwork striking times can be extended to minimise the cooling effect of the surrounding environment, thus reducing the temperature differential. This is also accounted for in the present model.
- Reinforcing the concrete element in three directions with a dense mesh can limit the amount of cracking. However, with the bond strength still developing, the amount of reinforcing required is substantial, resulting in significant cost implications.
- Modifications to the water-cement ratio will shift the maximum temperature peak and assist in balancing the temperature gradient. Adiabatic calorimetry would be required to track the changes of water-cement ratios and the corresponding time required to attain maximum temperatures.
- If none of the above methods are acceptable or possible, the engineer would be required to change the concrete mixture design and re-apply the model, resulting in an iterative process. It is proposed that a cement extender be introduced. This assists with the design of a concrete mixture with low early strength development.

Future developments of the finite element numerical model may incorporate several of the above measures. These modifications will be discussed in chapter 6.

4.5 SENSITIVITY ANALYSIS

A sensitivity analysis was performed to determine the effects of varying the element aspect ratio, element size and orientation. These results were then plotted against measured data to demonstrate the importance of selecting an aspect ratio as close to unity as possible with elements that are sufficiently small.

This analysis was executed with the Katse verification results only. Figures 4.29 to 4.32 indicate different element arrangements for the 2 m x 2 m cross-section of the Katse verification process.

Sensitivity Analysis 1:

Evidently the eight noded quadrilateral isoparametric elements are excessively elongated in the *x* direction. Therefore, it is anticipated that the finite element numerical model will represent a poor approximation.

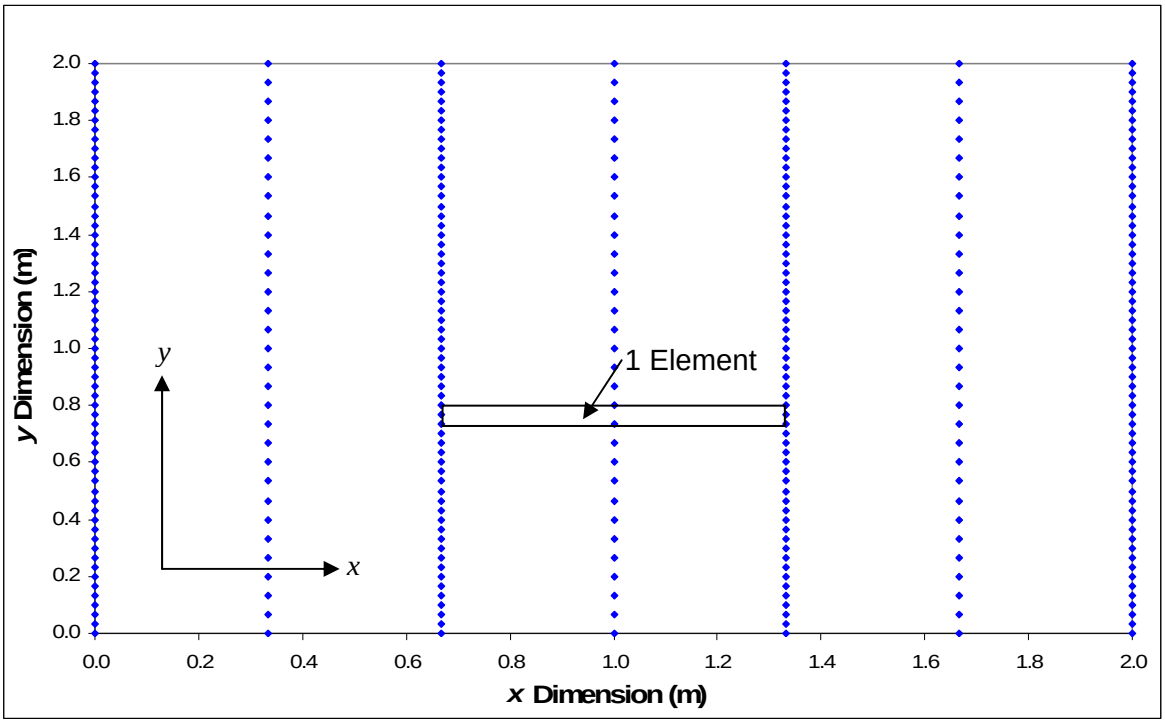


Figure 4.29 Sensitivity analysis 1 relative to the Katse verification

- Element size in the *x* direction = 0.667 m
- Element size in the *y* direction = 0.0667 m
- Aspect ratio = 0.667 m ÷ 0.0667 m = 10
- Number of elements in the *x* direction = 3
- Number of elements in the *y* direction = 30
- Total number of elements = 90

Sensitivity Analysis 2:

Once more the eight noded quadrilateral isoparametric elements are markedly stretched in the *y* direction. Consequently the finite element numerical model is expected to be a substandard estimation.

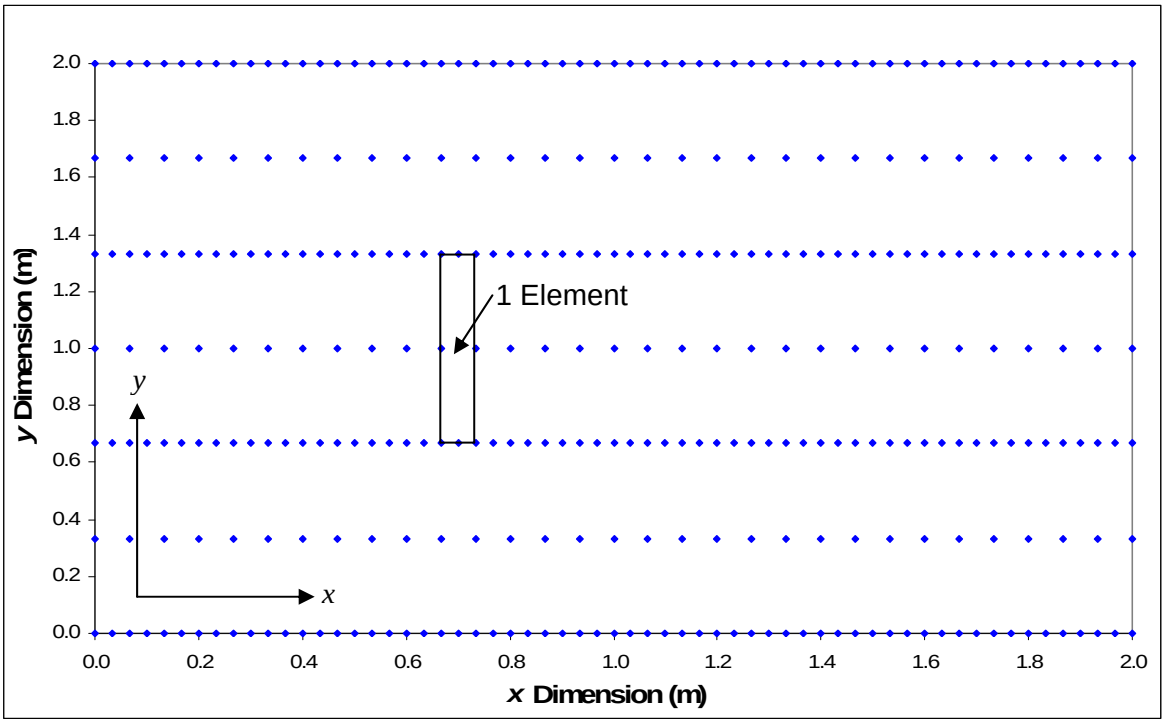


Figure 4.30 Sensitivity analysis 2 relative to the Katse verification

- Element size in the *x* direction = 0.0667 m
- Element size in the *y* direction = 0.667 m
- Aspect ratio = 0.667 m ÷ 0.0667 m = 10
- Number of elements in the *x* direction = 30
- Number of elements in the *y* direction = 3
- Total number of elements = 90

Sensitivity Analysis 3:

The eight noded quadrilateral isoparametric elements can be viewed as large with respect to the cross-section. Hence, it is anticipated that the finite element numerical model will typify a poor approximation.

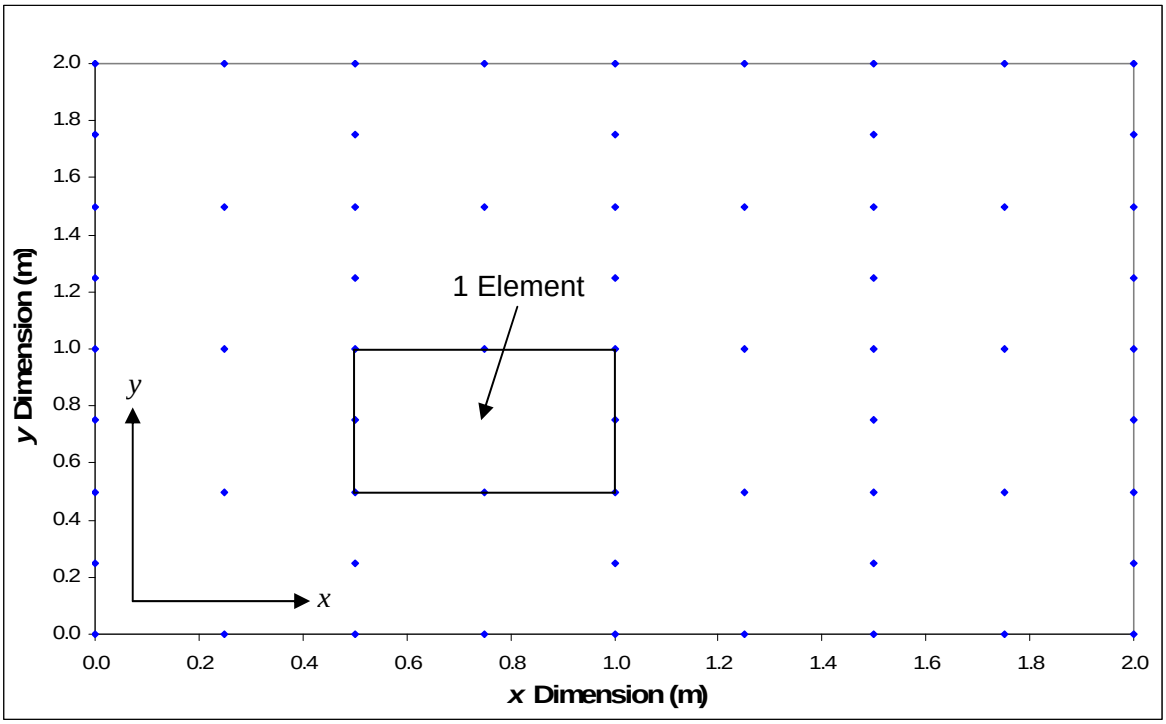


Figure 4.31 Sensitivity analysis 3 relative to the Katse verification

- Element size in the *x* direction = 0.5 m
- Element size in the *y* direction = 0.5 m
- Aspect ratio = 0.5 m ÷ 0.5 m = 1
- Number of elements in the *x* direction = 4
- Number of elements in the *y* direction = 4
- Total number of elements = 16

Sensitivity Analysis 4:

The eight noded quadrilateral isoparametric elements are small with respect to the cross-section. Accordingly it is anticipated that the finite element numerical model will provide a suitable approximation. This analysis can be compared with the previous analysis to demonstrate the effect of element size in relation to the accuracy of the obtained solutions.

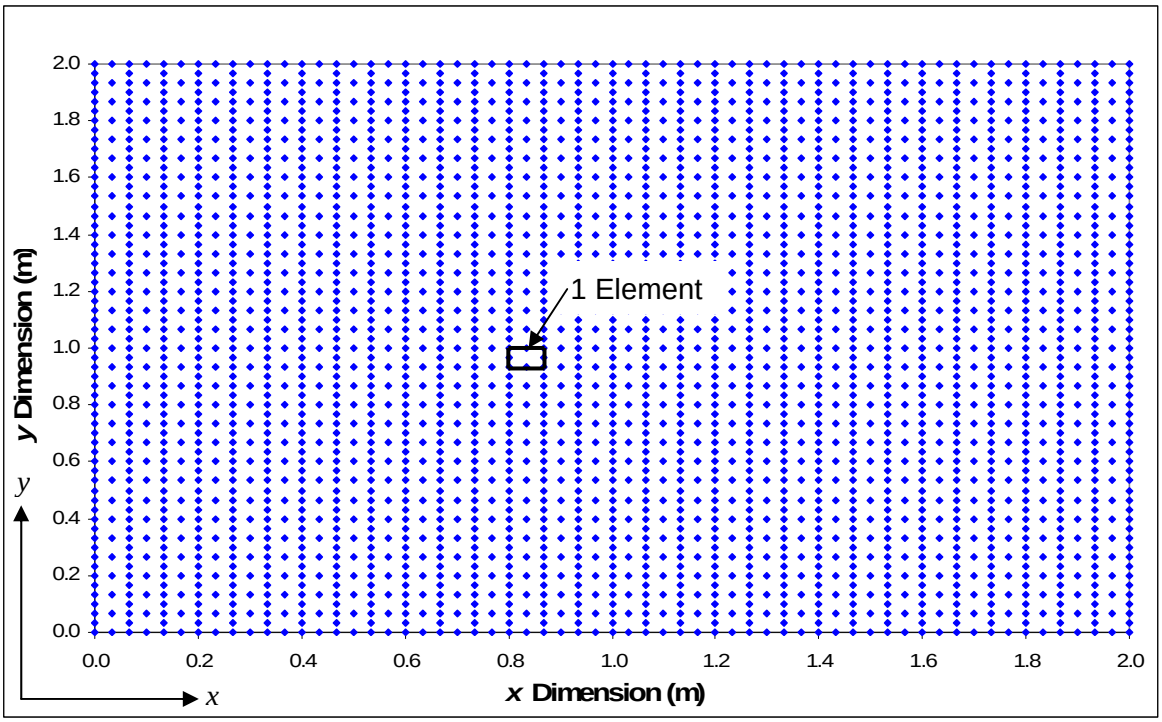


Figure 4.32 Sensitivity analysis 4 relative to the Katse verification

- Element size in the *x* direction = 0.0667 m
- Element size in the *y* direction = 0.0667 m
- Aspect ratio = 0.0667 m ÷ 0.0667 m = 1
- Number of elements in the *x* direction = 30
- Number of elements in the *y* direction = 30
- Total number of elements = 900

Sensitivity analysis - Position 1: At the centre of the rock and concrete interface

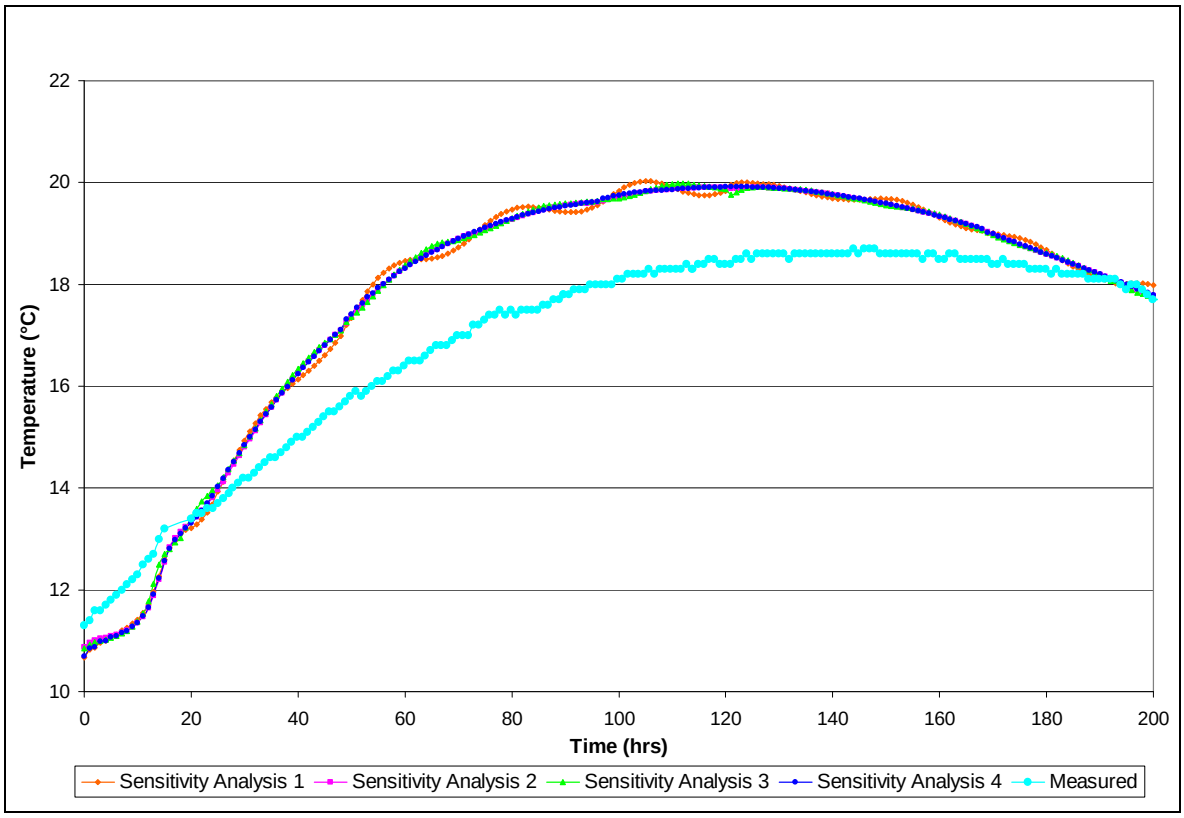


Figure 4.33 Sensitivity analysis at position 1

The temperature variation between each sensitivity analysis plot is minor, except for slight oscillations occurring for sensitivity analysis 1 to sensitivity analysis 3. Sensitivity analysis 1 oscillates more than all the other plots due to an insufficient number of elements covering the bottom surface. Thus, it is important to note, that all boundaries must be specified with a relatively large number of elements or elements not greater than 200 mm.

Sensitivity analysis - Position 2: 250 mm above the bottom probe

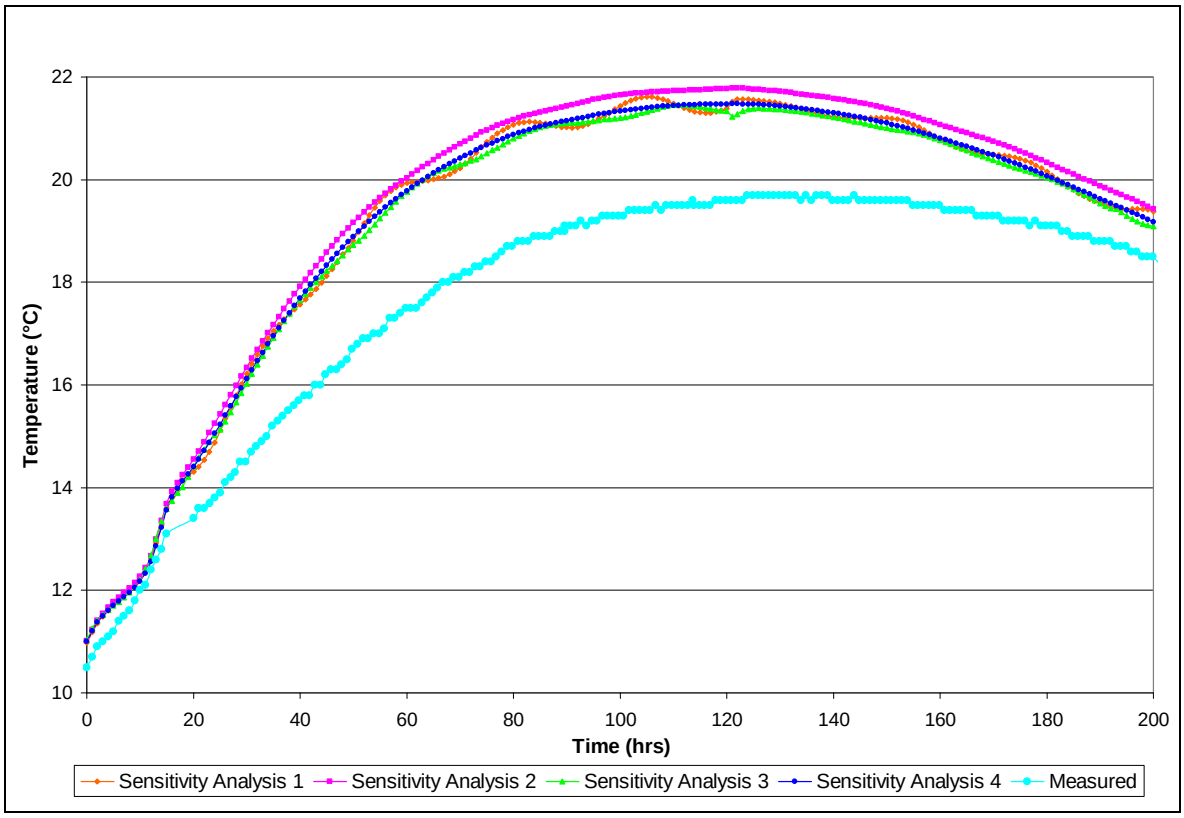


Figure 4.34 Sensitivity analysis at position 2

Following this analysis, it can be deduced that element orientation can alter the relative output of two analyses even if the elements are identical in aspect ratio and size. Referring to sensitivity analysis 1 and 2, the number of elements and aspect ratios are equivalent. However, these analyses will predict alternative temperatures throughout the cross-section due to a variation in the number of nodes in the *x* and *y* directions. In an effort to avoid this inaccuracy, aspect ratios nearest to unity should be adhered to.

Sensitivity analysis - Position 3: 500 mm above the bottom probe

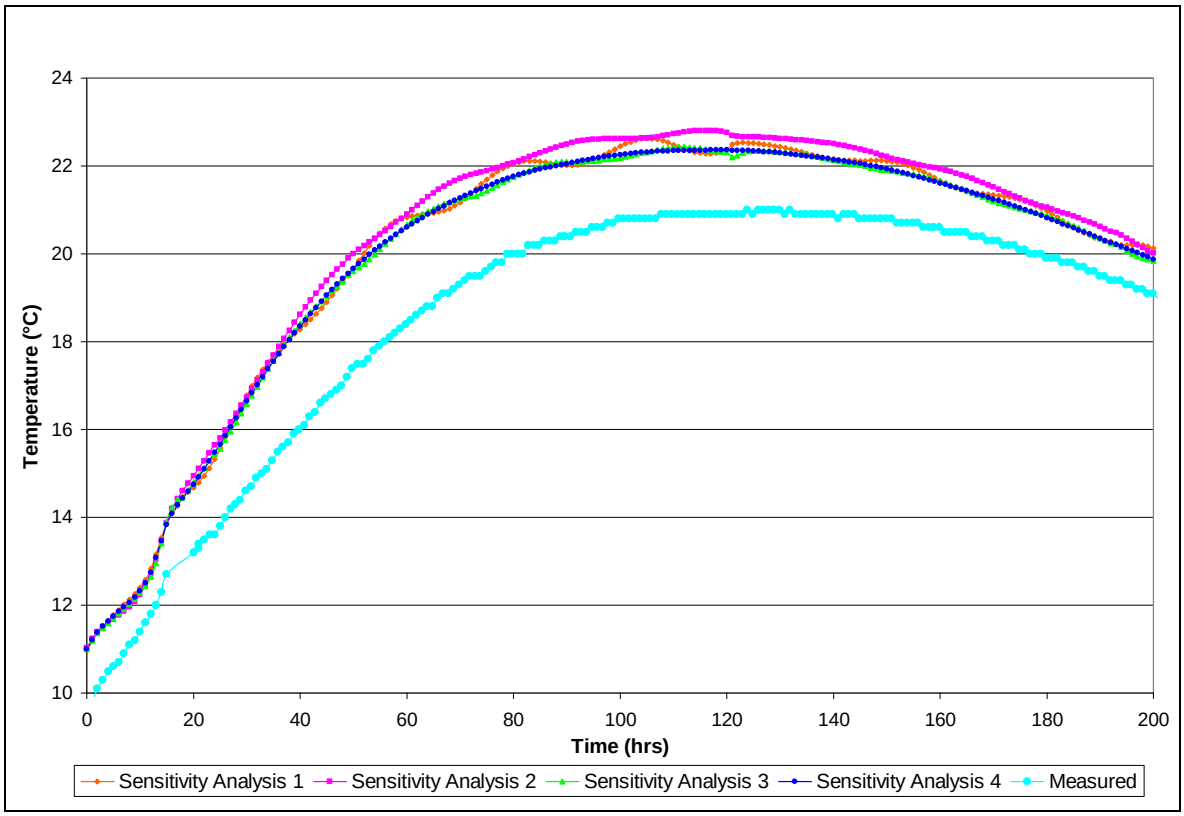


Figure 4.35 Sensitivity analysis at position 3

The preceding discussion for position 2 can be applied to the sensitivity analysis at position 3.

Sensitivity analysis - Position 4: At the centre of the concrete cube

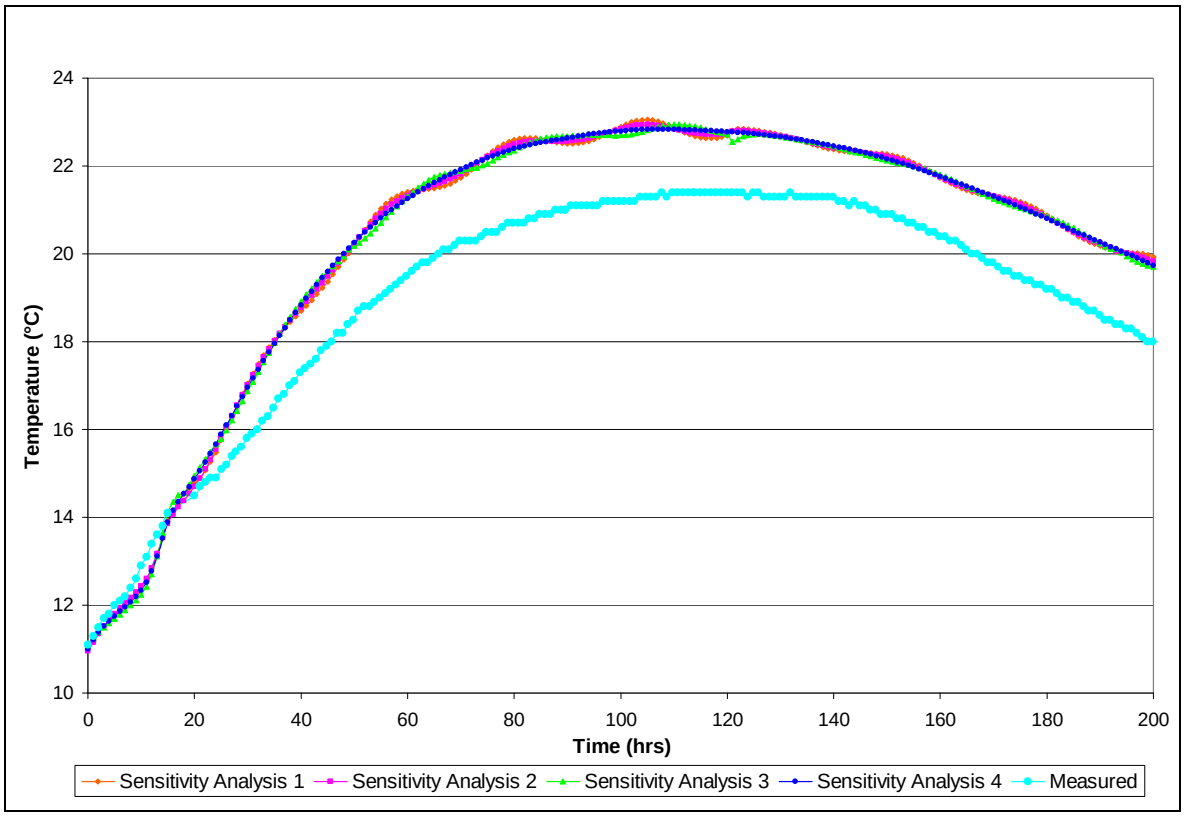


Figure 4.36 Sensitivity analysis at position 4

The likewise discussion for position 2 can be applied to the sensitivity analysis at position 4.

Sensitivity analysis - Position 5: 500 mm north of the central probe

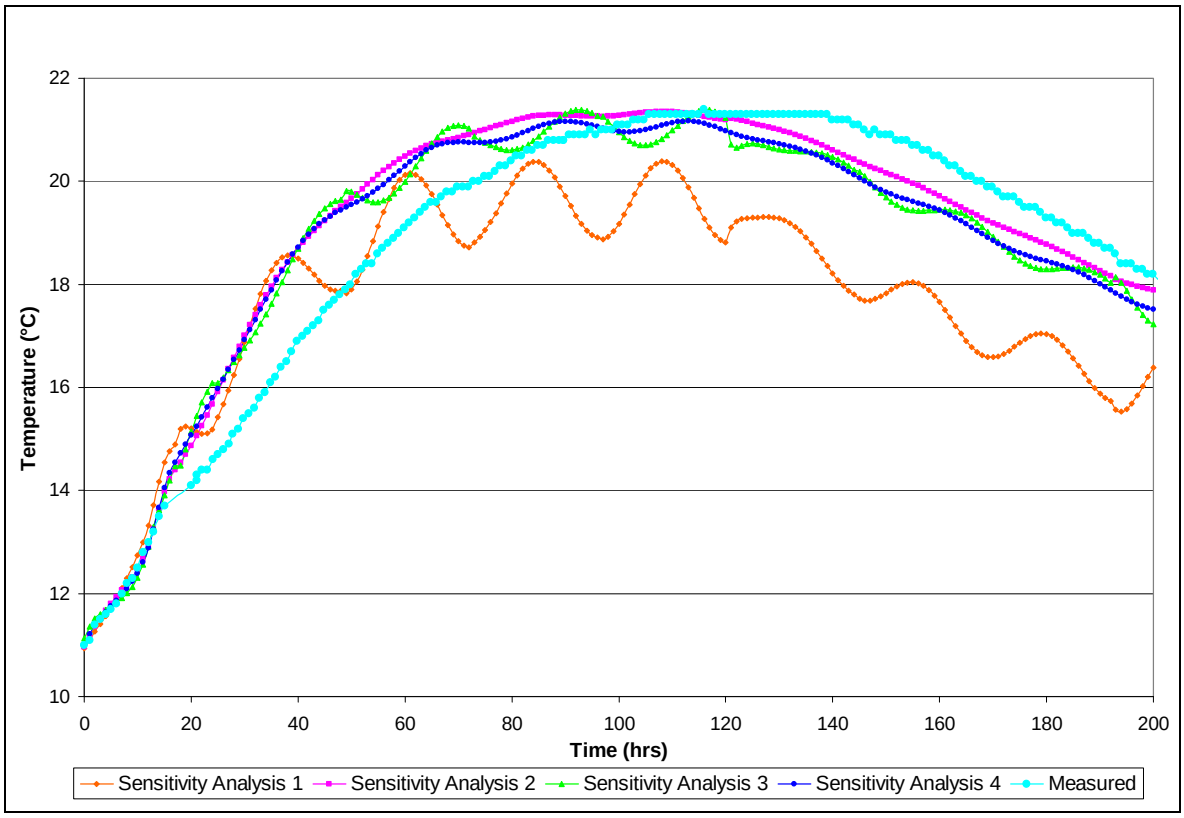


Figure 4.37 Sensitivity analysis at position 5

This analysis demonstrates the effect of element orientation. The graph clearly shows that sensitivity analysis 2 predicts the temperature profile to a greater degree of accuracy than sensitivity analysis 1. An insufficient number of nodes in the *y*-direction, for sensitivity analysis 2, produces the observed error. Another important factor to consider in this analysis is the element aspect ratio. As the aspect ratio approaches unity and the elements become smaller it becomes simpler to obtain a node within the finite element discretisation corresponding to a thermal probe position within the concrete block (i.e. for sensitivity analysis 2 the closest point within the discretised cross-section has coordinates of x = 0.667 m and y = 0.5 m compared to the actual coordinates of x = 0.5 m and y = 0.5 m). Consequently errors are generated which become evident when the results are compared.

Sensitivity analysis - Position 6: 50 mm from the east face level with the central probe

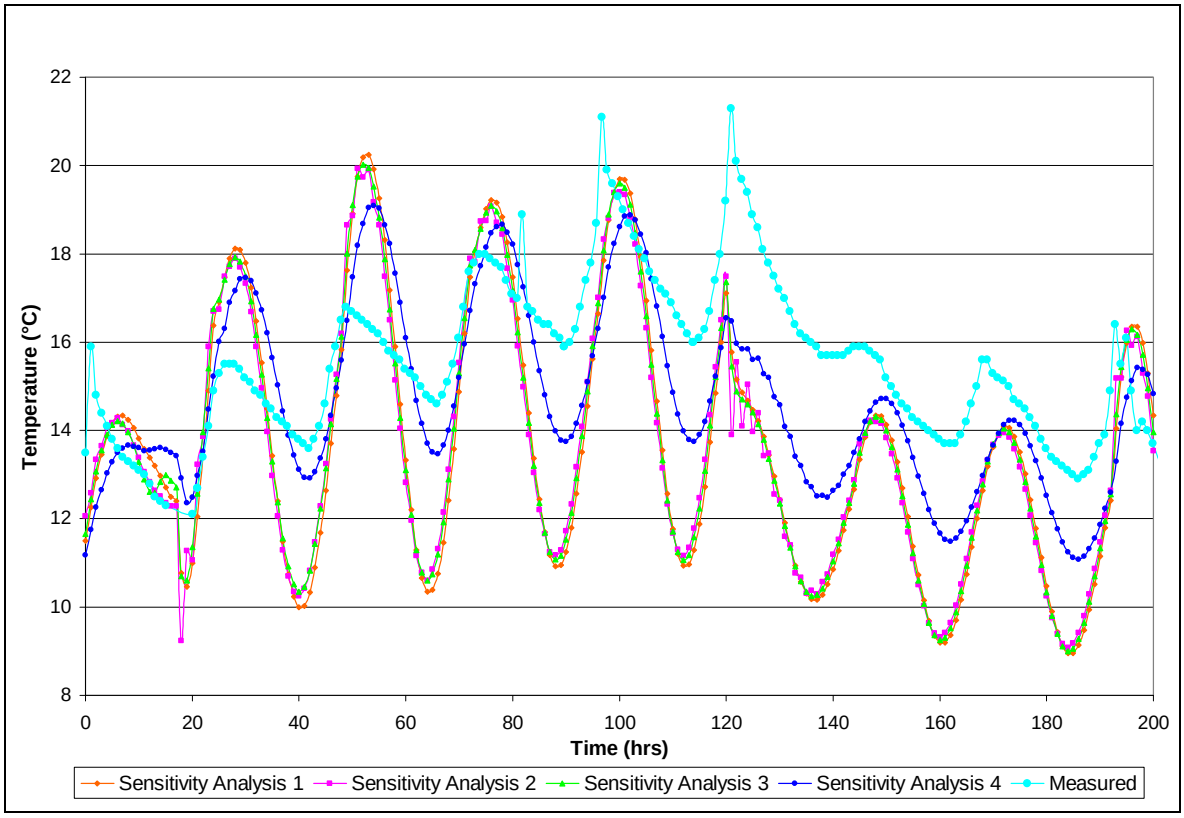


Figure 4.38 Sensitivity analysis at position 6

The role that element size plays is illustrated in the above temperature profile. Sensitivity analyses 1 to 3 included elements that are insufficiently small. As a result, no node exists at a distance of 50 mm from the side face. Sensitivity analysis 4 does have adequately small elements to obtain the predicted temperatures. Subsequently the temperature prediction is more accurate. Expected temperatures for sensitivity analyses 1 to 3 have been taken as surface temperatures. Nonetheless, sensitivity analyses 1 to 3 predict the same temperatures with different aspect ratios and element orientations.

4.6 FEM EXAMPLE OF AN IRREGULAR SHAPED CROSS-SECTION

An important advantage of the finite element numerical model over other numerical methods is that irregular cross-sectional shapes can be represented. An example of an irregular shaped cross-section is shown below to indicate the functionality of the finite element numerical model. A comparison with measured results is not possible due to the unavailability of this type of measured data.

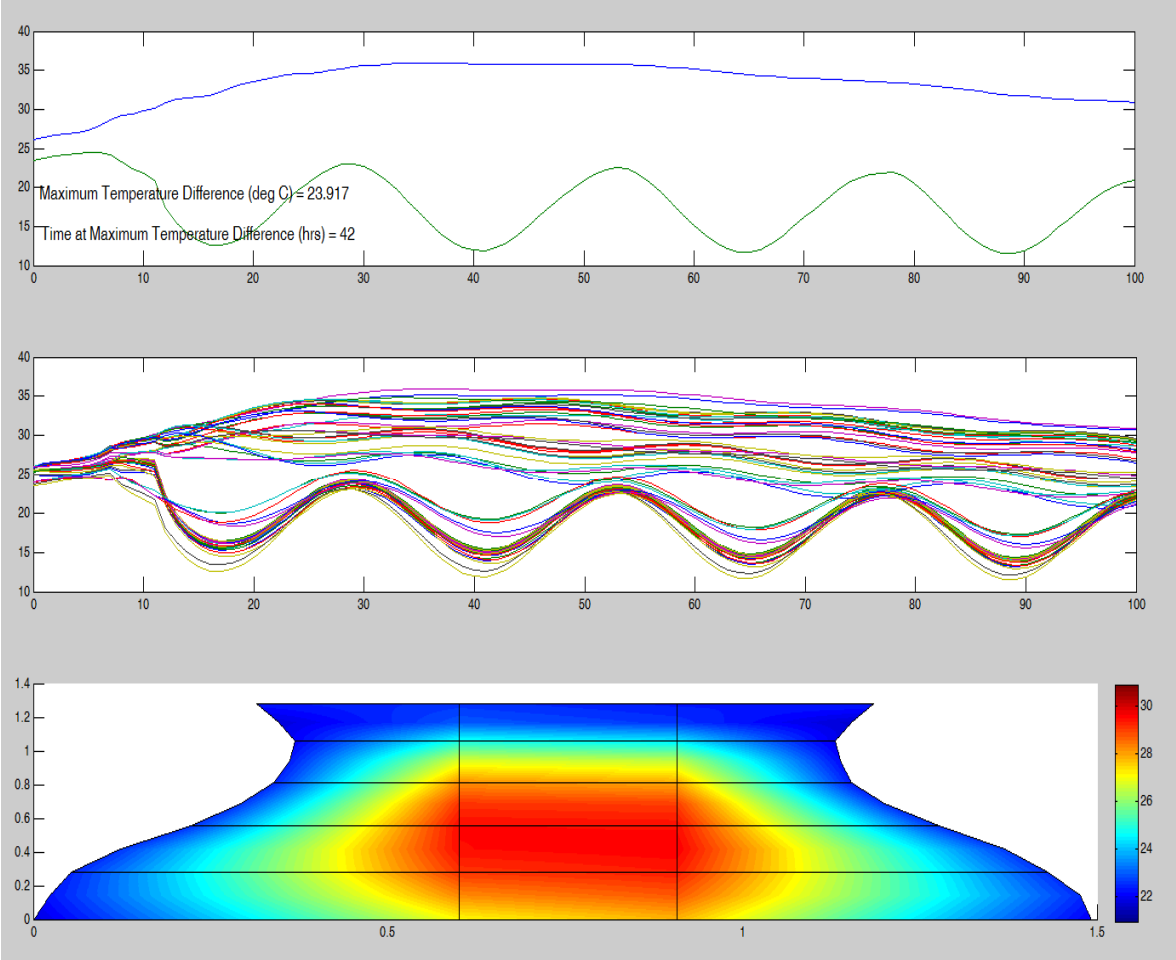


Figure 4.39 FEM Example of an Irregular Shaped Cross-Section

5 CONCLUSION

The finite element numerical model, designed to predict the rise in temperature and distribution of thermal energy in a concrete element, generates satisfactorily accurate estimations and correlates well with experimental results. An absolute maximum difference of 1.7ºC is achieved between the maximum predicted and measured temperatures, excluding all temperature profiles on the concrete elements' surfaces. The maximum temperature gradient attained within the concrete element and the time at which it occurs are also predicted to a high degree of accuracy. The maximum temperature gradient is predicted to within 4.0ºC and the time at which it occurs to within 4 hours. An accurate estimation of the potential stress and therefore the likelihood of cracking within the concrete element can be achieved with the well-defined finite element numerical model's temperature gradients.

Nevertheless, the finite difference numerical model is more user-friendly and operates on the generally available Microsoft Excel software package. Research has shown that the finite difference method produces satisfactory predictions under standard laboratory conditions. Thus, for controlled environments, this numerical model is preferred to the more complex finite element numerical model written in Matlab. However, when boundary conditions are intricate, the finite element numerical model generates more accurate temperature predictions when compared with the finite difference numerical model.

Enhancements of the finite element numerical model are possible, which could improve upon or entirely eliminate the shortcomings referred to in Chapter 4. The major enhancements that would need to be incorporated are:

- Moisture content versus thermal conductivity relationship
- Statistical model to predict boundary conditions and associated modifications to the finite element model

- Development of an improved modelling technique for the rock and concrete interface

The accuracy of the finite element numerical model's temperature predictions can also be improved if the aspect ratio of the elements is equal to unity and a finer mesh is generated. With an increased number of elements, refinement of the numerical model's mesh density will require an increase in the computer time required to solve the prediction model. Therefore, expansion of the concrete body increases the number of elements, resulting in extended computer runtime.

The research conducted in this project can be further developed into a practical application to industry through the creation of a global model that allows for the prediction and prevention of cracking by accounting for temperature differentials attained in the concrete mass. This would require the construction of a simulation that predicts a gain in stiffness as a function of heat liberated for a specific concrete mixture and relates a change in stiffness to a thermal strain model. Input data for such a stiffness model could be implemented empirically to avoid the use of broad assumptions.

The principal use of the finite element numerical model will be limited in practice to complex or water-retaining engineering structures where thermal cracking could lead to structural failure or loss of integrity. The application would be limited because of the time-consuming adiabatic calorimeter tests necessary for each concrete mixture design.

It is highly likely that the average value of multiple calorimeter tests would be required for all concrete mixtures. The limitation of this process is that each calorimeter test necessarily involves a time-period of approximately five days. Implementation of this model, particularly for large-scale projects, will necessitate the assembly of a purpose-built laboratory with adiabatic calorimeters and the employment of skilled laboratory technicians.

6 RECOMMENDATIONS

It is recommended that a cracking potential model be developed which can be applied to complex structures with compound boundary conditions. Potential future improvements will now be expanded upon.

Following the Mathematics in Industry Study Group South Africa (MISGSA) held at the University of the Witwatersrand from the 19th to the 23rd of January 2004, Charpin et al., 2004a, 2004b and Fowkes et al., 2004 proposed theoretical calculations which have become relevant and are briefly discussed in the subsequent sub-sections. Researchers intending to develop the current finite element model should refer to the relevant literature. Each of the discussed topics below is well suited for future postgraduate research.

6.1 PIPED WATER COOLING IN CONCRETE DAMS

Due to the low thermal conductivity of concrete, the rate of heat transferred to the surrounding environment occurs slowly. Casting of very large concrete elements results in a relatively small amount of heat being lost to the surroundings. A network of pipes is often cast into large concrete elements through which chilled water is pumped. This method is employed to extract some of the heat of hydration in order to reduce temperatures more quickly and minimise the temperature differential.

Explicit expressions for a simple cylindrical model have been produced for the maximum concrete temperature as a function of the dependent variables; flow rate through the piping, pipe length and inlet temperature et cetera. Expressions have also been determined for the pipe length and separation distance required to restrict the temperature rise in the concrete elements to a defined level. It has been

proposed that a financial model be incorporated to obtain an optimized solution that minimises cost.

6.2 MATURITY EFFECTS IN CONCRETE DAMS

In water-retaining, mass concrete structures such as dam walls, an allowance for shrinkage movement in the design is necessary to prevent excessive leakage. Thus, sequential concrete blocks are cast after sufficient stiffness has been attained in the preceding layer. Ballim and Graham (2003) observed that the heat of hydration is transferred across the contact surface from the upper block resulting in an increase in the heat of hydration within the first block, as shown in Figure 6.1. This is due to the dependence of hydration rate on temperature.

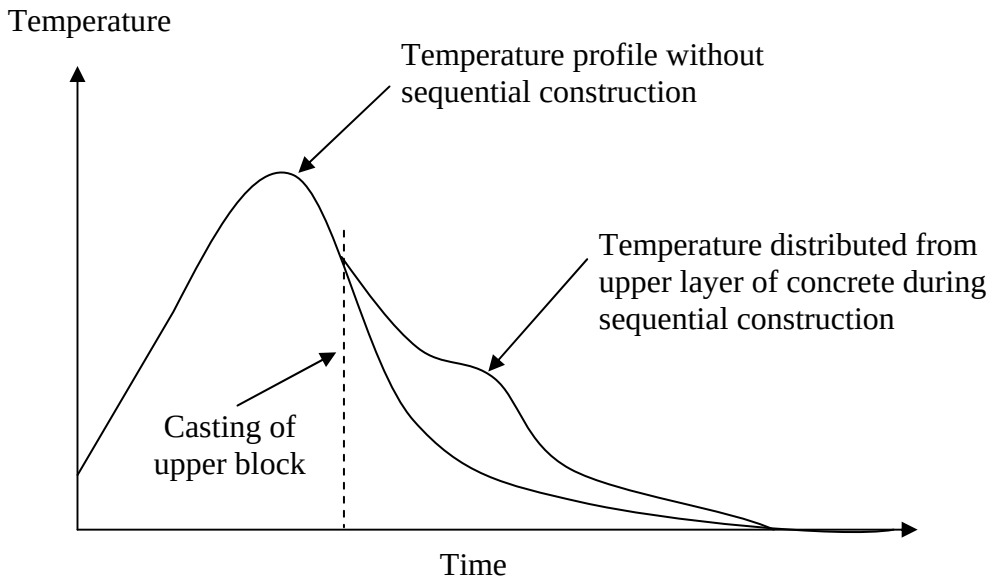


Figure 6.1 Temperature profile for the lower block of concrete

The proposed analysis to predict the above phenomenon is still in investigation phases, however a starting point has been put in place for future research work.

6.3 Modelling Surface Heat Exchanges from a Concrete Block into the Environment

Modelling the impact of environmental conditions on early age concrete provides an indication of the durability and strength of a concrete structure. These factors are directly related to thermal cracking.

The cooling conditions proposed by MISGSA are suitable replacements for the current modelling techniques. However, further development and testing of these conditions are required. It has been proposed that variations in the convective heat transfer coefficient (h) with respect to wind speed be investigated experimentally.

6.4 Industry Standard Database Generation

It is highly recommended that a database containing common heat rate curves be generated for varying binder types and aggregate classification and grade. A comprehensive database will assist engineers in their selection of concrete mixture compositions.

7 REFERENCES

Addis, B.J. (ed.) (1986) Fulton's Concrete Technology, 6th Revised Edition, Portland Cement Institute, Midrand, South Africa

Ballim, Y. (2004a) A numerical model and associated calorimeter for predicting temperature profiles in mass concrete, Cement and Concrete Composites, Elsevier, vol. 26

Ballim, Y. (2004b) Temperature rise in mass concrete elements – Model development and experimental verification using concrete at Katse dam. Journal of the SAICE, vol. 46, No. 1

Ballim, Y. and Graham, P.C. (2003) A maturity approach to the rate of heat evolution in concrete, Magazine of Concrete Research, Thomas Telford, vol. 55

Ballim, Y. and Graham, P.C. (2004) Early-age heat evolution of clinker cements in relation to microstructure and composition: implications for temperature development in large concrete elements, Cement and Concrete Composites, Elsevier, vol. 26

Ballim, Y. and Graham, P.C. (2005) A numerical model for predicting early age time-dependant profiles in large concrete structures, Research Monograph, No. 8, Department of Civil Engineering, University of the Witwatersrand, South Africa

Ballim, Y. and Graham, P.C. (2009) The effects of supplementary cementing materials in modifying the heat of hydration of concrete, RILEM Materials and Structures, vol. 42

Bentz, D.P. and Garboczi, E.J. (1993) Digital-Image Base Computer Modeling of Cement-Based Materials, In: Frost, J.D. et al. (ed.) Digital Image Processing: Techniques and Application in Civil Engineering, ASCE, New York, USA

Charpin, J.P.F. and Myers, T.G. and Fitt, A.D. and Ballim, Y. and Patini, A. (2004a) Modeling surface heat exchanges from a concrete block into the environment, Proc. of

the Mathematics in Industry Study Group, University of the Witwatersrand, Johannesburg, RSA

Charpin, J.P.F. and Myers, T.G. and Fitt, A.D. and Fowkes, N.D. and Ballim, Y. and Patini, A. (2004b) Piped water cooling of concrete dams, Proc. of the Mathematics in Industry Study Group, University of the Witwatersrand, Johannesburg, RSA

Clough, R.W. (1960) The finite element method in plane stress analysis, Proceedings of the 2nd ASCE conference on electronic computation, Pittsburgh, PA

Cope, R.J. and Clark, L.A. (1984) Concrete slabs: Analysis and Design. Elsevier Applied Science

Courant, R. (1943) Variational methods for the solutions of problems of equilibrium and vibrations, Bulletin of the American Mathematical Society, vol. 49

Emborg, M. and Bernander, S. (1994) Avoidance of Early Age Thermal Cracking in Concrete Structures – Predesign, Measure, Follow-up, In: Springenschmid, R (ed.) Thermal Cracking in Concrete at Early Ages, E&FN Spon, London, UK

Fowkes, N.D. and Mambili Mamboundou, H. and Makinde, O.D. and Ballim, Y. and Patini, A. (2004) Maturity effects in concrete dams, Proc. of the Mathematics in Industry Study Group, University of the Witwatersrand, Johannesburg, RSA

Ghosh, S.N. (1991) Cement and concrete science technology, Thomas Telford, London, UK

Gibbon, G.J. and Ballim, Y. and Grieve, G.R.H. (1997) A low-cost, computer-controlled adiabatic calorimeter for determining the heat of hydration of concrete, ASTM Journal of Testing and Evaluation, vol. 25, no. 2

Greensmith, C.G. (2005) The effects of cement extenders and water to binder ratio on the heat evolution characteristics of concrete, MSc Research Report, University of the Witwatersrand, Johannesburg, RSA

Harrison, T.A. (1981) Early-Age Thermal Crack Control in Concrete, Construction Industry Research and Information Association, Report No 91, London, UK

Holman, J.P. (1986) Heat transfer, 6th Edition, McGraw Hill Inc., New York, USA

Huebner, K.H. and Thornton, E.A. and Byrom, T.G. (1995) The Finite Element Method for Engineers, 3rd Edition, John Wiley and Sons, New York, USA

Isgor, O.B. and Razaqpur, A.G. (2004) Finite element modeling of coupled heat transfer, moisture transport and carbonation processes in concrete structures, Cement and Concrete Composites, Elsevier, vol. 26

Koenders, E.A.B and Van Breugel, K. (1994) Numerical and Experimental adiabatic Hydration Curve Determination, In: Springenschmid, R. (ed.) Thermal Cracking in Concrete at Early Ages, E&FN Spon, London, UK

Kumar, B. (1996) Information processing in civil and structural engineering design, Civil-Comp Press, Stirling, UK

Lewis, R.W. and Morgan, K. and Thomas, H.R. and Seetharamu, K.N. (1996) The Finite Element Method in Heat Transfer Analysis, John Wiley and Sons, West Sussex, UK

Maekawa, K. and Chaube, R. and Kishi, T. (1999) Modeling of Concrete Performance – Hydration, Microstructure formation and Mass Transport, E&FN Spon, London, UK

Maruyama, I. and Matsushita, T. and Noguchi, T. (2007) Kinetics and Phase Composition Model for Portland Cement Hydration, In: Sato, R. (ed) Proc. Of Int. Sem. On Durability and Lifecycle Evaluation of Concrete Structures, Hagashi, Hiroshima, Japan

Morabitu, P. (1998) Methods to Determine the Heat of Hydration of Concrete, In: Springenschmid, R. (ed.) Prevention of Thermal Cracking in Concrete at Early Ages, E&FN Spon, London, UK

Morabitu, P. and Barberis, F. (1993) Measurement of Adiabatic Temperature Rise in Concrete, In: Dhir , R.K. and Jones, M.R. Proceedings Concrete 2000, E&FN Spon, London, UK

Neville, A.M. (1981) Properties of Concrete, 3rd Edition, Pitman Publishing Ltd., London, UK.

Rogers, G.F.C. and Mayhew, Y.R. (1992) Engineering Thermodynamics – Work and Heat Transfer, 4th Edition, Longman Singapore Publishers, Singapore

Scanlon, J.M. and McDonald, J.E. (1994) Thermal Properties, In: Kleiger, P. and Lamond, J.F. (eds.) Significance of tests and properties of concrete and concrete-making materials, ASTM-STP 169C. American Society for Testing and Materials, Philadelphia, USA

Suzuki, Y. and Tsuji, Y. and Maekawa, K. and Okamura, H. (1990) Quantification of Heat Evolution During Hydration Process of Cement in Concrete, Proc. Of JSCE, JSCE

Taylor, P.C. and Addis, B.J. (1994) Concrete at Early Ages, In: Addis, B.J. (ed.) Fulton's Concrete Technology, 6th Revised Edition, Portland Cement Institute, Midrand, South Africa

Tetmayer, T. (1883) Deutsche Topfer-und Ziegler-Ztg., 234

Turner, M.J. and Clough, R.W. and Martin, H.C. and Topp, L.J. (1956) Stiffness and deflection analysis of complex structures, Journal of Aeronautical Sciences, vol. 23, no. 9

Ugural, A.C. (1999) Stresses in Plates and Shells, 2nd edition, McGraw Hill, Boston, USA

Wang, C.H. and Dilger, W.H. (1994) Prediction of Temperature Distribution in Hardening Concrete, In: Springenschmid, R (ed.) Thermal Cracking in Concrete at Early Ages, E&FN Spon, London, UK

White, G.R. (1977) Concrete technology, 3rd Edition, Van Nostrand Reinhold Co., New York, USA

Van Breugel, K. (1998) Prediction of Temperature Development in Hardening Concrete, Rilem Report 15, Rilem Technical Committee 119, E&FN Spon, London, UK

Zienkiewicz, O.C. and Cheung, Y.K. (1967) The finite element method in structural and continuum mechanics, McGraw Hill Book Co., London, UK

8 APPENDIX A:

8.1 GLOBAL FEM MATLAB CODE

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                  %%
%%          %-----------------------------INPUT----------------------------%          %%
%%                                                  %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

NoOfElements = input('Total number of elements  = ');
NoOfNodes = input('Total number of nodes  = ');
NoOfElementsYDirection = input('Total number of elements in the y-Direction  = ');
NoOfElementsXDirection = input('Total number of elements in the x-Direction  = ');
InitialTemp = input('Initial concrete temperature - deg C  = ');
CastTime = input('Time of day when concrete is cast - hrs  = ');
k = input('Thermal conductivity of concrete - W/m.K  = ');
rho = input('Concrete density - kg/m3 =  ');
cp = input('Concrete specific heat - J/kg.K  = ');
Ft = input('Formwork removal time - hrs  = ');
hE = input('Convective heat transfer coefficient for exposed concrete surface - W/K.m2  = ');
hC = input('Convective heat transfer coefficient for surfaces covered with formwork-W/K.m2 =');
kr = input('Thermal conductivity of rock - W/K.m2  = ');
Sigma = input('Stefan Boltzman constant - W/K4.m2  = ');
Emissivity = input('Emissivity of grey concrete surface  = ');
tm = input('Time at which the minimum overnight temperature occurs - hrs  = ');
bin = input('Binder content - kg/m3  = ');
E = input('Apparent activation energy - kJ/mol  = ');
R = input('Universal gas constant - kJ/mol.K  = ');
TimeIncrement = input('Time increment - hrs  = ');
FinalTime = input('Time duration - hrs  = ');

A = 2;
CDMHolder = zeros(NoOfNodes,1);
TimeStepFinal = zeros((FinalTime/TimeIncrement)+1,1);
TimeStep = zeros((FinalTime/TimeIncrement)+1,1);
Tfinal = zeros((FinalTime/TimeIncrement)+1,NoOfNodes);
Mx = zeros(FinalTime,1);
Mn = zeros(FinalTime,1);

%_______________________________________________________________________
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                    %%
%%       %-------------C----------C----------C----------C------------%        %%
%%                                                    %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global e;
   e = 1;
   C = rho.*cp.*(1/3600).*[dblquad(@CapMatrix1,-1,1,-1,1), dblquad(@CapMatrix2,-1,1,-1,1),
         dblquad(@CapMatrix3,-1,1,-1,1),...
         dblquad(@CapMatrix4,-1,1,-1,1), dblquad(@CapMatrix5,-1,1,-1,1),
         dblquad(@CapMatrix6,-1,1,-1,1),...
         dblquad(@CapMatrix7,-1,1,-1,1), dblquad(@CapMatrix8,-1,1,-1,1);
         dblquad(@CapMatrix9,-1,1,-1,1),...
         dblquad(@CapMatrix10,-1,1,-1,1), dblquad(@CapMatrix11,-1,1,-1,1),
         dblquad(@CapMatrix12,-1,1,-1,1),...
         dblquad(@CapMatrix13,-1,1,-1,1), dblquad(@CapMatrix14,-1,1,-1,1),
         dblquad(@CapMatrix15,-1,1,-1,1),...
         dblquad(@CapMatrix16,-1,1,-1,1); dblquad(@CapMatrix17,-1,1,-1,1),
         dblquad(@CapMatrix18,-1,1,-1,1),...
         dblquad(@CapMatrix19,-1,1,-1,1), dblquad(@CapMatrix20,-1,1,-1,1),
         dblquad(@CapMatrix21,-1,1,-1,1),...
         dblquad(@CapMatrix22,-1,1,-1,1), dblquad(@CapMatrix23,-1,1,-1,1),
         dblquad(@CapMatrix24,-1,1,-1,1);...
         dblquad(@CapMatrix25,-1,1,-1,1), dblquad(@CapMatrix26,-1,1,-1,1),
         dblquad(@CapMatrix27,-1,1,-1,1),...
         dblquad(@CapMatrix28,-1,1,-1,1), dblquad(@CapMatrix29,-1,1,-1,1),
         dblquad(@CapMatrix30,-1,1,-1,1),...
         dblquad(@CapMatrix31,-1,1,-1,1), dblquad(@CapMatrix32,-1,1,-1,1);
         dblquad(@CapMatrix33,-1,1,-1,1),...
         dblquad(@CapMatrix34,-1,1,-1,1), dblquad(@CapMatrix35,-1,1,-1,1),
         dblquad(@CapMatrix36,-1,1,-1,1),...
         dblquad(@CapMatrix37,-1,1,-1,1), dblquad(@CapMatrix38,-1,1,-1,1),
         dblquad(@CapMatrix39,-1,1,-1,1),...
         dblquad(@CapMatrix40,-1,1,-1,1); dblquad(@CapMatrix41,-1,1,-1,1),
         dblquad(@CapMatrix42,-1,1,-1,1),...
         dblquad(@CapMatrix43,-1,1,-1,1), dblquad(@CapMatrix44,-1,1,-1,1),
         dblquad(@CapMatrix45,-1,1,-1,1),...
         dblquad(@CapMatrix46,-1,1,-1,1), dblquad(@CapMatrix47,-1,1,-1,1),
         dblquad(@CapMatrix48,-1,1,-1,1);...
         dblquad(@CapMatrix49,-1,1,-1,1), dblquad(@CapMatrix50,-1,1,-1,1),
         dblquad(@CapMatrix51,-1,1,-1,1),...
         dblquad(@CapMatrix52,-1,1,-1,1), dblquad(@CapMatrix53,-1,1,-1,1),
         dblquad(@CapMatrix54,-1,1,-1,1),...
         dblquad(@CapMatrix55,-1,1,-1,1), dblquad(@CapMatrix56,-1,1,-1,1);
         dblquad(@CapMatrix57,-1,1,-1,1),...
```

```
          dblquad(@CapMatrix58,-1,1,-1,1), dblquad(@CapMatrix59,-1,1,-1,1),
          dblquad(@CapMatrix60,-1,1,-1,1),...
          dblquad(@CapMatrix61,-1,1,-1,1), dblquad(@CapMatrix62,-1,1,-1,1),
          dblquad(@CapMatrix63,-1,1,-1,1),...
          dblquad(@CapMatrix64,-1,1,-1,1)];

load NodesXElements.dat
NodesXElementsTranspose = transpose(NodesXElements);
AssembledCapacitanceMatrix = zeros(NoOfNodes,NoOfNodes);

for e = 1:NoOfElements
   for i = 1:8                                          %(8 Nodes per element)
      for j = 1:8                                       %(8 Nodes per element)
      rem = i;                                          %(rem = row element matrix)
      cem = j;                                          %(cem = column element matrix)
      ram = NodesXElementsTranspose(i,e);               %(ram = row assembled matrix)
      cam = NodesXElementsTranspose(j,e);               %(cam = column assembled matrix)
      ElementCapacitanceMatrix = C;                     %(3D Capacitance Matrix)
      AssembledCapacitanceMatrix(ram,cam) = AssembledCapacitanceMatrix(ram,cam) +
         ElementCapacitanceMatrix(rem,cem);
   end;
end;
end;
CAP = AssembledCapacitanceMatrix;



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                          %%
%%            %-----------Kc----------Kc----------Kc----------Kc----------%            %%
%%                                                                          %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global ek;
   ek = 1;
   Kc  =   [dblquad(@CondMatrix1,-1,1,-1,1), dblquad(@CondMatrix2,-1,1,-1,1),
           dblquad(@CondMatrix3,-1,1,-1,1),...
           dblquad(@CondMatrix4,-1,1,-1,1), dblquad(@CondMatrix5,-1,1,-1,1),
           dblquad(@CondMatrix6,-1,1,-1,1),...
           dblquad(@CondMatrix7,-1,1,-1,1), dblquad(@CondMatrix8,-1,1,-1,1);
           dblquad(@CondMatrix9,-1,1,-1,1),...
           dblquad(@CondMatrix10,-1,1,-1,1), dblquad(@CondMatrix11,-1,1,-1,1),
           dblquad(@CondMatrix12,-1,1,-1,1),...
           dblquad(@CondMatrix13,-1,1,-1,1), dblquad(@CondMatrix14,-1,1,-1,1),
           dblquad(@CondMatrix15,-1,1,-1,1),...
           dblquad(@CondMatrix16,-1,1,-1,1); dblquad(@CondMatrix17,-1,1,-1,1),
```

```
        dblquad(@CondMatrix18,-1,1,-1,1),...
        dblquad(@CondMatrix19,-1,1,-1,1), dblquad(@CondMatrix20,-1,1,-1,1),
        dblquad(@CondMatrix21,-1,1,-1,1),...
        dblquad(@CondMatrix22,-1,1,-1,1), dblquad(@CondMatrix23,-1,1,-1,1),
        dblquad(@CondMatrix24,-1,1,-1,1);...
        dblquad(@CondMatrix25,-1,1,-1,1), dblquad(@CondMatrix26,-1,1,-1,1),
        dblquad(@CondMatrix27,-1,1,-1,1),...
        dblquad(@CondMatrix28,-1,1,-1,1), dblquad(@CondMatrix29,-1,1,-1,1),
        dblquad(@CondMatrix30,-1,1,-1,1),...
        dblquad(@CondMatrix31,-1,1,-1,1), dblquad(@CondMatrix32,-1,1,-1,1);
        dblquad(@CondMatrix33,-1,1,-1,1),...
        dblquad(@CondMatrix34,-1,1,-1,1), dblquad(@CondMatrix35,-1,1,-1,1),
        dblquad(@CondMatrix36,-1,1,-1,1),...
        dblquad(@CondMatrix37,-1,1,-1,1), dblquad(@CondMatrix38,-1,1,-1,1),
        dblquad(@CondMatrix39,-1,1,-1,1),...
        dblquad(@CondMatrix40,-1,1,-1,1); dblquad(@CondMatrix41,-1,1,-1,1),
        dblquad(@CondMatrix42,-1,1,-1,1),...
        dblquad(@CondMatrix43,-1,1,-1,1), dblquad(@CondMatrix44,-1,1,-1,1),
        dblquad(@CondMatrix45,-1,1,-1,1),...
        dblquad(@CondMatrix46,-1,1,-1,1), dblquad(@CondMatrix47,-1,1,-1,1),
        dblquad(@CondMatrix48,-1,1,-1,1);...
        dblquad(@CondMatrix49,-1,1,-1,1), dblquad(@CondMatrix50,-1,1,-1,1),
        dblquad(@CondMatrix51,-1,1,-1,1),...
        dblquad(@CondMatrix52,-1,1,-1,1), dblquad(@CondMatrix53,-1,1,-1,1),
        dblquad(@CondMatrix54,-1,1,-1,1),...
        dblquad(@CondMatrix55,-1,1,-1,1), dblquad(@CondMatrix56,-1,1,-1,1);
        dblquad(@CondMatrix57,-1,1,-1,1),...
        dblquad(@CondMatrix58,-1,1,-1,1), dblquad(@CondMatrix59,-1,1,-1,1),
        dblquad(@CondMatrix60,-1,1,-1,1),...
        dblquad(@CondMatrix61,-1,1,-1,1), dblquad(@CondMatrix62,-1,1,-1,1),
        dblquad(@CondMatrix63,-1,1,-1,1),...
        dblquad(@CondMatrix64,-1,1,-1,1)];

load NodesXElements.dat
NodesXElementsTranspose = transpose(NodesXElements);
AssembledConductanceMatrix = zeros(NoOfNodes,NoOfNodes);

for e = 1:NoOfElements
   for i = 1:8                                          %(8 Nodes per element)
      for j = 1:8                                       %(8 Nodes per element)
      rem = i;                                          %(rem = row element matrix)
      cem = j;                                          %(cem = column element matrix)
      ram = NodesXElementsTranspose(i,e);               %(ram = row assembled matrix)
      cam = NodesXElementsTranspose(j,e);               %(cam = column assembled matrix)
      ElementConductanceMatrix = Kc;                    %(3D Conductance Matrix)
      AssembledConductanceMatrix(ram,cam) = AssembledConductanceMatrix(ram,cam) +
```

```
        ElementConductanceMatrix(rem,cem);
    end;
end;
end;

COND = AssembledConductanceMatrix;


%_____________________________________________________________________________
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                           %%
%%    %-----ODE SOLVER---ODE SOLVER---ODE SOLVER---ODE SOLVER----%          %%
%%                                                                           %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%_____________________________________________________________________________

Theta = 1/2;    %Crank Nicolson

for t = 0:TimeIncrement:FinalTime

    if t == 0
    Tprevious = InitialTemp*ones(NoOfNodes,1);
    else
    Tprevious = T;
    end

load AmbientTemp.dat
TempAmbient = (-sin(((2*pi*((((t/24)-
                floor(t/24))*24+(CastTime))+tm))/24)).*((interp1(AmbientTemp(:,1),AmbientTe
                mp(:,3),(ceil(t/24)*24))-...
                interp1(AmbientTemp(:,1),AmbientTemp(:,2),(ceil(t/24)*24)))/2)+...
                ((interp1(AmbientTemp(:,1),AmbientTemp(:,3),(ceil(t/24)*24))+interp1(Ambient
                Temp(:,1),AmbientTemp(:,2),(ceil(t/24)*24)))/2));


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                           %%
%%              %-------------Rq----------Rq----------Rq----------Rq------------%              %%
%%                                                                           %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global eq;
load Maturity.dat
eq = 1;
```

```
   RQ = bin.*[dblquad(@Heating1,-1,1,-1,1);...
              dblquad(@Heating2,-1,1,-1,1);...
              dblquad(@Heating3,-1,1,-1,1);...
              dblquad(@Heating4,-1,1,-1,1);...
              dblquad(@Heating5,-1,1,-1,1);...
              dblquad(@Heating6,-1,1,-1,1);...
              dblquad(@Heating7,-1,1,-1,1);...
              dblquad(@Heating8,-1,1,-1,1)];

load NodesXElements.dat
NodesXElementsTranspose = transpose(NodesXElements);
AssembledHeatLiberatedMatrix = zeros(NoOfNodes,1);

for erq = 1:NoOfElements
   for iq = 1:8                                        %(8 Nodes per element)
      remq = iq;                                       %(rem = row element matrix)
      cemq = 1;                                        %(cem = column element matrix)
      ramq = NodesXElementsTranspose(iq,erq);          %(ram = row assembled matrix)
      camq = 1;                                        %(cam = column assembled matrix)
      ElementHeatLiberatedMatrix = RQ;                 %(3D Heat Liberated Matrix)
      AssembledHeatLiberatedMatrix(ramq,camq) = AssembledHeatLiberatedMatrix(ramq,camq)
         + ElementHeatLiberatedMatrix(remq,cemq);
   end;
end;

if A == 2
   DeltaM(1,1) = 0;
end

DeltaM(A,1) = t;

Deltat = DeltaM(A,1)-DeltaM(A-1,1);

for ete = 1:NoOfNodes
      CumulativeMaturity(ete,1) = (exp((E/R)*((1/293)-(1/(Tprevious(ete,1)+273)))))*Deltat +
         CDMHolder(ete,1);
      MaturityAtNode = (interp1(Maturity(:,1),Maturity(:,2),CumulativeMaturity));
      MaturityChange(ete,1) = (exp((E/R)*((1/293)-(1/(Tprevious(ete,1)+273)))));
end;

CDMHolder = CumulativeMaturity;

A = A + 1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
RQT = (AssembledHeatLiberatedMatrix.*MaturityAtNode).*(MaturityChange);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                              %%
%%           %--------------Kh----------Kh----------Kh----------Kh-------------%           %%
%%                                                                              %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if t < Ft
   h = hC;
else
   h = hE;
end

global esS1;
esS1 = 1;
   KhS1 = h.*[quad(@SurConvS11,-1,1),quad(@SurConvS12,-1,1),quad(@SurConvS13,-1,1);...
              quad(@SurConvS14,-1,1),quad(@SurConvS15,-1,1),quad(@SurConvS16,-1,1);...
              quad(@SurConvS17,-1,1),quad(@SurConvS18,-1,1),quad(@SurConvS19,-1,1)];

load NodesXElementsS1.dat
NodesXElementsS1Transpose = transpose(NodesXElementsS1);
AssembledBoundCond1MatrixS1 = zeros(NoOfNodes,NoOfNodes);

for ekh1 = 1:NoOfElementsYDirection
   for ikh1 = 1:3                                         %(8 Nodes per element)
      for jkh1 = 1:3                                      %(8 Nodes per element)
      remkh1 = ikh1;                                      %(rem = row element matrix)
      cemkh1 = jkh1;                                      %(cem = column element matrix)
      ramkh1 = NodesXElementsS1Transpose(ikh1,ekh1);      %(ram = row assembled matrix)
      camkh1 = NodesXElementsS1Transpose(jkh1,ekh1);      %(cam = column assembled matrix)
      ElementBoundCond1MatrixS1 = KhS1;                   %(3D Boundary Conductance 1 Matrix)
      AssembledBoundCond1MatrixS1(ramkh1,camkh1) =
         AssembledBoundCond1MatrixS1(ramkh1,camkh1) +
         ElementBoundCond1MatrixS1(remkh1,cemkh1);
   end;
end;
end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global esS2;
esS2 = 1;
```

```
 KhS2 = hE.*[quad(@SurConvS21,-1,1),quad(@SurConvS22,-1,1),quad(@SurConvS23,-1,1);...
             quad(@SurConvS24,-1,1),quad(@SurConvS25,-1,1),quad(@SurConvS26,-1,1);...
             quad(@SurConvS27,-1,1),quad(@SurConvS28,-1,1),quad(@SurConvS29,-1,1)];

load NodesXElementsS2.dat
NodesXElementsS2Transpose = transpose(NodesXElementsS2);
AssembledBoundCond1MatrixS2 = zeros(NoOfNodes,NoOfNodes);

for ekh2 = 1:NoOfElementsXDirection
   for ikh2 = 1:3                                  %(8 Nodes per element)
      for jkh2 = 1:3                               %(8 Nodes per element)
      remkh2 = ikh2;                               %(rem = row element matrix)
      cemkh2 = jkh2;                               %(cem = column element matrix)
      ramkh2 = NodesXElementsS2Transpose(ikh2,ekh2);   %(ram = row assembled matrix)
      camkh2 = NodesXElementsS2Transpose(jkh2,ekh2);   %(cam = column assembled matrix)
      ElementBoundCond1MatrixS2 = KhS2;            %(3D Boundary Conductance 1 Matrix)
      AssembledBoundCond1MatrixS2(ramkh2,camkh2) =
         AssembledBoundCond1MatrixS2(ramkh2,camkh2) +
         ElementBoundCond1MatrixS2(remkh2,cemkh2);
   end;
end;
end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global esS3;
esS3 = 1;
   KhS3 = h.*[quad(@SurConvS31,-1,1),quad(@SurConvS32,-1,1),quad(@SurConvS33,-1,1);...
             quad(@SurConvS34,-1,1),quad(@SurConvS35,-1,1),quad(@SurConvS36,-1,1);...
             quad(@SurConvS37,-1,1),quad(@SurConvS38,-1,1),quad(@SurConvS39,-1,1)];

load NodesXElementsS3.dat
NodesXElementsS3Transpose = transpose(NodesXElementsS3);
AssembledBoundCond1MatrixS3 = zeros(NoOfNodes,NoOfNodes);

for ekh3 = 1:NoOfElementsYDirection
   for ikh3 = 1:3                                  %(8 Nodes per element)
      for jkh3 = 1:3                               %(8 Nodes per element)
      remkh3 = ikh3;                               %(rem = row element matrix)
      cemkh3 = jkh3;                               %(cem = column element matrix)
      ramkh3 = NodesXElementsS3Transpose(ikh3,ekh3);   %(ram = row assembled matrix)
      camkh3 = NodesXElementsS3Transpose(jkh3,ekh3);   %(cam = column assembled matrix)
      ElementBoundCond1MatrixS3 = KhS3;            %(3D Boundary Conductance 1 Matrix)
      AssembledBoundCond1MatrixS3(ramkh3,camkh3) =
         AssembledBoundCond1MatrixS3(ramkh3,camkh3) +
         ElementBoundCond1MatrixS3(remkh3,cemkh3);
```

```
   end;
end;
end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global esS4;
esS4 = 1;
  KhS4 = kr.*[quad(@SurConvS41,-1,1),quad(@SurConvS42,-1,1),quad(@SurConvS43,-1,1);...
              quad(@SurConvS44,-1,1),quad(@SurConvS45,-1,1),quad(@SurConvS46,-1,1);...
              quad(@SurConvS47,-1,1),quad(@SurConvS48,-1,1),quad(@SurConvS49,-1,1)];

load NodesXElementsS4.dat
NodesXElementsS4Transpose = transpose(NodesXElementsS4);
AssembledBoundCond1MatrixS4 = zeros(NoOfNodes,NoOfNodes);

for ekh4 = 1:NoOfElementsXDirection
   for ikh4 = 1:3                                       %(8 Nodes per element)
      for jkh4 = 1:3                                    %(8 Nodes per element)
      remkh4 = ikh4;                                    %(rem = row element matrix)
      cemkh4 = jkh4;                                    %(cem = column element matrix)
      ramkh4 = NodesXElementsS4Transpose(ikh4,ekh4);    %(ram = row assembled matrix)
      camkh4 = NodesXElementsS4Transpose(jkh4,ekh4);    %(cam = column assembled matrix)
      ElementBoundCond1MatrixS4 = KhS4;                %(3D Boundary Conductance 1 Matrix)
      AssembledBoundCond1MatrixS4(ramkh4,camkh4) =
         AssembledBoundCond1MatrixS4(ramkh4,camkh4) +
         ElementBoundCond1MatrixS4(remkh4,cemkh4);
   end;
end;
end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

AssembledBoundCond1MatrixTOTAL = AssembledBoundCond1MatrixS1 +
                         AssembledBoundCond1MatrixS2 + AssembledBoundCond1MatrixS3 + ...
                         AssembledBoundCond1MatrixS4;

KH = AssembledBoundCond1MatrixTOTAL;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                             %%
%%          %-------------Rh----------Rh----------Rh----------Rh------------%          %%
%%                                                                             %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global ecS1;

load AmbientTemp.dat

ecS1 = 1;
   RhS1 = h.*((-sin(((2*pi*((((t/24)-
                            floor(t/24))*24+(CastTime))+tm))/24)).*((interp1(AmbientTemp(:,1),
                            AmbientTemp(:,3),(ceil(t/24)*24))-...
                            interp1(AmbientTemp(:,1),AmbientTemp(:,2),(ceil(t/24)*24)))/2)+...
                            ((interp1(AmbientTemp(:,1),AmbientTemp(:,3),(ceil(t/24)*24))+interp1(
                            AmbientTemp(:,1),AmbientTemp(:,2),(ceil(t/24)*24)))/2))).*...
                            [quad(@BoundConvS11,-1,1);...
                             quad(@BoundConvS12,-1,1);...
                             quad(@BoundConvS13,-1,1)];

load NodesXElementsS1.dat
NodesXElementsS1Transpose = transpose(NodesXElementsS1);
AssembledBoundCond2MatrixS1 = zeros(NoOfNodes,1);

for erh1 = 1:NoOfElementsYDirection
   for irh1 = 1:3                                          %(3 Nodes per element)
      remrh1 = irh1;                                       %(rem = row element matrix)
      cemrh1 = 1;                                          %(cem = column element matrix)
      ramrh1 = NodesXElementsS1Transpose(irh1,erh1);       %(ram = row assembled matrix)
      camrh1 = 1;                                          %(cam = column assembled matrix)
      ElementBoundCond2MatrixS1 = RhS1;                  %(3D Boundary Conductance 2 Matrix)
      AssembledBoundCond2MatrixS1(ramrh1,camrh1) =
         AssembledBoundCond2MatrixS1(ramrh1,camrh1) +
         ElementBoundCond2MatrixS1(remrh1,cemrh1);
   end;
end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global ecS2;

ecS2 = 1;
RhS2 = hE.*((-sin(((2*pi*((((t/24)-
                            floor(t/24))*24+(CastTime))+tm))/24)).*((interp1(AmbientTemp(:,1),
                            AmbientTemp(:,3),(ceil(t/24)*24))-...
```

```
                    interp1(AmbientTemp(:,1),AmbientTemp(:,2),(ceil(t/24)*24)))/2)+...
                    ((interp1(AmbientTemp(:,1),AmbientTemp(:,3),(ceil(t/24)*24))+interp1(
                    AmbientTemp(:,1),AmbientTemp(:,2),(ceil(t/24)*24)))/2))).*...
                    [quad(@BoundConvS21,-1,1);...
                     quad(@BoundConvS22,-1,1);...
                     quad(@BoundConvS23,-1,1)];

load NodesXElementsS2.dat
NodesXElementsS2Transpose = transpose(NodesXElementsS2);
AssembledBoundCond2MatrixS2 = zeros(NoOfNodes,1);

for erh2 = 1:NoOfElementsXDirection
   for irh2 = 1:3                                    %(3 Nodes per element)
      remrh2 = irh2;                                 %(rem = row element matrix)
      cemrh2 = 1;                                    %(cem = column element matrix)
      ramrh2 = NodesXElementsS2Transpose(irh2,erh2); %(ram = row assembled matrix)
      camrh2 = 1;                                    %(cam = column assembled matrix)
      ElementBoundCond2MatrixS2 = RhS2;             %(3D Boundary Conductance 2 Matrix)
      AssembledBoundCond2MatrixS2(ramrh2,camrh2) =
         AssembledBoundCond2MatrixS2(ramrh2,camrh2) +
         ElementBoundCond2MatrixS2(remrh2,cemrh2);
   end;
end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global ecS3;

ecS3 = 1;
   RhS3 = h.*((-sin(((2*pi*((((t/24)-
                    floor(t/24))*24+(CastTime))+tm))/24)).*((interp1(AmbientTemp(:,1),
                    AmbientTemp(:,3),(ceil(t/24)*24))-...
                    interp1(AmbientTemp(:,1),AmbientTemp(:,2),(ceil(t/24)*24)))/2)+...
                    ((interp1(AmbientTemp(:,1),AmbientTemp(:,3),(ceil(t/24)*24))+interp1(
                    AmbientTemp(:,1),AmbientTemp(:,2),(ceil(t/24)*24)))/2))).*...
                    [quad(@BoundConvS31,-1,1);...
                     quad(@BoundConvS32,-1,1);...
                     quad(@BoundConvS33,-1,1)];

load NodesXElementsS3.dat
NodesXElementsS3Transpose = transpose(NodesXElementsS3);
AssembledBoundCond2MatrixS3 = zeros(NoOfNodes,1);

for erh3 = 1:NoOfElementsYDirection
   for irh3 = 1:3                                    %(3 Nodes per element)
      remrh3 = irh3;                                 %(rem = row element matrix)
```

```
        cemrh3 = 1;                                    %(cem = column element matrix)
        ramrh3 = NodesXElementsS3Transpose(irh3,erh3);  %(ram = row assembled matrix)
        camrh3 = 1;                                    %(cam = column assembled matrix)
        ElementBoundCond2MatrixS3 = RhS3;             %(3D Boundary Conductance 2 Matrix)
        AssembledBoundCond2MatrixS3(ramrh3,camrh3) =
           AssembledBoundCond2MatrixS3(ramrh3,camrh3) +
           ElementBoundCond2MatrixS3(remrh3,cemrh3);
    end;
end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global ecS4;

TimePreviousDay = t - 24;
TempRock = interp1(AmbientTemp(:,1),AmbientTemp(:,2),(ceil(TimePreviousDay/24)*24));

ecS4 = 1;
    RhS4 = kr.*(TempRock).*...
             [quad(@BoundConvS41,-1,1);...
              quad(@BoundConvS42,-1,1);...
              quad(@BoundConvS43,-1,1)];

load NodesXElementsS4.dat
NodesXElementsS4Transpose = transpose(NodesXElementsS4);
AssembledBoundCond2MatrixS4 = zeros(NoOfNodes,1);

for erh4 = 1:NoOfElementsXDirection
    for irh4 = 1:3                                     %(3 Nodes per element)
        remrh4 = irh4;                                 %(rem = row element matrix)
        cemrh4 = 1;                                    %(cem = column element matrix)
        ramrh4 = NodesXElementsS4Transpose(irh4,erh4);  %(ram = row assembled matrix)
        camrh4 = 1;                                    %(cam = column assembled matrix)
        ElementBoundCond2MatrixS4 = RhS4;             %(3D Boundary Conductance 2 Matrix)
        AssembledBoundCond2MatrixS4(ramrh4,camrh4) =
           AssembledBoundCond2MatrixS4(ramrh4,camrh4) +
           ElementBoundCond2MatrixS4(remrh4,cemrh4);
    end;
end;

AssembledBoundCond2MatrixS4Rock = AssembledBoundCond2MatrixS4;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

AssembledBoundCond2MatrixTOTAL = (AssembledBoundCond2MatrixS1 +
                     AssembledBoundCond2MatrixS2 + AssembledBoundCond2MatrixS3) + ...
```

```
                    (AssembledBoundCond2MatrixS4Rock);

RH = AssembledBoundCond2MatrixTOTAL;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                          %%
%%              %-------------Rr----------Rr----------Rr----------Rr------------%              %%
%%                                                                          %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global ewS1;

load AmbientTemp.dat
ewS1 = 1;
   RrS1 = Sigma.*Emissivity.*(((-sin(((2*pi*((((t/24)-floor(t/24))*24+(CastTime))+tm))/24)).*...
            ((interp1(AmbientTemp(:,1),AmbientTemp(:,3),(ceil(t/24)*24))-...
            interp1(AmbientTemp(:,1),AmbientTemp(:,2),(ceil(t/24)*24)))/2)+...
            ((interp1(AmbientTemp(:,1),AmbientTemp(:,3),(ceil(t/24)*24))+...
            interp1(AmbientTemp(:,1),AmbientTemp(:,2),(ceil(t/24)*24)))/2)))^4).*...
            [quad(@BoundRadS11,-1,1);...
             quad(@BoundRadS12,-1,1);...
             quad(@BoundRadS13,-1,1)];

load NodesXElementsS1.dat
NodesXElementsS1Transpose = transpose(NodesXElementsS1);
AssembledBoundRad2MatrixS1 = zeros(NoOfNodes,1);

for e = 1:NoOfElementsYDirection
   for i = 1:3                                          %(8 Nodes per element)
      rem = i;                                          %(rem = row element matrix)
      cem = 1;                                          %(cem = column element matrix)
      ram = NodesXElementsS1Transpose(i,e);             %(ram = row assembled matrix)
      cam = 1;                                          %(cam = column assembled matrix)
      ElementBoundRad2MatrixS1 = RrS1;                  %(3D Boundary Radiation 2 Matrix)
      AssembledBoundRad2MatrixS1(ram,cam) = AssembledBoundRad2MatrixS1(ram,cam) +
         ElementBoundRad2MatrixS1(rem,cem);
   end;
end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global ewS2;

ewS2 = 1;
   RrS2 = Sigma.*Emissivity.*(((-sin(((2*pi*((((t/24)-floor(t/24))*24+(CastTime))+tm))/24)).*...
```

```
            ((interp1(AmbientTemp(:,1),AmbientTemp(:,3),(ceil(t/24)*24))-...
            interp1(AmbientTemp(:,1),AmbientTemp(:,2),(ceil(t/24)*24)))/2)+...
            ((interp1(AmbientTemp(:,1),AmbientTemp(:,3),(ceil(t/24)*24))+...
            interp1(AmbientTemp(:,1),AmbientTemp(:,2),(ceil(t/24)*24)))/2)))^4).*...
            [quad(@BoundRadS21,-1,1);...
             quad(@BoundRadS22,-1,1);...
             quad(@BoundRadS23,-1,1)];

load NodesXElementsS2.dat
NodesXElementsS2Transpose = transpose(NodesXElementsS2);
AssembledBoundRad2MatrixS2 = zeros(NoOfNodes,1);

for e = 1:NoOfElementsXDirection
   for i = 1:3                                   %(8 Nodes per element)
      rem = i;                                   %(rem = row element matrix)
      cem = 1;                                   %(cem = column element matrix)
      ram = NodesXElementsS2Transpose(i,e);      %(ram = row assembled matrix)
      cam = 1;                                   %(cam = column assembled matrix)
      ElementBoundRad2MatrixS2 = RrS2;           %(3D Boundary Radiation 2 Matrix)
      AssembledBoundRad2MatrixS2(ram,cam) = AssembledBoundRad2MatrixS2(ram,cam) +
         ElementBoundRad2MatrixS2(rem,cem);
   end;
end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global ewS3;

ewS3 = 1;
   RrS3 = Sigma.*Emissivity.*(((-sin(((2*pi*((((t/24)-floor(t/24))*24+(CastTime))+tm))/24)).*...
            ((interp1(AmbientTemp(:,1),AmbientTemp(:,3),(ceil(t/24)*24))-...
            interp1(AmbientTemp(:,1),AmbientTemp(:,2),(ceil(t/24)*24)))/2)+...
            ((interp1(AmbientTemp(:,1),AmbientTemp(:,3),(ceil(t/24)*24))+...
            interp1(AmbientTemp(:,1),AmbientTemp(:,2),(ceil(t/24)*24)))/2)))^4).*...
            [quad(@BoundRadS31,-1,1);...
             quad(@BoundRadS32,-1,1);...
             quad(@BoundRadS33,-1,1)];

load NodesXElementsS3.dat
NodesXElementsS3Transpose = transpose(NodesXElementsS3);
AssembledBoundRad2MatrixS3 = zeros(NoOfNodes,1);

for e = 1:NoOfElementsYDirection
   for i = 1:3                                   %(8 Nodes per element)
      rem = i;                                   %(rem = row element matrix)
      cem = 1;                                   %(cem = column element matrix)
```

```
        ram = NodesXElementsS3Transpose(i,e);         %(ram = row assembled matrix)
        cam = 1;                                      %(cam = column assembled matrix)
        ElementBoundRad2MatrixS3 = RrS3;              %(3D Boundary Radiation 2 Matrix)
        AssembledBoundRad2MatrixS3(ram,cam) = AssembledBoundRad2MatrixS3(ram,cam) +
          ElementBoundRad2MatrixS3(rem,cem);
    end;
end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global ewS4;

ewS4 = 1;
    RrS4 = Sigma.*(1).*(((-sin(((2*pi*((((t/24)-floor(t/24))*24+(CastTime))+tm))/24)).*...
            ((interp1(AmbientTemp(:,1),AmbientTemp(:,3),(ceil(t/24)*24))-...
            interp1(AmbientTemp(:,1),AmbientTemp(:,2),(ceil(t/24)*24)))/2)+...
            ((interp1(AmbientTemp(:,1),AmbientTemp(:,3),(ceil(t/24)*24))+...
            interp1(AmbientTemp(:,1),AmbientTemp(:,2),(ceil(t/24)*24)))/2)))^4).*...
            [quad(@BoundRadS41,-1,1);...
             quad(@BoundRadS42,-1,1);...
             quad(@BoundRadS43,-1,1)];

load NodesXElementsS4.dat
NodesXElementsS4Transpose = transpose(NodesXElementsS4);
AssembledBoundRad2MatrixS4 = zeros(NoOfNodes,1);

for e = 1:NoOfElementsXDirection
    for i = 1:3                                       %(3 Nodes per element)
        rem = i;                                      %(rem = row element matrix)
        cem = 1;                                      %(cem = column element matrix)
        ram = NodesXElementsS4Transpose(i,e);         %(ram = row assembled matrix)
        cam = 1;                                      %(cam = column assembled matrix)
        ElementBoundRad2MatrixS4 = RrS4;              %(3D Boundary Conductance 2 Matrix)
        AssembledBoundRad2MatrixS4(ram,cam) = AssembledBoundRad2MatrixS4(ram,cam) +
          ElementBoundRad2MatrixS4(rem,cem);
    end;
end;

AssembledBoundRad2MatrixS4Rock = AssembledBoundRad2MatrixS4;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

AssembledBoundRad2MatrixTOTAL = (AssembledBoundRad2MatrixS1 +
                    AssembledBoundRad2MatrixS2 + AssembledBoundRad2MatrixS3) + ...
                    AssembledBoundRad2MatrixS4Rock;
```

```
RR = AssembledBoundRad2MatrixTOTAL;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                      %%
%%              %--------------Rs----------Rs----------Rs----------Rs-------------%              %%
%%                                                                      %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global elS1;

elS1 = 1;
   RsS1 = Sigma.*Emissivity.*[quad(@SurRadS11,-1,1);...
                              quad(@SurRadS12,-1,1);...
                              quad(@SurRadS13,-1,1)];

load NodesXElementsS1.dat
NodesXElementsS1Transpose = transpose(NodesXElementsS1);
AssembledBoundRad1MatrixS1 = zeros(NoOfNodes,1);

for e = 1:NoOfElementsYDirection
   for i = 1:3                                  %(8 Nodes per element)
      rem = i;                                  %(rem = row element matrix)
      cem = 1;                                  %(cem = column element matrix)
      ram = NodesXElementsS1Transpose(i,e);     %(ram = row assembled matrix)
      cam = 1;                                  %(cam = column assembled matrix)
      ElementBoundRad1MatrixS1 = RsS1;          %(3D Boundary Radiation 1 Matrix)
      AssembledBoundRad1MatrixS1(ram,cam) = AssembledBoundRad1MatrixS1(ram,cam) +
         ElementBoundRad1MatrixS1(rem,cem);
  end;
end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global elS2;

elS2 = 1;
   RsS2 = Sigma.*Emissivity.*[quad(@SurRadS21,-1,1);...
                              quad(@SurRadS22,-1,1);...
                              quad(@SurRadS23,-1,1)];

load NodesXElementsS2.dat
NodesXElementsS2Transpose = transpose(NodesXElementsS2);
AssembledBoundRad1MatrixS2 = zeros(NoOfNodes,1);
```

```
for e = 1:NoOfElementsXDirection
   for i = 1:3                                        %(8 Nodes per element)
      rem = i;                                        %(rem = row element matrix)
      cem = 1;                                        %(cem = column element matrix)
      ram = NodesXElementsS2Transpose(i,e);           %(ram = row assembled matrix)
      cam = 1;                                        %(cam = column assembled matrix)
      ElementBoundRad1MatrixS2 = RsS2;                %(3D Boundary Radiation 1 Matrix)
      AssembledBoundRad1MatrixS2(ram,cam) = AssembledBoundRad1MatrixS2(ram,cam) +
         ElementBoundRad1MatrixS2(rem,cem);
   end;
end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global elS3;

elS3 = 1;
   RsS3 = Sigma.*Emissivity.*[quad(@SurRadS31,-1,1);...
                              quad(@SurRadS32,-1,1);...
                              quad(@SurRadS33,-1,1)];

load NodesXElementsS3.dat
NodesXElementsS3Transpose = transpose(NodesXElementsS3);
AssembledBoundRad1MatrixS3 = zeros(NoOfNodes,1);

for e = 1:NoOfElementsYDirection
   for i = 1:3                                        %(8 Nodes per element)
      rem = i;                                        %(rem = row element matrix)
      cem = 1;                                        %(cem = column element matrix)
      ram = NodesXElementsS3Transpose(i,e);           %(ram = row assembled matrix)
      cam = 1;                                        %(cam = column assembled matrix)
      ElementBoundRad1MatrixS3 = RsS3;                %(3D Boundary Radiation 1 Matrix)
      AssembledBoundRad1MatrixS3(ram,cam) = AssembledBoundRad1MatrixS3(ram,cam) +
         ElementBoundRad1MatrixS3(rem,cem);
   end;
end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

AssembledBoundRad1MatrixTOTAL = AssembledBoundRad1MatrixS1 +
                    AssembledBoundRad1MatrixS2 + AssembledBoundRad1MatrixS3;

RS = (AssembledBoundRad1MatrixTOTAL.*(Tprevious.^4));
%_____________________________________________________________________
%_____________________________________________________________________
```

```
Res1 = (((1/TimeIncrement)*CAP)+(((k.*COND)+KH)*Theta));
Res2 = inv(Res1);
Res3 = ((((1/TimeIncrement)*CAP)-(((k.*COND)+KH)*(1-Theta)))*Tprevious) + (RQT + RH +
        RR - RS);
T = Res2*Res3;

Mx(t+1,:) = max(T);
Mn(t+1,:) = min(T);

ram = A-2;
TimeStep(ram,1) = (ram-1)*TimeIncrement;
TimeStepFinal(ram,1) = TimeStepFinal(ram,1)+TimeStep(ram,1);

Tfinal(ram,:) =  transpose(T);

end;


%_____________________________________________________________________________
%                                                                            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                                                          %%
%%    %------------ END ODE SOLVER--------------END ODE SOLVER---------------%    %%
%%                                                                          %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                            %
%_____________________________________________________________________________

load ElementCoord.dat
for eplot = 1:NoOfElements
        PlotCoordsX(:,eplot) = ElementCoord(((eplot*8)-7):(eplot*8),1);
        PlotCoordsY(:,eplot) = ElementCoord(((eplot*8)-7):(eplot*8),2);
end;

load NodesXElements.dat
NodesXElementsTranspose = transpose(NodesXElements);
Tplot = zeros(8,NoOfElements);

for eplotT = 1:NoOfElements
   for i = 1:8
        ram = NodesXElementsTranspose(i,eplotT);
        rem = i;
        Tplot(rem,eplotT) = Tplot(rem,eplotT) + T(ram,1);
   end;
```

```
end;

Md = Mx - Mn;
Maxdiff = max(Md)
MaxTimeT = find(Md == Maxdiff)

subplot(3,1,1); plot(TimeStepFinal,Mx,TimeStepFinal,Mn);
leg1 = text(1,10,['Maximum Temperature Difference (deg C) = ',num2str(Maxdiff)]);
leg2 = text(1,5,['Time at Maximum Temperature Difference (hrs) = ',num2str(MaxTimeT)]);
subplot(3,1,2); plot(TimeStepFinal,Tfinal);
subplot(3,1,3); patch(PlotCoordsX,PlotCoordsY,Tplot,'Tag','T')
colorbar

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

8.2 SELECTED FUNCTIONS – FEM MATLAB CODE

Example of one capacitance matrix entry:

```
function C1 = CapMatrix1(Xi, Eta);

global e;
load ElementCoord.dat

   X = ElementCoord(((e*8)-7):(e*8),1);
   Y = ElementCoord(((e*8)-7):(e*8),2);
   N1A = (0.25*(1-Xi).*(1-Eta).*(-1-Xi-Eta)) .* (((((0.5*Xi)-(0.5*Xi.*Eta)-
            (0.25*Eta.^2)+(0.25*Eta))*X(1,1) + ...
                           (-Xi + (Xi.*Eta))*X(2,1) + ...
                           ((0.5*Xi)-(0.5*Xi.*Eta)+(0.25*Eta.^2)-(0.25*Eta))*X(3,1) + ...
                           (0.5-(0.5*Eta.^2))*X(4,1) + ...
                           ((0.5*Xi)+(0.5*Xi.*Eta)+(0.25*Eta.^2)+(0.25*Eta))*X(5,1) + ...
                           (-Xi - (Xi.*Eta))*X(6,1) + ...
                           ((0.5*Xi)+(0.5*Xi.*Eta)-(0.25*Eta.^2)-(0.25*Eta))*X(7,1) + ...
                           (-0.5+(0.5*Eta.^2))*X(8,1)).* ...
                           (((0.5*Eta)-(0.5*Xi.*Eta)-(0.25*Xi.^2)+(0.25*Xi))*Y(1,1) + ...
                           (-0.5+(0.5*Xi.^2))*Y(2,1) + ...
                           ((0.5*Eta)+(0.5*Xi.*Eta)-(0.25*Xi.^2)-(0.25*Xi))*Y(3,1) + ...
                           (-Eta - (Xi.*Eta))*Y(4,1) + ...
                           ((0.5*Eta)+(0.5*Xi.*Eta)+(0.25*Xi.^2)+(0.25*Xi))*Y(5,1) + ...
                           (0.5-(0.5*Xi.^2))*Y(6,1) + ...
                           ((0.5*Eta)-(0.5*Xi.*Eta)+(0.25*Xi.^2)-(0.25*Xi))*Y(7,1) + ...
```

```
                    (-Eta + (Eta.*Xi))*Y(8,1))) - ...
                    ((((0.5*Xi)-(0.5*Xi.*Eta)-(0.25*Eta.^2)+(0.25*Eta))*Y(1,1) + ...
                    (-Xi + (Xi.*Eta))*Y(2,1) + ...
                    ((0.5*Xi)-(0.5*Xi.*Eta)+(0.25*Eta.^2)-(0.25*Eta))*Y(3,1) + ...
                    (0.5-(0.5*Eta.^2))*Y(4,1) + ...
                    ((0.5*Xi)+(0.5*Xi.*Eta)+(0.25*Eta.^2)+(0.25*Eta))*Y(5,1) + ...
                    (-Xi - (Xi.*Eta))*Y(6,1) + ...
                    ((0.5*Xi)+(0.5*Xi.*Eta)-(0.25*Eta.^2)-(0.25*Eta))*Y(7,1) + ...
                    (-0.5+(0.5*Eta.^2))*Y(8,1)).* ...
                    (((0.5*Eta)-(0.5*Xi.*Eta)-(0.25*Xi.^2)+(0.25*Xi))*X(1,1) + ...
                    (-0.5+(0.5*Xi.^2))*X(2,1) + ...
                    ((0.5*Eta)+(0.5*Xi.*Eta)-(0.25*Xi.^2)-(0.25*Xi))*X(3,1) + ...
                    (-Eta - (Xi.*Eta))*X(4,1) + ...
                    ((0.5*Eta)+(0.5*Xi.*Eta)+(0.25*Xi.^2)+(0.25*Xi))*X(5,1) + ...
                    (0.5-(0.5*Xi.^2))*X(6,1) + ...
                    ((0.5*Eta)-(0.5*Xi.*Eta)+(0.25*Xi.^2)-(0.25*Xi))*X(7,1) + ...
                    (-Eta + (Eta.*Xi))*X(8,1))));
  N1B = 0.25*(1-Xi).*(1-Eta).*(-1-Xi-Eta);
  C1 = (N1A.*N1B);
```

Example of one conductance matrix entry:

```
function Kc1 = CondMatrix1(Xi, Eta);

global ek;
load ElementCoord.dat

   X = ElementCoord(((ek*8)-7):(ek*8),1);
   Y = ElementCoord(((ek*8)-7):(ek*8),2);

   N1N1Xi = (((2*((2*Y(3,1)-4*Y(4,1)+2*Y(1,1)-2*Y(1,1)*Xi-
4*Y(4,1)*Xi+2*Y(5,1)+2*Y(7,1)-2*Y(7,1)*Xi+2*Y(3,1)*Xi+2*Y(5,1)*Xi-4*Y(8,1)...
 +4*Y(8,1)*Xi)*Eta-2*Y(2,1)+Y(5,1)*Xi.^2-Y(1,1)*Xi.^2+Y(1,1)*Xi-
Y(3,1)*Xi.^2+2*Y(2,1)*Xi.^2+Y(7,1)*Xi.^2+...
 2*Y(6,1)-Y(3,1)*Xi+Y(5,1)*Xi-2*Y(6,1)*Xi.^2-Y(7,1)*Xi)/(2*Y(2,1)*Xi.^2*X(1,1)-
2*Y(2,1)*Xi.^2*X(3,1)+2*Y(2,1)*Xi.^2*X(5,1)...
 -2*Y(2,1)*Xi.^2*X(7,1)+4*X(6,1)*Xi.^3*Y(1,1)-2*X(6,1)*Xi.^2*Y(1,1)-
8*X(6,1)*Xi.^3*Y(2,1)+4*X(6,1)*Xi.^3*Y(3,1)...
 +2*X(6,1)*Xi.^2*Y(3,1)-2*X(6,1)*Xi.^2*Y(5,1)+2*X(6,1)*Xi.^2*Y(7,1)-
2*X(7,1)*Xi.^3*Y(1,1)+2*X(7,1)*Xi.^2*Y(1,1)...
 +4*X(7,1)*Xi.^3*Y(2,1)-2*X(7,1)*Xi.^3*Y(3,1)+2*X(7,1)*Xi.^2*Y(5,1)-
2*Y(8,1)*X(2,1)+2*Y(8,1)*X(6,1)+X(8,1)*Y(7,1)*Xi+2*Y(1,1)*Xi*X(2,1)...
 -2*Y(1,1)*Xi*X(6,1)-2*Y(3,1)*Xi*X(6,1)+2*X(8,1)*Y(2,1)-
2*X(8,1)*Y(6,1)+2*X(2,1)*Xi.^2*Y(7,1)+2*X(3,1)*Xi.^2*Y(1,1)...
```

```
 +2*X(3,1)*Xi.^3*Y(5,1)+2*X(3,1)*Xi.^2*Y(5,1)-
4*X(3,1)*Xi.^3*Y(6,1)+2*X(3,1)*Xi.^3*Y(7,1)-2*X(5,1)*Xi.^3*Y(1,1)...
 +4*X(5,1)*Xi.^3*Y(2,1)-2*X(5,1)*Xi.^3*Y(3,1)-2*X(5,1)*Xi.^2*Y(3,1)-
2*X(5,1)*Xi.^2*Y(7,1)+2*Y(4,1)*X(2,1)-2*Y(4,1)*X(6,1)...
 +(2*X(5,1)*Y(1,1)-2*X(7,1)*Y(5,1)-
4*Y(8,1)*X(3,1)+4*X(8,1)*Y(3,1)+2*X(3,1)*Y(1,1)+2*X(3,1)*Y(7,1)+4*X(8,1)*Y(5,1)-
2*X(1,1)*Y(5,1)...
 -4*X(4,1)*Y(7,1)+8*X(4,1)*Y(8,1)+4*Y(4,1)*X(1,1)-4*X(4,1)*Y(1,1)+2*X(5,1)*Y(7,1)-
4*Y(8,1)*X(5,1)-2*X(1,1)*Y(3,1)+4*Y(4,1)*X(7,1)...
 -8*X(8,1)*Y(4,1)-2*X(7,1)*Y(3,1))*Eta.^3+(2*X(5,1)*Y(7,1)-
X(1,1)*Y(6,1)+2*X(4,1)*Y(1,1)+3*Y(2,1)*Xi.^2*X(1,1)-3*Y(2,1)*Xi.^2*X(3,1)...
 -3*Y(2,1)*Xi.^2*X(5,1)+3*Y(2,1)*Xi.^2*X(7,1)+3*X(6,1)*Xi.^2*Y(1,1)-
3*X(6,1)*Xi.^2*Y(3,1)-3*X(6,1)*Xi.^2*Y(5,1)+3*X(6,1)*Xi.^2*Y(7,1)...
 -3*X(7,1)*Xi.^2*Y(1,1)+3*X(7,1)*Xi.^2*Y(5,1)+2*Y(8,1)*X(2,1)-
2*Y(8,1)*X(6,1)+2*X(5,1)*Y(3,1)...
 +5*X(8,1)*Y(7,1)*Xi-X(3,1)*Y(2,1)+4*Y(1,1)*Xi*X(2,1)-4*Y(1,1)*Xi*X(6,1)-
4*Y(3,1)*Xi*X(6,1)+8*Y(2,1)*Xi*X(4,1)...
 +8*Y(2,1)*Xi*X(8,1)-5*X(3,1)*Xi*Y(5,1)-3*X(3,1)*Xi*Y(7,1)+8*X(6,1)*Xi*Y(8,1)-
3*X(1,1)*Xi*Y(5,1)-5*X(1,1)*Xi*Y(7,1)...
 -8*Y(6,1)*Xi*X(8,1)-8*X(2,1)*Xi*Y(4,1)-8*X(2,1)*Xi*Y(8,1)-
2*X(8,1)*Y(2,1)+2*X(8,1)*Y(6,1)-3*X(2,1)*Xi.^2*Y(7,1)...
 +3*X(3,1)*Xi.^2*Y(1,1)-3*X(3,1)*Xi.^2*Y(5,1)+3*X(5,1)*Xi.^2*Y(3,1)-
3*X(5,1)*Xi.^2*Y(7,1)-2*X(3,1)*Y(5,1)+Y(3,1)*X(2,1)...
 -2*Y(4,1)*X(2,1)+2*Y(4,1)*X(6,1)+4*X(1,1)*Xi*Y(6,1)-
3*X(4,1)*Y(1,1)*Xi+6*X(4,1)*Y(2,1)*Xi.^2-3*X(4,1)*Y(3,1)*Xi.^2-...

5*X(4,1)*Y(3,1)*Xi+3*X(4,1)*Y(5,1)*Xi.^2+5*X(4,1)*Y(5,1)*Xi+3*X(4,1)*Y(7,1)*Xi.^2+3*
X(4,1)*Y(7,1)*Xi+4*X(5,1)*Xi*Y(6,1)...
 -4*X(7,1)*Xi*Y(2,1)+4*X(7,1)*Xi*Y(6,1)+3*X(8,1)*Y(1,1)*Xi.^2-5*X(8,1)*Y(1,1)*Xi-
6*X(8,1)*Y(2,1)*Xi.^2+3*X(8,1)*Y(3,1)*Xi.^2-...
 3*X(8,1)*Y(3,1)*Xi-2*X(3,1)*Y(1,1)+3*X(8,1)*Y(5,1)*Xi-Y(3,1)*X(6,1)+2*X(4,1)*Y(2,1)-
2*X(4,1)*Y(6,1)+2*Y(8,1)*X(7,1)...
 +Y(1,1)*X(6,1)-3*X(1,1)*Xi.^2*Y(3,1)+3*X(1,1)*Xi.^2*Y(7,1)-
3*X(2,1)*Xi.^2*Y(1,1)+3*X(2,1)*Xi.^2*Y(3,1)+3*X(2,1)*Xi.^2*Y(5,1)...
 -2*X(8,1)*Y(7,1)-Y(1,1)*X(2,1)-2*Y(8,1)*X(5,1)+2*Y(8,1)*X(3,1)-4*X(1,1)*Xi*Y(2,1)-
2*X(7,1)*Y(5,1)-3*Y(6,1)*Xi.^2*X(1,1)...
 +3*Y(6,1)*Xi.^2*X(3,1)+3*Y(6,1)*Xi.^2*X(5,1)-3*Y(6,1)*Xi.^2*X(7,1)-
2*Y(8,1)*X(1,1)+5*X(7,1)*Xi*Y(1,1)+3*X(7,1)*Xi*Y(3,1)...
 +3*X(5,1)*Xi*Y(1,1)+5*X(5,1)*Xi*Y(3,1)+Y(5,1)*X(2,1)-Y(5,1)*X(6,1)-
2*X(7,1)*Y(1,1)+2*Y(4,1)*X(7,1)+8*X(6,1)*Xi*Y(4,1)...
 -8*Y(6,1)*Xi*X(4,1)+X(1,1)*Y(2,1)-
2*Y(4,1)*X(5,1)+4*Y(3,1)*Xi*X(2,1)+6*X(8,1)*Y(6,1)*Xi.^2-3*X(8,1)*Y(5,1)*Xi.^2-...
 4*X(3,1)*Xi*Y(2,1)-3*X(4,1)*Y(1,1)*Xi.^2-4*Y(7,1)*Xi*X(6,1)-
3*X(8,1)*Y(7,1)*Xi.^2+4*X(3,1)*Xi*Y(6,1)+4*Y(7,1)*Xi*X(2,1)...
```

```
 -4*X(5,1)*Xi*Y(2,1)-
6*X(4,1)*Y(6,1)*Xi.^2+2*Y(4,1)*X(3,1)+2*X(1,1)*Y(7,1)+2*X(1,1)*Y(3,1)-
2*X(8,1)*Y(3,1)+2*X(8,1)*Y(5,1)-...
 2*Y(4,1)*X(1,1)+3*Y(4,1)*X(1,1)*Xi.^2+3*Y(4,1)*X(1,1)*Xi-
6*Y(4,1)*X(2,1)*Xi.^2+3*Y(4,1)*X(3,1)*Xi.^2+5*Y(4,1)*X(3,1)*Xi...
 -3*Y(4,1)*X(5,1)*Xi.^2-5*Y(4,1)*X(5,1)*Xi+6*Y(4,1)*X(6,1)*Xi.^2-3*Y(4,1)*X(7,1)*Xi.^2-
3*Y(4,1)*X(7,1)*Xi+4*Y(5,1)*Xi*X(2,1)...
 -4*Y(5,1)*Xi*X(6,1)-3*Y(8,1)*X(1,1)*Xi.^2+5*Y(8,1)*X(1,1)*Xi+6*Y(8,1)*X(2,1)*Xi.^2-
3*Y(8,1)*X(3,1)*Xi.^2+...
 3*Y(8,1)*X(3,1)*Xi+3*Y(8,1)*X(5,1)*Xi.^2-3*Y(8,1)*X(5,1)*Xi-
6*Y(8,1)*X(6,1)*Xi.^2+3*Y(8,1)*X(7,1)*Xi.^2-5*Y(8,1)*X(7,1)*Xi...
 +X(3,1)*Y(6,1)-X(7,1)*Y(6,1)-Y(7,1)*X(2,1)-
X(5,1)*Y(2,1)+2*X(8,1)*Y(1,1)+X(7,1)*Y(2,1)+Y(7,1)*X(6,1)+2*X(4,1)*Y(5,1)-
2*X(4,1)*Y(7,1)...
 +X(5,1)*Y(6,1)-2*X(4,1)*Y(3,1))*Eta.^2+(X(1,1)*Y(6,1)+2*X(4,1)*Y(1,1)-
5*Y(2,1)*Xi.^2*X(1,1)+5*Y(2,1)*Xi.^2*X(3,1)...
 +3*Y(2,1)*Xi.^2*X(5,1)-3*Y(2,1)*Xi.^2*X(7,1)+3*X(6,1)*Xi.^2*Y(1,1)-
3*X(6,1)*Xi.^2*Y(3,1)-5*X(6,1)*Xi.^2*Y(5,1)...
 +5*X(6,1)*Xi.^2*Y(7,1)+5*X(7,1)*Xi.^2*Y(5,1)+6*X(8,1)*Y(7,1)*Xi+X(3,1)*Y(2,1)-
6*Y(1,1)*Xi*X(2,1)-2*Y(1,1)*Xi*X(6,1)...
 -2*Y(3,1)*Xi*X(6,1)-8*X(4,1)*Y(8,1)+8*X(8,1)*Y(4,1)-8*Y(2,1)*Xi*X(4,1)-
8*Y(2,1)*Xi*X(8,1)+8*X(6,1)*Xi*Y(8,1)...
 +3*X(1,1)*Xi.^2*Y(5,1)-
8*Y(6,1)*Xi*X(8,1)+8*X(2,1)*Xi*Y(4,1)+8*X(2,1)*Xi*Y(8,1)+3*X(2,1)*Xi.^2*Y(7,1)-
5*X(3,1)*Xi.^2*Y(1,1)...
 -5*X(5,1)*Xi.^2*Y(7,1)-Y(3,1)*X(2,1)+2*X(1,1)*Xi*Y(6,1)+2*X(4,1)*Y(1,1)*Xi-
8*X(4,1)*Y(2,1)*Xi.^2+...

4*X(4,1)*Y(3,1)*Xi.^2+6*X(4,1)*Y(3,1)*Xi+4*X(4,1)*Y(5,1)*Xi.^2+6*X(4,1)*Y(5,1)*Xi+4*
X(4,1)*Y(7,1)*Xi.^2+2*X(4,1)*Y(7,1)*Xi...
 +6*X(5,1)*Xi*Y(6,1)+2*X(7,1)*Xi*Y(2,1)+6*X(7,1)*Xi*Y(6,1)-
4*X(8,1)*Y(1,1)*Xi.^2+6*X(8,1)*Y(1,1)*Xi...
 +8*X(8,1)*Y(2,1)*Xi.^2-
4*X(8,1)*Y(3,1)*Xi.^2+2*X(8,1)*Y(3,1)*Xi+2*X(8,1)*Y(5,1)*Xi+Y(3,1)*X(6,1)+2*Y(8,1)*X
(7,1)-Y(1,1)*X(6,1)...
 +5*X(1,1)*Xi.^2*Y(3,1)+5*X(2,1)*Xi.^2*Y(1,1)-5*X(2,1)*Xi.^2*Y(3,1)-
3*X(2,1)*Xi.^2*Y(5,1)-2*X(8,1)*Y(7,1)...
 +Y(1,1)*X(2,1)+2*Y(8,1)*X(5,1)+2*Y(8,1)*X(3,1)+6*X(1,1)*Xi*Y(2,1)-
3*Y(6,1)*Xi.^2*X(1,1)+3*Y(6,1)*Xi.^2*X(3,1)...
 +5*Y(6,1)*Xi.^2*X(5,1)-5*Y(6,1)*Xi.^2*X(7,1)+2*Y(8,1)*X(1,1)-
3*X(5,1)*Xi.^2*Y(1,1)+Y(5,1)*X(2,1)-Y(5,1)*X(6,1)-2*Y(4,1)*X(7,1)...
 -3*X(3,1)*Xi.^2*Y(7,1)+8*X(6,1)*Xi*Y(4,1)-8*Y(6,1)*Xi*X(4,1)-X(1,1)*Y(2,1)-
2*Y(4,1)*X(5,1)+3*X(7,1)*Xi.^2*Y(3,1)...
 -6*Y(3,1)*Xi*X(2,1)+8*X(8,1)*Y(6,1)*Xi.^2-
4*X(8,1)*Y(5,1)*Xi.^2+6*X(3,1)*Xi*Y(2,1)+4*X(4,1)*Y(1,1)*Xi.^2-6*Y(7,1)*Xi*X(6,1)...
```

```
 -4*X(8,1)*Y(7,1)*Xi.^2+2*X(3,1)*Xi*Y(6,1)-2*Y(7,1)*Xi*X(2,1)+2*X(5,1)*Xi*Y(2,1)-
8*X(4,1)*Y(6,1)*Xi.^2-2*Y(4,1)*X(3,1)...
 -2*X(8,1)*Y(3,1)-2*X(8,1)*Y(5,1)-2*Y(4,1)*X(1,1)-4*Y(4,1)*X(1,1)*Xi.^2-
2*Y(4,1)*X(1,1)*Xi+8*Y(4,1)*X(2,1)*Xi.^2-...
 4*Y(4,1)*X(3,1)*Xi.^2-6*Y(4,1)*X(3,1)*Xi-4*Y(4,1)*X(5,1)*Xi.^2-
6*Y(4,1)*X(5,1)*Xi+8*Y(4,1)*X(6,1)*Xi.^2-4*Y(4,1)*X(7,1)*Xi.^2-...
 2*Y(4,1)*X(7,1)*Xi-2*Y(5,1)*Xi*X(2,1)-6*Y(5,1)*Xi*X(6,1)+4*Y(8,1)*X(1,1)*Xi.^2-
6*Y(8,1)*X(1,1)*Xi...
 -8*Y(8,1)*X(2,1)*Xi.^2+4*Y(8,1)*X(3,1)*Xi.^2-2*Y(8,1)*X(3,1)*Xi+4*Y(8,1)*X(5,1)*Xi.^2-
2*Y(8,1)*X(5,1)*Xi-8*Y(8,1)*X(6,1)*Xi.^2+...
 4*Y(8,1)*X(7,1)*Xi.^2-6*Y(8,1)*X(7,1)*Xi-X(3,1)*Y(6,1)-X(7,1)*Y(6,1)-Y(7,1)*X(2,1)-
X(5,1)*Y(2,1)-2*X(8,1)*Y(1,1)...

+X(7,1)*Y(2,1)+Y(7,1)*X(6,1)+2*X(4,1)*Y(5,1)+2*X(4,1)*Y(7,1)+X(5,1)*Y(6,1)+2*X(4,1)*Y
(3,1))*Eta+2*X(1,1)*Xi*Y(6,1)...
 -8*X(2,1)*Xi*Y(6,1)+X(4,1)*Y(1,1)*Xi+2*X(4,1)*Y(2,1)*Xi.^2-X(4,1)*Y(3,1)*Xi.^2-
X(4,1)*Y(3,1)*Xi+X(4,1)*Y(5,1)*Xi.^2+X(4,1)*Y(5,1)*Xi...
 +X(4,1)*Y(7,1)*Xi.^2-X(4,1)*Y(7,1)*Xi+2*X(5,1)*Xi*Y(6,1)+8*X(6,1)*Xi*Y(2,1)-
2*X(7,1)*Xi*Y(2,1)+2*X(7,1)*Xi*Y(6,1)...
 +X(8,1)*Y(1,1)*Xi.^2-X(8,1)*Y(1,1)*Xi-
2*X(8,1)*Y(2,1)*Xi.^2+X(8,1)*Y(3,1)*Xi.^2+X(8,1)*Y(3,1)*Xi-X(8,1)*Y(5,1)*Xi-
2*X(4,1)*Y(2,1)...
 +2*X(4,1)*Y(6,1)-2*X(1,1)*Xi.^2*Y(3,1)+2*X(1,1)*Xi.^3*Y(5,1)-
4*X(1,1)*Xi.^3*Y(6,1)+2*X(1,1)*Xi.^3*Y(7,1)-2*X(1,1)*Xi.^2*Y(7,1)...
 -2*X(2,1)*Xi.^2*Y(1,1)+2*X(2,1)*Xi.^2*Y(3,1)-4*X(2,1)*Xi.^3*Y(5,1)-
2*X(2,1)*Xi.^2*Y(5,1)...
 +8*X(2,1)*Xi.^3*Y(6,1)-4*X(2,1)*Xi.^3*Y(7,1)-2*X(1,1)*Xi*Y(2,1)+2*Y(6,1)*Xi.^2*X(1,1)-
2*Y(6,1)*Xi.^2*X(3,1)+2*Y(6,1)*Xi.^2*X(5,1)...
 -2*Y(6,1)*Xi.^2*X(7,1)+2*Y(3,1)*Xi*X(2,1)+2*X(8,1)*Y(6,1)*Xi.^2-X(8,1)*Y(5,1)*Xi.^2-...
 2*X(3,1)*Xi*Y(2,1)-X(4,1)*Y(1,1)*Xi.^2-2*Y(7,1)*Xi*X(6,1)-
X(8,1)*Y(7,1)*Xi.^2+2*X(3,1)*Xi*Y(6,1)+2*Y(7,1)*Xi*X(2,1)...
 -2*X(5,1)*Xi*Y(2,1)-2*X(4,1)*Y(6,1)*Xi.^2+Y(4,1)*X(1,1)*Xi.^2-Y(4,1)*X(1,1)*Xi-
2*Y(4,1)*X(2,1)*Xi.^2+...
 Y(4,1)*X(3,1)*Xi.^2+Y(4,1)*X(3,1)*Xi-Y(4,1)*X(5,1)*Xi.^2-
Y(4,1)*X(5,1)*Xi+2*Y(4,1)*X(6,1)*Xi.^2-Y(4,1)*X(7,1)*Xi.^2+...
 Y(4,1)*X(7,1)*Xi+2*Y(5,1)*Xi*X(2,1)-2*Y(5,1)*Xi*X(6,1)-
Y(8,1)*X(1,1)*Xi.^2+Y(8,1)*X(1,1)*Xi+2*Y(8,1)*X(2,1)*Xi.^2-...
 Y(8,1)*X(3,1)*Xi.^2-Y(8,1)*X(3,1)*Xi+Y(8,1)*X(5,1)*Xi.^2+Y(8,1)*X(5,1)*Xi-
2*Y(8,1)*X(6,1)*Xi.^2+Y(8,1)*X(7,1)*Xi.^2-Y(8,1)*X(7,1)*Xi)*...
          (0.5*(Xi-(Xi.*Eta))+0.25*(-(Eta.^2)+Eta)))...
 +(-2*((-2*Y(4,1)-Y(1,1)-Y(7,1)+Y(3,1)+Y(5,1)+2*Y(8,1))*Eta.^2+(-2*Y(3,1)*Xi-
2*Y(1,1)*Xi+Y(5,1)+2*Y(7,1)*Xi-Y(7,1)+2*Y(5,1)*Xi...
 +4*Y(2,1)*Xi+Y(1,1)-4*Y(6,1)*Xi-Y(3,1))*Eta+2*Y(1,1)*Xi+2*Y(7,1)*Xi+2*Y(3,1)*Xi...
 -4*Y(6,1)*Xi-4*Y(2,1)*Xi-2*Y(8,1)+2*Y(4,1)+2*Y(5,1)*Xi)/(2*Y(2,1)*Xi.^2*X(1,1)-
2*Y(2,1)*Xi.^2*X(3,1)+2*Y(2,1)*Xi.^2*X(5,1)...
```

```
 -2*Y(2,1)*Xi.^2*X(7,1)+4*X(6,1)*Xi.^3*Y(1,1)-2*X(6,1)*Xi.^2*Y(1,1)-
8*X(6,1)*Xi.^3*Y(2,1)...
 +4*X(6,1)*Xi.^3*Y(3,1)+2*X(6,1)*Xi.^2*Y(3,1)-
2*X(6,1)*Xi.^2*Y(5,1)+2*X(6,1)*Xi.^2*Y(7,1)-2*X(7,1)*Xi.^3*Y(1,1)...
 +2*X(7,1)*Xi.^2*Y(1,1)+4*X(7,1)*Xi.^3*Y(2,1)-
2*X(7,1)*Xi.^3*Y(3,1)+2*X(7,1)*Xi.^2*Y(5,1)-2*Y(8,1)*X(2,1)+2*Y(8,1)*X(6,1)...
 +X(8,1)*Y(7,1)*Xi+2*Y(1,1)*Xi*X(2,1)-2*Y(1,1)*Xi*X(6,1)-
2*Y(3,1)*Xi*X(6,1)+2*X(8,1)*Y(2,1)-2*X(8,1)*Y(6,1)...

+2*X(2,1)*Xi.^2*Y(7,1)+2*X(3,1)*Xi.^2*Y(1,1)+2*X(3,1)*Xi.^3*Y(5,1)+2*X(3,1)*Xi.^2*Y(5
,1)-4*X(3,1)*Xi.^3*Y(6,1)+2*X(3,1)*Xi.^3*Y(7,1)-...
 2*X(5,1)*Xi.^3*Y(1,1)+4*X(5,1)*Xi.^3*Y(2,1)-2*X(5,1)*Xi.^3*Y(3,1)-
2*X(5,1)*Xi.^2*Y(3,1)-2*X(5,1)*Xi.^2*Y(7,1)...
 +2*Y(4,1)*X(2,1)-2*Y(4,1)*X(6,1)+(2*X(5,1)*Y(1,1)-2*X(7,1)*Y(5,1)-
4*Y(8,1)*X(3,1)+4*X(8,1)*Y(3,1)+2*X(3,1)*Y(1,1)+2*X(3,1)*Y(7,1)...
 +4*X(8,1)*Y(5,1)-2*X(1,1)*Y(5,1)-4*X(4,1)*Y(7,1)+8*X(4,1)*Y(8,1)+4*Y(4,1)*X(1,1)-
4*X(4,1)*Y(1,1)+2*X(5,1)*Y(7,1)...
 -4*Y(8,1)*X(5,1)-2*X(1,1)*Y(3,1)+4*Y(4,1)*X(7,1)-8*X(8,1)*Y(4,1)-
2*X(7,1)*Y(3,1))*Eta.^3+(2*X(5,1)*Y(7,1)-X(1,1)*Y(6,1)+2*X(4,1)*Y(1,1)...
 +3*Y(2,1)*Xi.^2*X(1,1)-3*Y(2,1)*Xi.^2*X(3,1)-
3*Y(2,1)*Xi.^2*X(5,1)+3*Y(2,1)*Xi.^2*X(7,1)+3*X(6,1)*Xi.^2*Y(1,1)...
 -3*X(6,1)*Xi.^2*Y(3,1)-3*X(6,1)*Xi.^2*Y(5,1)+3*X(6,1)*Xi.^2*Y(7,1)-
3*X(7,1)*Xi.^2*Y(1,1)+3*X(7,1)*Xi.^2*Y(5,1)...
 +2*Y(8,1)*X(2,1)-2*Y(8,1)*X(6,1)+2*X(5,1)*Y(3,1)+5*X(8,1)*Y(7,1)*Xi-
X(3,1)*Y(2,1)+4*Y(1,1)*Xi*X(2,1)-4*Y(1,1)*Xi*X(6,1)...
 -4*Y(3,1)*Xi*X(6,1)+8*Y(2,1)*Xi*X(4,1)+8*Y(2,1)*Xi*X(8,1)-5*X(3,1)*Xi*Y(5,1)-
3*X(3,1)*Xi*Y(7,1)+8*X(6,1)*Xi*Y(8,1)...
 -3*X(1,1)*Xi*Y(5,1)-5*X(1,1)*Xi*Y(7,1)-8*Y(6,1)*Xi*X(8,1)-8*X(2,1)*Xi*Y(4,1)-
8*X(2,1)*Xi*Y(8,1)-2*X(8,1)*Y(2,1)...
 +2*X(8,1)*Y(6,1)-3*X(2,1)*Xi.^2*Y(7,1)+3*X(3,1)*Xi.^2*Y(1,1)-
3*X(3,1)*Xi.^2*Y(5,1)+3*X(5,1)*Xi.^2*Y(3,1)...
 -3*X(5,1)*Xi.^2*Y(7,1)-2*X(3,1)*Y(5,1)+Y(3,1)*X(2,1)-
2*Y(4,1)*X(2,1)+2*Y(4,1)*X(6,1)+4*X(1,1)*Xi*Y(6,1)...
 -3*X(4,1)*Y(1,1)*Xi+6*X(4,1)*Y(2,1)*Xi.^2-3*X(4,1)*Y(3,1)*Xi.^2-
5*X(4,1)*Y(3,1)*Xi+3*X(4,1)*Y(5,1)*Xi.^2+5*X(4,1)*Y(5,1)*Xi...
 +3*X(4,1)*Y(7,1)*Xi.^2+3*X(4,1)*Y(7,1)*Xi+4*X(5,1)*Xi*Y(6,1)-
4*X(7,1)*Xi*Y(2,1)+4*X(7,1)*Xi*Y(6,1)+3*X(8,1)*Y(1,1)*Xi.^2-...
 5*X(8,1)*Y(1,1)*Xi-6*X(8,1)*Y(2,1)*Xi.^2+3*X(8,1)*Y(3,1)*Xi.^2-3*X(8,1)*Y(3,1)*Xi-
2*X(3,1)*Y(1,1)...
 +3*X(8,1)*Y(5,1)*Xi-Y(3,1)*X(6,1)+2*X(4,1)*Y(2,1)-
2*X(4,1)*Y(6,1)+2*Y(8,1)*X(7,1)+Y(1,1)*X(6,1)-3*X(1,1)*Xi.^2*Y(3,1)...
 +3*X(1,1)*Xi.^2*Y(7,1)-
3*X(2,1)*Xi.^2*Y(1,1)+3*X(2,1)*Xi.^2*Y(3,1)+3*X(2,1)*Xi.^2*Y(5,1)-2*X(8,1)*Y(7,1)-
Y(1,1)*X(2,1)...
 -2*Y(8,1)*X(5,1)+2*Y(8,1)*X(3,1)-4*X(1,1)*Xi*Y(2,1)-2*X(7,1)*Y(5,1)-
3*Y(6,1)*Xi.^2*X(1,1)+3*Y(6,1)*Xi.^2*X(3,1)...
```

```
 +3*Y(6,1)*Xi.^2*X(5,1)-3*Y(6,1)*Xi.^2*X(7,1)-
2*Y(8,1)*X(1,1)+5*X(7,1)*Xi*Y(1,1)+3*X(7,1)*Xi*Y(3,1)+3*X(5,1)*Xi*Y(1,1)...
 +5*X(5,1)*Xi*Y(3,1)+Y(5,1)*X(2,1)-Y(5,1)*X(6,1)-
2*X(7,1)*Y(1,1)+2*Y(4,1)*X(7,1)+8*X(6,1)*Xi*Y(4,1)-8*Y(6,1)*Xi*X(4,1)...
 +X(1,1)*Y(2,1)-2*Y(4,1)*X(5,1)+4*Y(3,1)*Xi*X(2,1)+6*X(8,1)*Y(6,1)*Xi.^2-
3*X(8,1)*Y(5,1)*Xi.^2-4*X(3,1)*Xi*Y(2,1)-3*X(4,1)*Y(1,1)*Xi.^2-...
 4*Y(7,1)*Xi*X(6,1)-3*X(8,1)*Y(7,1)*Xi.^2+4*X(3,1)*Xi*Y(6,1)+4*Y(7,1)*Xi*X(2,1)-
4*X(5,1)*Xi*Y(2,1)-6*X(4,1)*Y(6,1)*Xi.^2+...
 2*Y(4,1)*X(3,1)+2*X(1,1)*Y(7,1)+2*X(1,1)*Y(3,1)-2*X(8,1)*Y(3,1)+2*X(8,1)*Y(5,1)-
2*Y(4,1)*X(1,1)+3*Y(4,1)*X(1,1)*Xi.^2+...
 3*Y(4,1)*X(1,1)*Xi-6*Y(4,1)*X(2,1)*Xi.^2+3*Y(4,1)*X(3,1)*Xi.^2+5*Y(4,1)*X(3,1)*Xi-
3*Y(4,1)*X(5,1)*Xi.^2-5*Y(4,1)*X(5,1)*Xi...
 +6*Y(4,1)*X(6,1)*Xi.^2-3*Y(4,1)*X(7,1)*Xi.^2-3*Y(4,1)*X(7,1)*Xi+4*Y(5,1)*Xi*X(2,1)-
4*Y(5,1)*Xi*X(6,1)...
 -3*Y(8,1)*X(1,1)*Xi.^2+5*Y(8,1)*X(1,1)*Xi+6*Y(8,1)*X(2,1)*Xi.^2-
3*Y(8,1)*X(3,1)*Xi.^2+3*Y(8,1)*X(3,1)*Xi+3*Y(8,1)*X(5,1)*Xi.^2-...
 3*Y(8,1)*X(5,1)*Xi-6*Y(8,1)*X(6,1)*Xi.^2+3*Y(8,1)*X(7,1)*Xi.^2-
5*Y(8,1)*X(7,1)*Xi+X(3,1)*Y(6,1)...
 -X(7,1)*Y(6,1)-Y(7,1)*X(2,1)-
X(5,1)*Y(2,1)+2*X(8,1)*Y(1,1)+X(7,1)*Y(2,1)+Y(7,1)*X(6,1)+2*X(4,1)*Y(5,1)-
2*X(4,1)*Y(7,1)...
 +X(5,1)*Y(6,1)-2*X(4,1)*Y(3,1))*Eta.^2+(X(1,1)*Y(6,1)+2*X(4,1)*Y(1,1)-
5*Y(2,1)*Xi.^2*X(1,1)+5*Y(2,1)*Xi.^2*X(3,1)+3*Y(2,1)*Xi.^2*X(5,1)...
 -3*Y(2,1)*Xi.^2*X(7,1)+3*X(6,1)*Xi.^2*Y(1,1)-3*X(6,1)*Xi.^2*Y(3,1)-
5*X(6,1)*Xi.^2*Y(5,1)+5*X(6,1)*Xi.^2*Y(7,1)+...
 5*X(7,1)*Xi.^2*Y(5,1)+6*X(8,1)*Y(7,1)*Xi+X(3,1)*Y(2,1)-6*Y(1,1)*Xi*X(2,1)-
2*Y(1,1)*Xi*X(6,1)-2*Y(3,1)*Xi*X(6,1)...
 -8*X(4,1)*Y(8,1)+8*X(8,1)*Y(4,1)-8*Y(2,1)*Xi*X(4,1)-
8*Y(2,1)*Xi*X(8,1)+8*X(6,1)*Xi*Y(8,1)+3*X(1,1)*Xi.^2*Y(5,1)...
 -8*Y(6,1)*Xi*X(8,1)+8*X(2,1)*Xi*Y(4,1)+8*X(2,1)*Xi*Y(8,1)+3*X(2,1)*Xi.^2*Y(7,1)-
5*X(3,1)*Xi.^2*Y(1,1)-5*X(5,1)*Xi.^2*Y(7,1)...
 -Y(3,1)*X(2,1)+2*X(1,1)*Xi*Y(6,1)+2*X(4,1)*Y(1,1)*Xi-
8*X(4,1)*Y(2,1)*Xi.^2+4*X(4,1)*Y(3,1)*Xi.^2+6*X(4,1)*Y(3,1)*Xi...

+4*X(4,1)*Y(5,1)*Xi.^2+6*X(4,1)*Y(5,1)*Xi+4*X(4,1)*Y(7,1)*Xi.^2+2*X(4,1)*Y(7,1)*Xi+6
*X(5,1)*Xi*Y(6,1)+2*X(7,1)*Xi*Y(2,1)...
 +6*X(7,1)*Xi*Y(6,1)-4*X(8,1)*Y(1,1)*Xi.^2+6*X(8,1)*Y(1,1)*Xi+8*X(8,1)*Y(2,1)*Xi.^2-
4*X(8,1)*Y(3,1)*Xi.^2+2*X(8,1)*Y(3,1)*Xi...
 +2*X(8,1)*Y(5,1)*Xi+Y(3,1)*X(6,1)+2*Y(8,1)*X(7,1)-
Y(1,1)*X(6,1)+5*X(1,1)*Xi.^2*Y(3,1)+5*X(2,1)*Xi.^2*Y(1,1)...
 -5*X(2,1)*Xi.^2*Y(3,1)-3*X(2,1)*Xi.^2*Y(5,1)-
2*X(8,1)*Y(7,1)+Y(1,1)*X(2,1)+2*Y(8,1)*X(5,1)+2*Y(8,1)*X(3,1)...
 +6*X(1,1)*Xi*Y(2,1)-
3*Y(6,1)*Xi.^2*X(1,1)+3*Y(6,1)*Xi.^2*X(3,1)+5*Y(6,1)*Xi.^2*X(5,1)-
5*Y(6,1)*Xi.^2*X(7,1)+2*Y(8,1)*X(1,1)...
```

```
 -3*X(5,1)*Xi.^2*Y(1,1)+Y(5,1)*X(2,1)-Y(5,1)*X(6,1)-2*Y(4,1)*X(7,1)-
3*X(3,1)*Xi.^2*Y(7,1)+8*X(6,1)*Xi*Y(4,1)...
 -8*Y(6,1)*Xi*X(4,1)-X(1,1)*Y(2,1)-2*Y(4,1)*X(5,1)+3*X(7,1)*Xi.^2*Y(3,1)-
6*Y(3,1)*Xi*X(2,1)+8*X(8,1)*Y(6,1)*Xi.^2-4*X(8,1)*Y(5,1)*Xi.^2+...
 6*X(3,1)*Xi*Y(2,1)+4*X(4,1)*Y(1,1)*Xi.^2-6*Y(7,1)*Xi*X(6,1)-
4*X(8,1)*Y(7,1)*Xi.^2+2*X(3,1)*Xi*Y(6,1)-2*Y(7,1)*Xi*X(2,1)...
 +2*X(5,1)*Xi*Y(2,1)-8*X(4,1)*Y(6,1)*Xi.^2-2*Y(4,1)*X(3,1)-2*X(8,1)*Y(3,1)-
2*X(8,1)*Y(5,1)-2*Y(4,1)*X(1,1)...
 -4*Y(4,1)*X(1,1)*Xi.^2-2*Y(4,1)*X(1,1)*Xi+8*Y(4,1)*X(2,1)*Xi.^2-4*Y(4,1)*X(3,1)*Xi.^2-
6*Y(4,1)*X(3,1)*Xi-4*Y(4,1)*X(5,1)*Xi.^2-...
 6*Y(4,1)*X(5,1)*Xi+8*Y(4,1)*X(6,1)*Xi.^2-4*Y(4,1)*X(7,1)*Xi.^2-2*Y(4,1)*X(7,1)*Xi...
 -2*Y(5,1)*Xi*X(2,1)-6*Y(5,1)*Xi*X(6,1)+4*Y(8,1)*X(1,1)*Xi.^2-6*Y(8,1)*X(1,1)*Xi-
8*Y(8,1)*X(2,1)*Xi.^2+4*Y(8,1)*X(3,1)*Xi.^2-...
 2*Y(8,1)*X(3,1)*Xi+4*Y(8,1)*X(5,1)*Xi.^2-2*Y(8,1)*X(5,1)*Xi-8*Y(8,1)*X(6,1)*Xi.^2+...
 4*Y(8,1)*X(7,1)*Xi.^2-6*Y(8,1)*X(7,1)*Xi-X(3,1)*Y(6,1)-X(7,1)*Y(6,1)-Y(7,1)*X(2,1)-
X(5,1)*Y(2,1)-2*X(8,1)*Y(1,1)...

+X(7,1)*Y(2,1)+Y(7,1)*X(6,1)+2*X(4,1)*Y(5,1)+2*X(4,1)*Y(7,1)+X(5,1)*Y(6,1)+2*X(4,1)*Y
(3,1))*Eta+2*X(1,1)*Xi*Y(6,1)-8*X(2,1)*Xi*Y(6,1)...
 +X(4,1)*Y(1,1)*Xi+2*X(4,1)*Y(2,1)*Xi.^2-X(4,1)*Y(3,1)*Xi.^2-
X(4,1)*Y(3,1)*Xi+X(4,1)*Y(5,1)*Xi.^2+X(4,1)*Y(5,1)*Xi...
 +X(4,1)*Y(7,1)*Xi.^2-X(4,1)*Y(7,1)*Xi+2*X(5,1)*Xi*Y(6,1)+8*X(6,1)*Xi*Y(2,1)-
2*X(7,1)*Xi*Y(2,1)+2*X(7,1)*Xi*Y(6,1)...
 +X(8,1)*Y(1,1)*Xi.^2-X(8,1)*Y(1,1)*Xi-
2*X(8,1)*Y(2,1)*Xi.^2+X(8,1)*Y(3,1)*Xi.^2+X(8,1)*Y(3,1)*Xi-X(8,1)*Y(5,1)*Xi...
 -2*X(4,1)*Y(2,1)+2*X(4,1)*Y(6,1)-2*X(1,1)*Xi.^2*Y(3,1)+2*X(1,1)*Xi.^3*Y(5,1)-
4*X(1,1)*Xi.^3*Y(6,1)+2*X(1,1)*Xi.^3*Y(7,1)...
 -2*X(1,1)*Xi.^2*Y(7,1)-2*X(2,1)*Xi.^2*Y(1,1)+2*X(2,1)*Xi.^2*Y(3,1)-
4*X(2,1)*Xi.^3*Y(5,1)-2*X(2,1)*Xi.^2*Y(5,1)...
 +8*X(2,1)*Xi.^3*Y(6,1)-4*X(2,1)*Xi.^3*Y(7,1)-2*X(1,1)*Xi*Y(2,1)+2*Y(6,1)*Xi.^2*X(1,1)-
2*Y(6,1)*Xi.^2*X(3,1)...
 +2*Y(6,1)*Xi.^2*X(5,1)-
2*Y(6,1)*Xi.^2*X(7,1)+2*Y(3,1)*Xi*X(2,1)+2*X(8,1)*Y(6,1)*Xi.^2-X(8,1)*Y(5,1)*Xi.^2-
2*X(3,1)*Xi*Y(2,1)...
 -X(4,1)*Y(1,1)*Xi.^2-2*Y(7,1)*Xi*X(6,1)-
X(8,1)*Y(7,1)*Xi.^2+2*X(3,1)*Xi*Y(6,1)+2*Y(7,1)*Xi*X(2,1)...
 -2*X(5,1)*Xi*Y(2,1)-2*X(4,1)*Y(6,1)*Xi.^2+Y(4,1)*X(1,1)*Xi.^2-Y(4,1)*X(1,1)*Xi-
2*Y(4,1)*X(2,1)*Xi.^2+Y(4,1)*X(3,1)*Xi.^2+...
 Y(4,1)*X(3,1)*Xi-Y(4,1)*X(5,1)*Xi.^2-Y(4,1)*X(5,1)*Xi+2*Y(4,1)*X(6,1)*Xi.^2-
Y(4,1)*X(7,1)*Xi.^2+Y(4,1)*X(7,1)*Xi...
 +2*Y(5,1)*Xi*X(2,1)-2*Y(5,1)*Xi*X(6,1)-
Y(8,1)*X(1,1)*Xi.^2+Y(8,1)*X(1,1)*Xi+2*Y(8,1)*X(2,1)*Xi.^2-Y(8,1)*X(3,1)*Xi.^2-...
 Y(8,1)*X(3,1)*Xi+Y(8,1)*X(5,1)*Xi.^2+Y(8,1)*X(5,1)*Xi-
2*Y(8,1)*X(6,1)*Xi.^2+Y(8,1)*X(7,1)*Xi.^2-Y(8,1)*X(7,1)*Xi)*...
                        (0.5*(Eta-(Xi.*Eta))+0.25*(-(Xi.^2)+Xi)))).^2)...
                                .*(((((0.5*Xi)-(0.5*Xi.*Eta)-(0.25*Eta.^2)+(0.25*Eta))*X(1,1) + ...
```

```
(-Xi + (Xi.*Eta))*X(2,1) + ...
((0.5*Xi)-(0.5*Xi.*Eta)+(0.25*Eta.^2)-(0.25*Eta))*X(3,1) + ...
(0.5-(0.5*Eta.^2))*X(4,1) + ...
((0.5*Xi)+(0.5*Xi.*Eta)+(0.25*Eta.^2)+(0.25*Eta))*X(5,1) + ...
(-Xi - (Xi.*Eta))*X(6,1) + ...
((0.5*Xi)+(0.5*Xi.*Eta)-(0.25*Eta.^2)-(0.25*Eta))*X(7,1) + ...
(-0.5+(0.5*Eta.^2))*X(8,1)).* ...
(((0.5*Eta)-(0.5*Xi.*Eta)-(0.25*Xi.^2)+(0.25*Xi))*Y(1,1) + ...
(-0.5+(0.5*Xi.^2))*Y(2,1) + ...
((0.5*Eta)+(0.5*Xi.*Eta)-(0.25*Xi.^2)-(0.25*Xi))*Y(3,1) + ...
(-Eta - (Xi.*Eta))*Y(4,1) + ...
((0.5*Eta)+(0.5*Xi.*Eta)+(0.25*Xi.^2)+(0.25*Xi))*Y(5,1) + ...
(0.5-(0.5*Xi.^2))*Y(6,1) + ...
((0.5*Eta)-(0.5*Xi.*Eta)+(0.25*Xi.^2)-(0.25*Xi))*Y(7,1) + ...
(-Eta + (Eta.*Xi))*Y(8,1))) - ...
((((0.5*Xi)-(0.5*Xi.*Eta)-(0.25*Eta.^2)+(0.25*Eta))*Y(1,1) + ...
(-Xi + (Xi.*Eta))*Y(2,1) + ...
((0.5*Xi)-(0.5*Xi.*Eta)+(0.25*Eta.^2)-(0.25*Eta))*Y(3,1) + ...
(0.5-(0.5*Eta.^2))*Y(4,1) + ...
((0.5*Xi)+(0.5*Xi.*Eta)+(0.25*Eta.^2)+(0.25*Eta))*Y(5,1) + ...
(-Xi - (Xi.*Eta))*Y(6,1) + ...
((0.5*Xi)+(0.5*Xi.*Eta)-(0.25*Eta.^2)-(0.25*Eta))*Y(7,1) + ...
(-0.5+(0.5*Eta.^2))*Y(8,1)).* ...
(((0.5*Eta)-(0.5*Xi.*Eta)-(0.25*Xi.^2)+(0.25*Xi))*X(1,1) + ...
(-0.5+(0.5*Xi.^2))*X(2,1) + ...
((0.5*Eta)+(0.5*Xi.*Eta)-(0.25*Xi.^2)-(0.25*Xi))*X(3,1) + ...
(-Eta - (Xi.*Eta))*X(4,1) + ...
((0.5*Eta)+(0.5*Xi.*Eta)+(0.25*Xi.^2)+(0.25*Xi))*X(5,1) + ...
(0.5-(0.5*Xi.^2))*X(6,1) + ...
((0.5*Eta)-(0.5*Xi.*Eta)+(0.25*Xi.^2)-(0.25*Xi))*X(7,1) + ...
(-Eta + (Eta.*Xi))*X(8,1))));
```

N1N1Eta = (((2*((-2*X(3,1)+4*X(4,1)-2*X(1,1)+2*X(1,1)*Xi+4*X(4,1)*Xi-2*X(5,1)-2*X(7,1)+2*X(7,1)*Xi-2*X(3,1)*Xi-2*X(5,1)*Xi...
+4*X(8,1)-4*X(8,1)*Xi)*Eta+2*X(2,1)-X(5,1)*Xi.^2+X(1,1)*Xi.^2-X(1,1)*Xi+X(3,1)*Xi.^2-2*X(2,1)*Xi.^2-...
X(7,1)*Xi.^2-2*X(6,1)+X(3,1)*Xi-X(5,1)*Xi+2*X(6,1)*Xi.^2+X(7,1)*Xi)/(2*Y(2,1)*Xi.^2*X(1,1)-2*Y(2,1)*Xi.^2*X(3,1)...
+2*Y(2,1)*Xi.^2*X(5,1)-2*Y(2,1)*Xi.^2*X(7,1)+4*X(6,1)*Xi.^3*Y(1,1)-2*X(6,1)*Xi.^2*Y(1,1)...
-8*X(6,1)*Xi.^3*Y(2,1)+4*X(6,1)*Xi.^3*Y(3,1)+2*X(6,1)*Xi.^2*Y(3,1)-2*X(6,1)*Xi.^2*Y(5,1)+2*X(6,1)*Xi.^2*Y(7,1)...
-2*X(7,1)*Xi.^3*Y(1,1)+2*X(7,1)*Xi.^2*Y(1,1)+4*X(7,1)*Xi.^3*Y(2,1)-2*X(7,1)*Xi.^3*Y(3,1)...

+2*X(7,1)*Xi.^2*Y(5,1)-
2*Y(8,1)*X(2,1)+2*Y(8,1)*X(6,1)+X(8,1)*Y(7,1)*Xi+2*Y(1,1)*Xi*X(2,1)-
2*Y(1,1)*Xi*X(6,1)...
-2*Y(3,1)*Xi*X(6,1)+2*X(8,1)*Y(2,1)-
2*X(8,1)*Y(6,1)+2*X(2,1)*Xi.^2*Y(7,1)+2*X(3,1)*Xi.^2*Y(1,1)...
+2*X(3,1)*Xi.^3*Y(5,1)+2*X(3,1)*Xi.^2*Y(5,1)-
4*X(3,1)*Xi.^3*Y(6,1)+2*X(3,1)*Xi.^3*Y(7,1)-2*X(5,1)*Xi.^3*Y(1,1)...
+4*X(5,1)*Xi.^3*Y(2,1)-2*X(5,1)*Xi.^3*Y(3,1)-2*X(5,1)*Xi.^2*Y(3,1)-
2*X(5,1)*Xi.^2*Y(7,1)...
+2*Y(4,1)*X(2,1)-2*Y(4,1)*X(6,1)+(2*X(5,1)*Y(1,1)-2*X(7,1)*Y(5,1)-
4*Y(8,1)*X(3,1)+4*X(8,1)*Y(3,1)+2*X(3,1)*Y(1,1)...
+2*X(3,1)*Y(7,1)+4*X(8,1)*Y(5,1)-2*X(1,1)*Y(5,1)-
4*X(4,1)*Y(7,1)+8*X(4,1)*Y(8,1)+4*Y(4,1)*X(1,1)-4*X(4,1)*Y(1,1)+2*X(5,1)*Y(7,1)...
-4*Y(8,1)*X(5,1)-2*X(1,1)*Y(3,1)+4*Y(4,1)*X(7,1)-8*X(8,1)*Y(4,1)-
2*X(7,1)*Y(3,1))*Eta.^3+(2*X(5,1)*Y(7,1)...
-X(1,1)*Y(6,1)+2*X(4,1)*Y(1,1)+3*Y(2,1)*Xi.^2*X(1,1)-3*Y(2,1)*Xi.^2*X(3,1)-
3*Y(2,1)*Xi.^2*X(5,1)...
+3*Y(2,1)*Xi.^2*X(7,1)+3*X(6,1)*Xi.^2*Y(1,1)-3*X(6,1)*Xi.^2*Y(3,1)-
3*X(6,1)*Xi.^2*Y(5,1)+3*X(6,1)*Xi.^2*Y(7,1)...
-3*X(7,1)*Xi.^2*Y(1,1)+3*X(7,1)*Xi.^2*Y(5,1)+2*Y(8,1)*X(2,1)-2*Y(8,1)*X(6,1)...
+2*X(5,1)*Y(3,1)+5*X(8,1)*Y(7,1)*Xi-X(3,1)*Y(2,1)+4*Y(1,1)*Xi*X(2,1)-
4*Y(1,1)*Xi*X(6,1)-4*Y(3,1)*Xi*X(6,1)...
+8*Y(2,1)*Xi*X(4,1)+8*Y(2,1)*Xi*X(8,1)-5*X(3,1)*Xi*Y(5,1)-
3*X(3,1)*Xi*Y(7,1)+8*X(6,1)*Xi*Y(8,1)...
-3*X(1,1)*Xi*Y(5,1)-5*X(1,1)*Xi*Y(7,1)-8*Y(6,1)*Xi*X(8,1)-8*X(2,1)*Xi*Y(4,1)-
8*X(2,1)*Xi*Y(8,1)-2*X(8,1)*Y(2,1)...
+2*X(8,1)*Y(6,1)-3*X(2,1)*Xi.^2*Y(7,1)+3*X(3,1)*Xi.^2*Y(1,1)-3*X(3,1)*Xi.^2*Y(5,1)...
+3*X(5,1)*Xi.^2*Y(3,1)-3*X(5,1)*Xi.^2*Y(7,1)-2*X(3,1)*Y(5,1)+Y(3,1)*X(2,1)-
2*Y(4,1)*X(2,1)+2*Y(4,1)*X(6,1)...
+4*X(1,1)*Xi*Y(6,1)-3*X(4,1)*Y(1,1)*Xi+6*X(4,1)*Y(2,1)*Xi.^2-3*X(4,1)*Y(3,1)*Xi.^2-
5*X(4,1)*Y(3,1)*Xi...

+3*X(4,1)*Y(5,1)*Xi.^2+5*X(4,1)*Y(5,1)*Xi+3*X(4,1)*Y(7,1)*Xi.^2+3*X(4,1)*Y(7,1)*Xi+4
*X(5,1)*Xi*Y(6,1)...
-4*X(7,1)*Xi*Y(2,1)+4*X(7,1)*Xi*Y(6,1)+3*X(8,1)*Y(1,1)*Xi.^2-5*X(8,1)*Y(1,1)*Xi-
6*X(8,1)*Y(2,1)*Xi.^2+...
3*X(8,1)*Y(3,1)*Xi.^2-3*X(8,1)*Y(3,1)*Xi-2*X(3,1)*Y(1,1)+3*X(8,1)*Y(5,1)*Xi-
Y(3,1)*X(6,1)+2*X(4,1)*Y(2,1)...
-2*X(4,1)*Y(6,1)+2*Y(8,1)*X(7,1)+Y(1,1)*X(6,1)-
3*X(1,1)*Xi.^2*Y(3,1)+3*X(1,1)*Xi.^2*Y(7,1)-3*X(2,1)*Xi.^2*Y(1,1)...
+3*X(2,1)*Xi.^2*Y(3,1)+3*X(2,1)*Xi.^2*Y(5,1)-2*X(8,1)*Y(7,1)-Y(1,1)*X(2,1)-
2*Y(8,1)*X(5,1)+2*Y(8,1)*X(3,1)...
-4*X(1,1)*Xi*Y(2,1)-2*X(7,1)*Y(5,1)-
3*Y(6,1)*Xi.^2*X(1,1)+3*Y(6,1)*Xi.^2*X(3,1)+3*Y(6,1)*Xi.^2*X(5,1)...

```
 -3*Y(6,1)*Xi.^2*X(7,1)-
2*Y(8,1)*X(1,1)+5*X(7,1)*Xi*Y(1,1)+3*X(7,1)*Xi*Y(3,1)+3*X(5,1)*Xi*Y(1,1)+5*X(5,1)*Xi
*Y(3,1)...
 +Y(5,1)*X(2,1)-Y(5,1)*X(6,1)-2*X(7,1)*Y(1,1)+2*Y(4,1)*X(7,1)+8*X(6,1)*Xi*Y(4,1)-
8*Y(6,1)*Xi*X(4,1)...
 +X(1,1)*Y(2,1)-2*Y(4,1)*X(5,1)+4*Y(3,1)*Xi*X(2,1)+6*X(8,1)*Y(6,1)*Xi.^2-
3*X(8,1)*Y(5,1)*Xi.^2-4*X(3,1)*Xi*Y(2,1)...
 -3*X(4,1)*Y(1,1)*Xi.^2-4*Y(7,1)*Xi*X(6,1)-3*X(8,1)*Y(7,1)*Xi.^2+4*X(3,1)*Xi*Y(6,1)...
 +4*Y(7,1)*Xi*X(2,1)-4*X(5,1)*Xi*Y(2,1)-
6*X(4,1)*Y(6,1)*Xi.^2+2*Y(4,1)*X(3,1)+2*X(1,1)*Y(7,1)+2*X(1,1)*Y(3,1)...
 -2*X(8,1)*Y(3,1)+2*X(8,1)*Y(5,1)-
2*Y(4,1)*X(1,1)+3*Y(4,1)*X(1,1)*Xi.^2+3*Y(4,1)*X(1,1)*Xi...
 -6*Y(4,1)*X(2,1)*Xi.^2+3*Y(4,1)*X(3,1)*Xi.^2+5*Y(4,1)*X(3,1)*Xi-3*Y(4,1)*X(5,1)*Xi.^2-
5*Y(4,1)*X(5,1)*Xi...
 +6*Y(4,1)*X(6,1)*Xi.^2-3*Y(4,1)*X(7,1)*Xi.^2-3*Y(4,1)*X(7,1)*Xi+4*Y(5,1)*Xi*X(2,1)...
 -4*Y(5,1)*Xi*X(6,1)-3*Y(8,1)*X(1,1)*Xi.^2+5*Y(8,1)*X(1,1)*Xi+6*Y(8,1)*X(2,1)*Xi.^2-
3*Y(8,1)*X(3,1)*Xi.^2+3*Y(8,1)*X(3,1)*Xi...
 +3*Y(8,1)*X(5,1)*Xi.^2-3*Y(8,1)*X(5,1)*Xi-6*Y(8,1)*X(6,1)*Xi.^2+...
 3*Y(8,1)*X(7,1)*Xi.^2-5*Y(8,1)*X(7,1)*Xi+X(3,1)*Y(6,1)-X(7,1)*Y(6,1)-Y(7,1)*X(2,1)-
X(5,1)*Y(2,1)+2*X(8,1)*Y(1,1)...
 +X(7,1)*Y(2,1)+Y(7,1)*X(6,1)+2*X(4,1)*Y(5,1)-2*X(4,1)*Y(7,1)+X(5,1)*Y(6,1)-
2*X(4,1)*Y(3,1))*Eta.^2+(X(1,1)*Y(6,1)+2*X(4,1)*Y(1,1)...
 -5*Y(2,1)*Xi.^2*X(1,1)+5*Y(2,1)*Xi.^2*X(3,1)+3*Y(2,1)*Xi.^2*X(5,1)-
3*Y(2,1)*Xi.^2*X(7,1)+3*X(6,1)*Xi.^2*Y(1,1)...
 -3*X(6,1)*Xi.^2*Y(3,1)-5*X(6,1)*Xi.^2*Y(5,1)+5*X(6,1)*Xi.^2*Y(7,1)...
 +5*X(7,1)*Xi.^2*Y(5,1)+6*X(8,1)*Y(7,1)*Xi+X(3,1)*Y(2,1)-6*Y(1,1)*Xi*X(2,1)-
2*Y(1,1)*Xi*X(6,1)-2*Y(3,1)*Xi*X(6,1)...
 -8*X(4,1)*Y(8,1)+8*X(8,1)*Y(4,1)-8*Y(2,1)*Xi*X(4,1)-
8*Y(2,1)*Xi*X(8,1)+8*X(6,1)*Xi*Y(8,1)...
 +3*X(1,1)*Xi.^2*Y(5,1)-
8*Y(6,1)*Xi*X(8,1)+8*X(2,1)*Xi*Y(4,1)+8*X(2,1)*Xi*Y(8,1)+3*X(2,1)*Xi.^2*Y(7,1)-
5*X(3,1)*Xi.^2*Y(1,1)...
 -5*X(5,1)*Xi.^2*Y(7,1)-Y(3,1)*X(2,1)+2*X(1,1)*Xi*Y(6,1)+2*X(4,1)*Y(1,1)*Xi...
 -
8*X(4,1)*Y(2,1)*Xi.^2+4*X(4,1)*Y(3,1)*Xi.^2+6*X(4,1)*Y(3,1)*Xi+4*X(4,1)*Y(5,1)*Xi.^2+
6*X(4,1)*Y(5,1)*Xi+4*X(4,1)*Y(7,1)*Xi.^2+...
 2*X(4,1)*Y(7,1)*Xi+6*X(5,1)*Xi*Y(6,1)+2*X(7,1)*Xi*Y(2,1)+6*X(7,1)*Xi*Y(6,1)...
 -4*X(8,1)*Y(1,1)*Xi.^2+6*X(8,1)*Y(1,1)*Xi+8*X(8,1)*Y(2,1)*Xi.^2-
4*X(8,1)*Y(3,1)*Xi.^2+2*X(8,1)*Y(3,1)*Xi+2*X(8,1)*Y(5,1)*Xi...
 +Y(3,1)*X(6,1)+2*Y(8,1)*X(7,1)-
Y(1,1)*X(6,1)+5*X(1,1)*Xi.^2*Y(3,1)+5*X(2,1)*Xi.^2*Y(1,1)...
 -5*X(2,1)*Xi.^2*Y(3,1)-3*X(2,1)*Xi.^2*Y(5,1)-
2*X(8,1)*Y(7,1)+Y(1,1)*X(2,1)+2*Y(8,1)*X(5,1)+2*Y(8,1)*X(3,1)+6*X(1,1)*Xi*Y(2,1)...
 -3*Y(6,1)*Xi.^2*X(1,1)+3*Y(6,1)*Xi.^2*X(3,1)+5*Y(6,1)*Xi.^2*X(5,1)-
5*Y(6,1)*Xi.^2*X(7,1)...
```

```
 +2*Y(8,1)*X(1,1)-3*X(5,1)*Xi.^2*Y(1,1)+Y(5,1)*X(2,1)-Y(5,1)*X(6,1)-2*Y(4,1)*X(7,1)-
3*X(3,1)*Xi.^2*Y(7,1)+8*X(6,1)*Xi*Y(4,1)...
 -8*Y(6,1)*Xi*X(4,1)-X(1,1)*Y(2,1)-2*Y(4,1)*X(5,1)+3*X(7,1)*Xi.^2*Y(3,1)-
6*Y(3,1)*Xi*X(2,1)...
 +8*X(8,1)*Y(6,1)*Xi.^2-
4*X(8,1)*Y(5,1)*Xi.^2+6*X(3,1)*Xi*Y(2,1)+4*X(4,1)*Y(1,1)*Xi.^2-6*Y(7,1)*Xi*X(6,1)-
4*X(8,1)*Y(7,1)*Xi.^2+...
 2*X(3,1)*Xi*Y(6,1)-2*Y(7,1)*Xi*X(2,1)+2*X(5,1)*Xi*Y(2,1)-8*X(4,1)*Y(6,1)*Xi.^2-...
 2*Y(4,1)*X(3,1)-2*X(8,1)*Y(3,1)-2*X(8,1)*Y(5,1)-2*Y(4,1)*X(1,1)-4*Y(4,1)*X(1,1)*Xi.^2-
2*Y(4,1)*X(1,1)*Xi+8*Y(4,1)*X(2,1)*Xi.^2-...
 4*Y(4,1)*X(3,1)*Xi.^2-6*Y(4,1)*X(3,1)*Xi-4*Y(4,1)*X(5,1)*Xi.^2-6*Y(4,1)*X(5,1)*Xi...
 +8*Y(4,1)*X(6,1)*Xi.^2-4*Y(4,1)*X(7,1)*Xi.^2-2*Y(4,1)*X(7,1)*Xi-2*Y(5,1)*Xi*X(2,1)-
6*Y(5,1)*Xi*X(6,1)+4*Y(8,1)*X(1,1)*Xi.^2-...
 6*Y(8,1)*X(1,1)*Xi-8*Y(8,1)*X(2,1)*Xi.^2+4*Y(8,1)*X(3,1)*Xi.^2-2*Y(8,1)*X(3,1)*Xi...
 +4*Y(8,1)*X(5,1)*Xi.^2-2*Y(8,1)*X(5,1)*Xi-8*Y(8,1)*X(6,1)*Xi.^2+4*Y(8,1)*X(7,1)*Xi.^2-
6*Y(8,1)*X(7,1)*Xi-X(3,1)*Y(6,1)...
 -X(7,1)*Y(6,1)-Y(7,1)*X(2,1)-X(5,1)*Y(2,1)-
2*X(8,1)*Y(1,1)+X(7,1)*Y(2,1)+Y(7,1)*X(6,1)+2*X(4,1)*Y(5,1)+2*X(4,1)*Y(7,1)...
 +X(5,1)*Y(6,1)+2*X(4,1)*Y(3,1))*Eta+2*X(1,1)*Xi*Y(6,1)-
8*X(2,1)*Xi*Y(6,1)+X(4,1)*Y(1,1)*Xi+2*X(4,1)*Y(2,1)*Xi.^2-...
 X(4,1)*Y(3,1)*Xi.^2-
X(4,1)*Y(3,1)*Xi+X(4,1)*Y(5,1)*Xi.^2+X(4,1)*Y(5,1)*Xi+X(4,1)*Y(7,1)*Xi.^2-
X(4,1)*Y(7,1)*Xi...
 +2*X(5,1)*Xi*Y(6,1)+8*X(6,1)*Xi*Y(2,1)-
2*X(7,1)*Xi*Y(2,1)+2*X(7,1)*Xi*Y(6,1)+X(8,1)*Y(1,1)*Xi.^2-X(8,1)*Y(1,1)*Xi...
 -2*X(8,1)*Y(2,1)*Xi.^2+X(8,1)*Y(3,1)*Xi.^2+X(8,1)*Y(3,1)*Xi-X(8,1)*Y(5,1)*Xi-
2*X(4,1)*Y(2,1)+2*X(4,1)*Y(6,1)...
 -2*X(1,1)*Xi.^2*Y(3,1)+2*X(1,1)*Xi.^3*Y(5,1)-
4*X(1,1)*Xi.^3*Y(6,1)+2*X(1,1)*Xi.^3*Y(7,1)-2*X(1,1)*Xi.^2*Y(7,1)...
 -2*X(2,1)*Xi.^2*Y(1,1)+2*X(2,1)*Xi.^2*Y(3,1)-4*X(2,1)*Xi.^3*Y(5,1)-
2*X(2,1)*Xi.^2*Y(5,1)...
 +8*X(2,1)*Xi.^3*Y(6,1)-4*X(2,1)*Xi.^3*Y(7,1)-2*X(1,1)*Xi*Y(2,1)+2*Y(6,1)*Xi.^2*X(1,1)-
2*Y(6,1)*Xi.^2*X(3,1)...
 +2*Y(6,1)*Xi.^2*X(5,1)-
2*Y(6,1)*Xi.^2*X(7,1)+2*Y(3,1)*Xi*X(2,1)+2*X(8,1)*Y(6,1)*Xi.^2-X(8,1)*Y(5,1)*Xi.^2-...
 2*X(3,1)*Xi*Y(2,1)-X(4,1)*Y(1,1)*Xi.^2-2*Y(7,1)*Xi*X(6,1)-
X(8,1)*Y(7,1)*Xi.^2+2*X(3,1)*Xi*Y(6,1)+2*Y(7,1)*Xi*X(2,1)...
 -2*X(5,1)*Xi*Y(2,1)-2*X(4,1)*Y(6,1)*Xi.^2+Y(4,1)*X(1,1)*Xi.^2-Y(4,1)*X(1,1)*Xi-
2*Y(4,1)*X(2,1)*Xi.^2+...
 Y(4,1)*X(3,1)*Xi.^2+Y(4,1)*X(3,1)*Xi-Y(4,1)*X(5,1)*Xi.^2-
Y(4,1)*X(5,1)*Xi+2*Y(4,1)*X(6,1)*Xi.^2-Y(4,1)*X(7,1)*Xi.^2+...
 Y(4,1)*X(7,1)*Xi+2*Y(5,1)*Xi*X(2,1)-2*Y(5,1)*Xi*X(6,1)-
Y(8,1)*X(1,1)*Xi.^2+Y(8,1)*X(1,1)*Xi+2*Y(8,1)*X(2,1)*Xi.^2-...
 Y(8,1)*X(3,1)*Xi.^2-Y(8,1)*X(3,1)*Xi+Y(8,1)*X(5,1)*Xi.^2+Y(8,1)*X(5,1)*Xi-
2*Y(8,1)*X(6,1)*Xi.^2+Y(8,1)*X(7,1)*Xi.^2-Y(8,1)*X(7,1)*Xi)...
        *(0.5*(Xi-(Xi.*Eta))+0.25*(-(Eta.^2)+Eta)))...
```

```
 +(-2*((2*X(4,1)+X(1,1)+X(7,1)-X(3,1)-X(5,1)-2*X(8,1))*Eta.^2+(2*X(3,1)*Xi+2*X(1,1)*Xi-
X(5,1)-2*X(7,1)*Xi+X(7,1)-2*X(5,1)*Xi...
 -4*X(2,1)*Xi-X(1,1)+4*X(6,1)*Xi+X(3,1))*Eta-2*X(1,1)*Xi-2*X(7,1)*Xi...
 -2*X(3,1)*Xi+4*X(6,1)*Xi+4*X(2,1)*Xi+2*X(8,1)-2*X(4,1)-
2*X(5,1)*Xi)/(2*Y(2,1)*Xi.^2*X(1,1)-2*Y(2,1)*Xi.^2*X(3,1)...
 +2*Y(2,1)*Xi.^2*X(5,1)-2*Y(2,1)*Xi.^2*X(7,1)+4*X(6,1)*Xi.^3*Y(1,1)...
 -2*X(6,1)*Xi.^2*Y(1,1)-
8*X(6,1)*Xi.^3*Y(2,1)+4*X(6,1)*Xi.^3*Y(3,1)+2*X(6,1)*Xi.^2*Y(3,1)-
2*X(6,1)*Xi.^2*Y(5,1)...
 +2*X(6,1)*Xi.^2*Y(7,1)-
2*X(7,1)*Xi.^3*Y(1,1)+2*X(7,1)*Xi.^2*Y(1,1)+4*X(7,1)*Xi.^3*Y(2,1)...
 -2*X(7,1)*Xi.^3*Y(3,1)+2*X(7,1)*Xi.^2*Y(5,1)-
2*Y(8,1)*X(2,1)+2*Y(8,1)*X(6,1)+X(8,1)*Y(7,1)*Xi+2*Y(1,1)*Xi*X(2,1)...
 -2*Y(1,1)*Xi*X(6,1)-2*Y(3,1)*Xi*X(6,1)+2*X(8,1)*Y(2,1)-
2*X(8,1)*Y(6,1)+2*X(2,1)*Xi.^2*Y(7,1)...
 +2*X(3,1)*Xi.^2*Y(1,1)+2*X(3,1)*Xi.^3*Y(5,1)+2*X(3,1)*Xi.^2*Y(5,1)-
4*X(3,1)*Xi.^3*Y(6,1)+2*X(3,1)*Xi.^3*Y(7,1)...
 -2*X(5,1)*Xi.^3*Y(1,1)+4*X(5,1)*Xi.^3*Y(2,1)-2*X(5,1)*Xi.^3*Y(3,1)-
2*X(5,1)*Xi.^2*Y(3,1)...
 -2*X(5,1)*Xi.^2*Y(7,1)+2*Y(4,1)*X(2,1)-2*Y(4,1)*X(6,1)+(2*X(5,1)*Y(1,1)-
2*X(7,1)*Y(5,1)-4*Y(8,1)*X(3,1)...
 +4*X(8,1)*Y(3,1)+2*X(3,1)*Y(1,1)+2*X(3,1)*Y(7,1)+4*X(8,1)*Y(5,1)-2*X(1,1)*Y(5,1)-
4*X(4,1)*Y(7,1)+8*X(4,1)*Y(8,1)+4*Y(4,1)*X(1,1)...
 -4*X(4,1)*Y(1,1)+2*X(5,1)*Y(7,1)-4*Y(8,1)*X(5,1)-2*X(1,1)*Y(3,1)+4*Y(4,1)*X(7,1)-
8*X(8,1)*Y(4,1)-2*X(7,1)*Y(3,1))*Eta.^3+...
 (2*X(5,1)*Y(7,1)-X(1,1)*Y(6,1)+2*X(4,1)*Y(1,1)+3*Y(2,1)*Xi.^2*X(1,1)-
3*Y(2,1)*Xi.^2*X(3,1)...
 -3*Y(2,1)*Xi.^2*X(5,1)+3*Y(2,1)*Xi.^2*X(7,1)+3*X(6,1)*Xi.^2*Y(1,1)-
3*X(6,1)*Xi.^2*Y(3,1)-3*X(6,1)*Xi.^2*Y(5,1)...
 +3*X(6,1)*Xi.^2*Y(7,1)-3*X(7,1)*Xi.^2*Y(1,1)+3*X(7,1)*Xi.^2*Y(5,1)+2*Y(8,1)*X(2,1)-
2*Y(8,1)*X(6,1)...
 +2*X(5,1)*Y(3,1)+5*X(8,1)*Y(7,1)*Xi-X(3,1)*Y(2,1)+4*Y(1,1)*Xi*X(2,1)-
4*Y(1,1)*Xi*X(6,1)-4*Y(3,1)*Xi*X(6,1)...
 +8*Y(2,1)*Xi*X(4,1)+8*Y(2,1)*Xi*X(8,1)-5*X(3,1)*Xi*Y(5,1)-
3*X(3,1)*Xi*Y(7,1)+8*X(6,1)*Xi*Y(8,1)...
 -3*X(1,1)*Xi*Y(5,1)-5*X(1,1)*Xi*Y(7,1)-8*Y(6,1)*Xi*X(8,1)-8*X(2,1)*Xi*Y(4,1)-
8*X(2,1)*Xi*Y(8,1)-2*X(8,1)*Y(2,1)...
 +2*X(8,1)*Y(6,1)-3*X(2,1)*Xi.^2*Y(7,1)+3*X(3,1)*Xi.^2*Y(1,1)-
3*X(3,1)*Xi.^2*Y(5,1)+3*X(5,1)*Xi.^2*Y(3,1)...
 -3*X(5,1)*Xi.^2*Y(7,1)-2*X(3,1)*Y(5,1)+Y(3,1)*X(2,1)-
2*Y(4,1)*X(2,1)+2*Y(4,1)*X(6,1)+4*X(1,1)*Xi*Y(6,1)...
 -3*X(4,1)*Y(1,1)*Xi+6*X(4,1)*Y(2,1)*Xi.^2-3*X(4,1)*Y(3,1)*Xi.^2-
5*X(4,1)*Y(3,1)*Xi+3*X(4,1)*Y(5,1)*Xi.^2+...
 5*X(4,1)*Y(5,1)*Xi+3*X(4,1)*Y(7,1)*Xi.^2+3*X(4,1)*Y(7,1)*Xi+4*X(5,1)*Xi*Y(6,1)-
4*X(7,1)*Xi*Y(2,1)+4*X(7,1)*Xi*Y(6,1)...
```

+3*X(8,1)*Y(1,1)*Xi.^2-5*X(8,1)*Y(1,1)*Xi-6*X(8,1)*Y(2,1)*Xi.^2+3*X(8,1)*Y(3,1)*Xi.^2-
...
3*X(8,1)*Y(3,1)*Xi-2*X(3,1)*Y(1,1)+3*X(8,1)*Y(5,1)*Xi-Y(3,1)*X(6,1)+2*X(4,1)*Y(2,1)-
2*X(4,1)*Y(6,1)...
+2*Y(8,1)*X(7,1)+Y(1,1)*X(6,1)-3*X(1,1)*Xi.^2*Y(3,1)+3*X(1,1)*Xi.^2*Y(7,1)-
3*X(2,1)*Xi.^2*Y(1,1)+3*X(2,1)*Xi.^2*Y(3,1)...
+3*X(2,1)*Xi.^2*Y(5,1)-2*X(8,1)*Y(7,1)-Y(1,1)*X(2,1)-2*Y(8,1)*X(5,1)+2*Y(8,1)*X(3,1)-
4*X(1,1)*Xi*Y(2,1)...
-2*X(7,1)*Y(5,1)-3*Y(6,1)*Xi.^2*X(1,1)+3*Y(6,1)*Xi.^2*X(3,1)+3*Y(6,1)*Xi.^2*X(5,1)-
3*Y(6,1)*Xi.^2*X(7,1)...
-
2*Y(8,1)*X(1,1)+5*X(7,1)*Xi*Y(1,1)+3*X(7,1)*Xi*Y(3,1)+3*X(5,1)*Xi*Y(1,1)+5*X(5,1)*Xi
*Y(3,1)+Y(5,1)*X(2,1)-Y(5,1)*X(6,1)...
-2*X(7,1)*Y(1,1)+2*Y(4,1)*X(7,1)+8*X(6,1)*Xi*Y(4,1)-8*Y(6,1)*Xi*X(4,1)+X(1,1)*Y(2,1)-
2*Y(4,1)*X(5,1)...
+4*Y(3,1)*Xi*X(2,1)+6*X(8,1)*Y(6,1)*Xi.^2-3*X(8,1)*Y(5,1)*Xi.^2-4*X(3,1)*Xi*Y(2,1)-
3*X(4,1)*Y(1,1)*Xi.^2-4*Y(7,1)*Xi*X(6,1)...
-3*X(8,1)*Y(7,1)*Xi.^2+4*X(3,1)*Xi*Y(6,1)+4*Y(7,1)*Xi*X(2,1)-4*X(5,1)*Xi*Y(2,1)-...
6*X(4,1)*Y(6,1)*Xi.^2+2*Y(4,1)*X(3,1)+2*X(1,1)*Y(7,1)+2*X(1,1)*Y(3,1)-
2*X(8,1)*Y(3,1)+2*X(8,1)*Y(5,1)-2*Y(4,1)*X(1,1)...
+3*Y(4,1)*X(1,1)*Xi.^2+3*Y(4,1)*X(1,1)*Xi-
6*Y(4,1)*X(2,1)*Xi.^2+3*Y(4,1)*X(3,1)*Xi.^2+5*Y(4,1)*X(3,1)*Xi...
-3*Y(4,1)*X(5,1)*Xi.^2-5*Y(4,1)*X(5,1)*Xi+6*Y(4,1)*X(6,1)*Xi.^2-3*Y(4,1)*X(7,1)*Xi.^2-
3*Y(4,1)*X(7,1)*Xi+4*Y(5,1)*Xi*X(2,1)...
-4*Y(5,1)*Xi*X(6,1)-3*Y(8,1)*X(1,1)*Xi.^2+5*Y(8,1)*X(1,1)*Xi+6*Y(8,1)*X(2,1)*Xi.^2-...
3*Y(8,1)*X(3,1)*Xi.^2+3*Y(8,1)*X(3,1)*Xi+3*Y(8,1)*X(5,1)*Xi.^2-3*Y(8,1)*X(5,1)*Xi-
6*Y(8,1)*X(6,1)*Xi.^2+3*Y(8,1)*X(7,1)*Xi.^2-...
5*Y(8,1)*X(7,1)*Xi+X(3,1)*Y(6,1)-X(7,1)*Y(6,1)-Y(7,1)*X(2,1)-
X(5,1)*Y(2,1)+2*X(8,1)*Y(1,1)+X(7,1)*Y(2,1)...
+Y(7,1)*X(6,1)+2*X(4,1)*Y(5,1)-2*X(4,1)*Y(7,1)+X(5,1)*Y(6,1)-
2*X(4,1)*Y(3,1))*Eta.^2+(X(1,1)*Y(6,1)+2*X(4,1)*Y(1,1)...
-5*Y(2,1)*Xi.^2*X(1,1)+5*Y(2,1)*Xi.^2*X(3,1)+3*Y(2,1)*Xi.^2*X(5,1)-
3*Y(2,1)*Xi.^2*X(7,1)+3*X(6,1)*Xi.^2*Y(1,1)...
-3*X(6,1)*Xi.^2*Y(3,1)-
5*X(6,1)*Xi.^2*Y(5,1)+5*X(6,1)*Xi.^2*Y(7,1)+5*X(7,1)*Xi.^2*Y(5,1)+6*X(8,1)*Y(7,1)*Xi+
X(3,1)*Y(2,1)...
-6*Y(1,1)*Xi*X(2,1)-2*Y(1,1)*Xi*X(6,1)-2*Y(3,1)*Xi*X(6,1)-
8*X(4,1)*Y(8,1)+8*X(8,1)*Y(4,1)...
-8*Y(2,1)*Xi*X(4,1)-8*Y(2,1)*Xi*X(8,1)+8*X(6,1)*Xi*Y(8,1)+3*X(1,1)*Xi.^2*Y(5,1)-
8*Y(6,1)*Xi*X(8,1)+8*X(2,1)*Xi*Y(4,1)...
+8*X(2,1)*Xi*Y(8,1)+3*X(2,1)*Xi.^2*Y(7,1)-5*X(3,1)*Xi.^2*Y(1,1)-
5*X(5,1)*Xi.^2*Y(7,1)...
-Y(3,1)*X(2,1)+2*X(1,1)*Xi*Y(6,1)+2*X(4,1)*Y(1,1)*Xi-
8*X(4,1)*Y(2,1)*Xi.^2+4*X(4,1)*Y(3,1)*Xi.^2+6*X(4,1)*Y(3,1)*Xi...

+4*X(4,1)*Y(5,1)*Xi.^2+6*X(4,1)*Y(5,1)*Xi+4*X(4,1)*Y(7,1)*Xi.^2+2*X(4,1)*Y(7,1)*Xi+6
*X(5,1)*Xi*Y(6,1)...
+2*X(7,1)*Xi*Y(2,1)+6*X(7,1)*Xi*Y(6,1)-
4*X(8,1)*Y(1,1)*Xi.^2+6*X(8,1)*Y(1,1)*Xi+8*X(8,1)*Y(2,1)*Xi.^2-
4*X(8,1)*Y(3,1)*Xi.^2+...
2*X(8,1)*Y(3,1)*Xi+2*X(8,1)*Y(5,1)*Xi+Y(3,1)*X(6,1)+2*Y(8,1)*X(7,1)-
Y(1,1)*X(6,1)+5*X(1,1)*Xi.^2*Y(3,1)...
+5*X(2,1)*Xi.^2*Y(1,1)-5*X(2,1)*Xi.^2*Y(3,1)-3*X(2,1)*Xi.^2*Y(5,1)-
2*X(8,1)*Y(7,1)+Y(1,1)*X(2,1)+2*Y(8,1)*X(5,1)...
+2*Y(8,1)*X(3,1)+6*X(1,1)*Xi*Y(2,1)-
3*Y(6,1)*Xi.^2*X(1,1)+3*Y(6,1)*Xi.^2*X(3,1)+5*Y(6,1)*Xi.^2*X(5,1)...
-5*Y(6,1)*Xi.^2*X(7,1)+2*Y(8,1)*X(1,1)-3*X(5,1)*Xi.^2*Y(1,1)+Y(5,1)*X(2,1)-
Y(5,1)*X(6,1)-2*Y(4,1)*X(7,1)-3*X(3,1)*Xi.^2*Y(7,1)...
+8*X(6,1)*Xi*Y(4,1)-8*Y(6,1)*Xi*X(4,1)-X(1,1)*Y(2,1)-
2*Y(4,1)*X(5,1)+3*X(7,1)*Xi.^2*Y(3,1)-6*Y(3,1)*Xi*X(2,1)...
+8*X(8,1)*Y(6,1)*Xi.^2-
4*X(8,1)*Y(5,1)*Xi.^2+6*X(3,1)*Xi*Y(2,1)+4*X(4,1)*Y(1,1)*Xi.^2-6*Y(7,1)*Xi*X(6,1)-
4*X(8,1)*Y(7,1)*Xi.^2+...
2*X(3,1)*Xi*Y(6,1)-2*Y(7,1)*Xi*X(2,1)+2*X(5,1)*Xi*Y(2,1)-8*X(4,1)*Y(6,1)*Xi.^2-...
2*Y(4,1)*X(3,1)-2*X(8,1)*Y(3,1)-2*X(8,1)*Y(5,1)-2*Y(4,1)*X(1,1)-4*Y(4,1)*X(1,1)*Xi.^2-
2*Y(4,1)*X(1,1)*Xi+8*Y(4,1)*X(2,1)*Xi.^2-...
4*Y(4,1)*X(3,1)*Xi.^2-6*Y(4,1)*X(3,1)*Xi-4*Y(4,1)*X(5,1)*Xi.^2-6*Y(4,1)*X(5,1)*Xi...
+8*Y(4,1)*X(6,1)*Xi.^2-4*Y(4,1)*X(7,1)*Xi.^2-2*Y(4,1)*X(7,1)*Xi-2*Y(5,1)*Xi*X(2,1)-
6*Y(5,1)*Xi*X(6,1)...
+4*Y(8,1)*X(1,1)*Xi.^2-6*Y(8,1)*X(1,1)*Xi-8*Y(8,1)*X(2,1)*Xi.^2+4*Y(8,1)*X(3,1)*Xi.^2-
2*Y(8,1)*X(3,1)*Xi...
+4*Y(8,1)*X(5,1)*Xi.^2-2*Y(8,1)*X(5,1)*Xi-8*Y(8,1)*X(6,1)*Xi.^2+4*Y(8,1)*X(7,1)*Xi.^2-
6*Y(8,1)*X(7,1)*Xi...
-X(3,1)*Y(6,1)-X(7,1)*Y(6,1)-Y(7,1)*X(2,1)-X(5,1)*Y(2,1)-
2*X(8,1)*Y(1,1)+X(7,1)*Y(2,1)+Y(7,1)*X(6,1)+2*X(4,1)*Y(5,1)+2*X(4,1)*Y(7,1)...
+X(5,1)*Y(6,1)+2*X(4,1)*Y(3,1))*Eta+2*X(1,1)*Xi*Y(6,1)-
8*X(2,1)*Xi*Y(6,1)+X(4,1)*Y(1,1)*Xi+2*X(4,1)*Y(2,1)*Xi.^2-...
X(4,1)*Y(3,1)*Xi.^2-
X(4,1)*Y(3,1)*Xi+X(4,1)*Y(5,1)*Xi.^2+X(4,1)*Y(5,1)*Xi+X(4,1)*Y(7,1)*Xi.^2-
X(4,1)*Y(7,1)*Xi...
+2*X(5,1)*Xi*Y(6,1)+8*X(6,1)*Xi*Y(2,1)-
2*X(7,1)*Xi*Y(2,1)+2*X(7,1)*Xi*Y(6,1)+X(8,1)*Y(1,1)*Xi.^2-X(8,1)*Y(1,1)*Xi-...
2*X(8,1)*Y(2,1)*Xi.^2+X(8,1)*Y(3,1)*Xi.^2+X(8,1)*Y(3,1)*Xi-X(8,1)*Y(5,1)*Xi-
2*X(4,1)*Y(2,1)+2*X(4,1)*Y(6,1)...
-2*X(1,1)*Xi.^2*Y(3,1)+2*X(1,1)*Xi.^3*Y(5,1)-
4*X(1,1)*Xi.^3*Y(6,1)+2*X(1,1)*Xi.^3*Y(7,1)-2*X(1,1)*Xi.^2*Y(7,1)...
-2*X(2,1)*Xi.^2*Y(1,1)+2*X(2,1)*Xi.^2*Y(3,1)-4*X(2,1)*Xi.^3*Y(5,1)-
2*X(2,1)*Xi.^2*Y(5,1)...
+8*X(2,1)*Xi.^3*Y(6,1)-4*X(2,1)*Xi.^3*Y(7,1)-2*X(1,1)*Xi*Y(2,1)+2*Y(6,1)*Xi.^2*X(1,1)-
2*Y(6,1)*Xi.^2*X(3,1)...

```
 +2*Y(6,1)*Xi.^2*X(5,1)-
2*Y(6,1)*Xi.^2*X(7,1)+2*Y(3,1)*Xi*X(2,1)+2*X(8,1)*Y(6,1)*Xi.^2-...
 X(8,1)*Y(5,1)*Xi.^2-2*X(3,1)*Xi*Y(2,1)-X(4,1)*Y(1,1)*Xi.^2-2*Y(7,1)*Xi*X(6,1)-
X(8,1)*Y(7,1)*Xi.^2+2*X(3,1)*Xi*Y(6,1)...
 +2*Y(7,1)*Xi*X(2,1)-2*X(5,1)*Xi*Y(2,1)-2*X(4,1)*Y(6,1)*Xi.^2+Y(4,1)*X(1,1)*Xi.^2-
Y(4,1)*X(1,1)*Xi...
 -2*Y(4,1)*X(2,1)*Xi.^2+Y(4,1)*X(3,1)*Xi.^2+Y(4,1)*X(3,1)*Xi-Y(4,1)*X(5,1)*Xi.^2-
Y(4,1)*X(5,1)*Xi+2*Y(4,1)*X(6,1)*Xi.^2-...
 Y(4,1)*X(7,1)*Xi.^2+Y(4,1)*X(7,1)*Xi+2*Y(5,1)*Xi*X(2,1)-2*Y(5,1)*Xi*X(6,1)-
Y(8,1)*X(1,1)*Xi.^2+...
 Y(8,1)*X(1,1)*Xi+2*Y(8,1)*X(2,1)*Xi.^2-Y(8,1)*X(3,1)*Xi.^2-
Y(8,1)*X(3,1)*Xi+Y(8,1)*X(5,1)*Xi.^2+Y(8,1)*X(5,1)*Xi...
 -2*Y(8,1)*X(6,1)*Xi.^2+Y(8,1)*X(7,1)*Xi.^2-Y(8,1)*X(7,1)*Xi)*...
                        (0.5*(Eta-(Xi.*Eta))+0.25*(-(Xi.^2)+Xi))).^2)...
                                .*(((((0.5*Xi)-(0.5*Xi.*Eta)-(0.25*Eta.^2)+(0.25*Eta))*X(1,1) + ...
                                (-Xi + (Xi.*Eta))*X(2,1) + ...
                                ((0.5*Xi)-(0.5*Xi.*Eta)+(0.25*Eta.^2)-(0.25*Eta))*X(3,1) + ...
                                (0.5-(0.5*Eta.^2))*X(4,1) + ...
                                ((0.5*Xi)+(0.5*Xi.*Eta)+(0.25*Eta.^2)+(0.25*Eta))*X(5,1) + ...
                                (-Xi - (Xi.*Eta))*X(6,1) + ...
                                ((0.5*Xi)+(0.5*Xi.*Eta)-(0.25*Eta.^2)-(0.25*Eta))*X(7,1) + ...
                                (-0.5+(0.5*Eta.^2))*X(8,1)).* ...
                                (((0.5*Eta)-(0.5*Xi.*Eta)-(0.25*Xi.^2)+(0.25*Xi))*Y(1,1) + ...
                                (-0.5+(0.5*Xi.^2))*Y(2,1) + ...
                                ((0.5*Eta)+(0.5*Xi.*Eta)-(0.25*Xi.^2)-(0.25*Xi))*Y(3,1) + ...
                                (-Eta - (Xi.*Eta))*Y(4,1) + ...
                                ((0.5*Eta)+(0.5*Xi.*Eta)+(0.25*Xi.^2)+(0.25*Xi))*Y(5,1) + ...
                                (0.5-(0.5*Xi.^2))*Y(6,1) + ...
                                ((0.5*Eta)-(0.5*Xi.*Eta)+(0.25*Xi.^2)-(0.25*Xi))*Y(7,1) + ...
                                (-Eta + (Eta.*Xi))*Y(8,1))) - ...
                                ((((0.5*Xi)-(0.5*Xi.*Eta)-(0.25*Eta.^2)+(0.25*Eta))*Y(1,1) + ...
                                (-Xi + (Xi.*Eta))*Y(2,1) + ...
                                ((0.5*Xi)-(0.5*Xi.*Eta)+(0.25*Eta.^2)-(0.25*Eta))*Y(3,1) + ...
                                (0.5-(0.5*Eta.^2))*Y(4,1) + ...
                                ((0.5*Xi)+(0.5*Xi.*Eta)+(0.25*Eta.^2)+(0.25*Eta))*Y(5,1) + ...
                                (-Xi - (Xi.*Eta))*Y(6,1) + ...
                                ((0.5*Xi)+(0.5*Xi.*Eta)-(0.25*Eta.^2)-(0.25*Eta))*Y(7,1) + ...
                                (-0.5+(0.5*Eta.^2))*Y(8,1)).* ...
                                (((0.5*Eta)-(0.5*Xi.*Eta)-(0.25*Xi.^2)+(0.25*Xi))*X(1,1) + ...
                                (-0.5+(0.5*Xi.^2))*X(2,1) + ...
                                ((0.5*Eta)+(0.5*Xi.*Eta)-(0.25*Xi.^2)-(0.25*Xi))*X(3,1) + ...
                                (-Eta - (Xi.*Eta))*X(4,1) + ...
                                ((0.5*Eta)+(0.5*Xi.*Eta)+(0.25*Xi.^2)+(0.25*Xi))*X(5,1) + ...
                                (0.5-(0.5*Xi.^2))*X(6,1) + ...
                                ((0.5*Eta)-(0.5*Xi.*Eta)+(0.25*Xi.^2)-(0.25*Xi))*X(7,1) + ...
                                (-Eta + (Eta.*Xi))*X(8,1))));
```

```
Kc1 = (N1N1Xi + N1N1Eta);
```

Example of one liberated heat matrix entry:

```
function Q1 = Heating1(Xi, Eta);

global eq;
load ElementCoord.dat

   X = ElementCoord(((eq*8)-7):(eq*8),1);
   Y = ElementCoord(((eq*8)-7):(eq*8),2);
   Q1 = (0.25*(1-Xi).*(1-Eta).*(-1-Xi-Eta)) .* (((((0.5*Xi)-(0.5*Xi.*Eta)-
        (0.25*Eta.^2)+(0.25*Eta))*X(1,1) + ...
                                (-Xi + (Xi.*Eta))*X(2,1) + ...
                                ((0.5*Xi)-(0.5*Xi.*Eta)+(0.25*Eta.^2)-(0.25*Eta))*X(3,1) + ...
                                (0.5-(0.5*Eta.^2))*X(4,1) + ...
                                ((0.5*Xi)+(0.5*Xi.*Eta)+(0.25*Eta.^2)+(0.25*Eta))*X(5,1) + ...
                                (-Xi - (Xi.*Eta))*X(6,1) + ...
                                ((0.5*Xi)+(0.5*Xi.*Eta)-(0.25*Eta.^2)-(0.25*Eta))*X(7,1) + ...
                                (-0.5+(0.5*Eta.^2))*X(8,1)).* ...
                                (((0.5*Eta)-(0.5*Xi.*Eta)-(0.25*Xi.^2)+(0.25*Xi))*Y(1,1) + ...
                                (-0.5+(0.5*Xi.^2))*Y(2,1) + ...
                                ((0.5*Eta)+(0.5*Xi.*Eta)-(0.25*Xi.^2)-(0.25*Xi))*Y(3,1) + ...
                                (-Eta - (Xi.*Eta))*Y(4,1) + ...
                                ((0.5*Eta)+(0.5*Xi.*Eta)+(0.25*Xi.^2)+(0.25*Xi))*Y(5,1) + ...
                                (0.5-(0.5*Xi.^2))*Y(6,1) + ...
                                ((0.5*Eta)-(0.5*Xi.*Eta)+(0.25*Xi.^2)-(0.25*Xi))*Y(7,1) + ...
                                (-Eta + (Eta.*Xi))*Y(8,1))) - ...
                                ((((0.5*Xi)-(0.5*Xi.*Eta)-(0.25*Eta.^2)+(0.25*Eta))*Y(1,1) + ...
                                (-Xi + (Xi.*Eta))*Y(2,1) + ...
                                ((0.5*Xi)-(0.5*Xi.*Eta)+(0.25*Eta.^2)-(0.25*Eta))*Y(3,1) + ...
                                (0.5-(0.5*Eta.^2))*Y(4,1) + ...
                                ((0.5*Xi)+(0.5*Xi.*Eta)+(0.25*Eta.^2)+(0.25*Eta))*Y(5,1) + ...
                                (-Xi - (Xi.*Eta))*Y(6,1) + ...
                                ((0.5*Xi)+(0.5*Xi.*Eta)-(0.25*Eta.^2)-(0.25*Eta))*Y(7,1) + ...
                                (-0.5+(0.5*Eta.^2))*Y(8,1)).* ...
                                (((0.5*Eta)-(0.5*Xi.*Eta)-(0.25*Xi.^2)+(0.25*Xi))*X(1,1) + ...
                                (-0.5+(0.5*Xi.^2))*X(2,1) + ...
                                ((0.5*Eta)+(0.5*Xi.*Eta)-(0.25*Xi.^2)-(0.25*Xi))*X(3,1) + ...
                                (-Eta - (Xi.*Eta))*X(4,1) + ...
                                ((0.5*Eta)+(0.5*Xi.*Eta)+(0.25*Xi.^2)+(0.25*Xi))*X(5,1) + ...
                                (0.5-(0.5*Xi.^2))*X(6,1) + ...
                                ((0.5*Eta)-(0.5*Xi.*Eta)+(0.25*Xi.^2)-(0.25*Xi))*X(7,1) + ...
                                (-Eta + (Eta.*Xi))*X(8,1))));
```

Example of one surface convection matrix entry:

```
function KhS11 = SurConvS11(Xi);

global esS1;
load S1.dat
if esS1 == 1
   x = S1(((esS1*3)-2):(esS1*3),1);
   y = S1(((esS1*3)-2):(esS1*3),2);
else
   x = S1(((esS1*3)-1-esS1):((esS1*3)-esS1+1),1);
   y = S1(((esS1*3)-1-esS1):((esS1*3)-esS1+1),2);
end

   NS1A = ((-0.5*Xi)+(0.5*(Xi.^2))).*((((((Xi-0.5)*x(1,1))-
               (2*Xi*(x(2,1)))+((Xi+0.5)*x(3,1))).^2) + ...
               ((((Xi-0.5)*y(1,1))-(2*Xi*(y(2,1)))+((Xi+0.5)*y(3,1))).^2)).^(0.5));

   NS1B = ((-0.5*Xi)+(0.5*(Xi.^2)));

   KhS11 = (NS1A.*NS1B);
```

Example of one boundary convection matrix entry:

```
function RhS11 = BoundConvS11(Xi);

global ecS1;
load S1.dat
if ecS1 == 1
   x = S1(((ecS1*3)-2):(ecS1*3),1);
   y = S1(((ecS1*3)-2):(ecS1*3),2);
else
   x = S1(((ecS1*3)-1-ecS1):((ecS1*3)-ecS1+1),1);
   y = S1(((ecS1*3)-1-ecS1):((ecS1*3)-ecS1+1),2);
end

   NS1 = ((-0.5*Xi)+(0.5*(Xi.^2))).*((((((Xi-0.5)*x(1,1))-(2*Xi*(x(2,1)))+
            ((Xi+0.5)*x(3,1))).^2) + ...
            ((((Xi-0.5)*y(1,1))-(2*Xi*(y(2,1)))+((Xi+0.5)*y(3,1))).^2)).^(0.5));

   RhS11 = (NS1);
```

Example of one boundary radiation matrix entry:

```
function RrS11 = BoundRadS11(Xi);

global ewS1;
load S1.dat
if ewS1 == 1
   x = S1(((ewS1*3)-2):(ewS1*3),1);
   y = S1(((ewS1*3)-2):(ewS1*3),2);
else
   x = S1(((ewS1*3)-1-ewS1):((ewS1*3)-ewS1+1),1);
   y = S1(((ewS1*3)-1-ewS1):((ewS1*3)-ewS1+1),2);
end

   NS1 = ((-0.5*Xi)+(0.5*(Xi.^2))).*((((((Xi-0.5)*x(1,1))-(2*Xi*(x(2,1)))+
            ((Xi+0.5)*x(3,1))).^2) + ...
            ((((Xi-0.5)*y(1,1))-(2*Xi*(y(2,1)))+((Xi+0.5)*y(3,1))).^2)).^(0.5));

   RrS11 = (NS1);
```

Example of one surface radiation matrix entry:

```
function RsS11 = SurRadS11(Xi);

global elS1;
load S1.dat
if elS1 == 1
   x = S1(((elS1*3)-2):(elS1*3),1);
   y = S1(((elS1*3)-2):(elS1*3),2);
else
   x = S1(((elS1*3)-1-elS1):((elS1*3)-elS1+1),1);
   y = S1(((elS1*3)-1-elS1):((elS1*3)-elS1+1),2);
end

   NS1 = ((-0.5*Xi)+(0.5*(Xi.^2))).*((((((Xi-0.5)*x(1,1))-(2*Xi*(x(2,1)))+
            ((Xi+0.5)*x(3,1))).^2) + ...
            ((((Xi-0.5)*y(1,1))-(2*Xi*(y(2,1)))+((Xi+0.5)*y(3,1))).^2)).^(0.5));

   NSS = (((((-0.5*Xi)+(0.5*(Xi.^2))))+(((1-(Xi.^2))))+((((0.5*Xi)+(0.5*(Xi.^2)))))).^4);

   RsS11 = (NS1.*NSS);
```

8.3 MICROSOFT VISUAL BASIC CODE

8.3.1 Co-ordinates.xls

Module: Coordinates

```
Sub Coordinates()
'
'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

    Sheets("Sheet1").Select
    Range("A6").Select
        ActiveCell.Value = 0
    Range("A7").Select
    For Counter = 1 To Range("J4") Step 1
        ActiveCell.Value = Counter
        ActiveCell.Offset(1, 0).Range("A1").Select
    Next Counter
    XDist = Range("C2").Value
    YDist = Range("C3").Value
    NoXElements = Range("F2").Value
    NoYElements = Range("F3").Value
    DeltaX = XDist / (2 * NoXElements)
    DeltaY = YDist / (2 * NoYElements)
    NoCoordsX = (2 * NoXElements) + 1
    NoCoordsY = (2 * NoYElements) + 1
    Range("C6").Select
    For i = 1 To NoCoordsY
        For j = 1 To NoCoordsX
            XCoord = (DeltaX * (j - 1))
            YCoord = (DeltaY * (i - 1))
            checki = i Mod 2
            checkj = j Mod 2
                If (checki <> 0) Or (checkj <> 0) Then
                ActiveCell.Offset(1, -1).Range("A1").Select
                ActiveCell.Value = XCoord
                ActiveCell.Offset(0, 1).Range("A1").Select
                Value = YCoord
                End If
        Next j
    Next i

Range("A1").Select
```

```
ActiveSheet.ChartObjects("Chart 1").Activate
        ActiveChart.Axes(xlValue).Select
        With ActiveChart.Axes(xlValue)
                .MinimumScale = 0
                .MaximumScale = YDist
                .MinorUnitIsAuto = True
                .MajorUnitIsAuto = True
                .Crosses = xlAutomatic
                .ReversePlotOrder = False
                .ScaleType = xlLinear
                .DisplayUnit = xlNone
        End With

        ActiveChart.Axes(xlCategory).Select
        With ActiveChart.Axes(xlCategory)
                .MinimumScale = 0
                .MaximumScale = XDist
                .MinorUnitIsAuto = True
                .MajorUnitIsAuto = True
                .Crosses = xlAutomatic
                .ReversePlotOrder = False
                .ScaleType = xlLinear
                .DisplayUnit = xlNone
        End With

Range("A1").Select

End Sub
```

Module: SurfaceCoords

```
Sub SurfaceCoords()
'
'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

   Sheets("Sheet1").Select
   Range("A7:C32000").Select
   Selection.Copy
   Range("A1").Select
   Sheets("S1").Select
   Range("A9").Select
   ActiveSheet.Paste
   Range("B9").Select
   Application.CutCopyMode = False
   Range("A8:C32002").AdvancedFilter Action:=xlFilterCopy, CriteriaRange:= _
```

```
Range("A4:C5"), CopyToRange:=Range("E8:G32002"), Unique:=False
Sheets("Sheet1").Select
NoYNodes = Range("J3").Value
Sheets("S1").Select
Range("F9:G" & NoYNodes + 8).Select
Selection.Interior.ColorIndex = 40
Selection.Copy
Sheets("S1.dat").Select
Range("A1").Select
Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
   :=False, Transpose:=False
Sheets("S1").Select
Range("A1").Select

Sheets("Sheet1").Select
Range("A7:C32000").Select
Selection.Copy
Range("A1").Select
Sheets("S2").Select
Range("A9").Select
ActiveSheet.Paste
Range("C9").Select
Application.CutCopyMode = False
Range("A8:C32002").AdvancedFilter Action:=xlFilterCopy, CriteriaRange:= _
Range("A4:C5"), CopyToRange:=Range("E8:G32002"), Unique:=False
Sheets("Sheet1").Select
NoXNodes = Range("J2").Value
Sheets("S2").Select
Range("F9:G" & NoXNodes + 8).Select
Selection.Interior.ColorIndex = 40
Selection.Copy
Sheets("S2.dat").Select
Range("A1").Select
Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
   :=False, Transpose:=False
Sheets("S2").Select
Range("A1").Select

Sheets("Sheet1").Select
Range("A7:C32000").Select
Selection.Copy
Range("A1").Select
Sheets("S3").Select
Range("A9").Select
ActiveSheet.Paste
Range("B9").Select
```

```
    Application.CutCopyMode = False
    Range("A8:C32002").AdvancedFilter Action:=xlFilterCopy, CriteriaRange:= _
    Range("A4:C5"), CopyToRange:=Range("E8:G32002"), Unique:=False
    Sheets("Sheet1").Select
    NoYNodes = Range("J3").Value
    Sheets("S3").Select
    Range("F9:G" & NoYNodes + 8).Select
    Selection.Interior.ColorIndex = 40
    Selection.Copy
    Sheets("S3.dat").Select
    Range("A1").Select
    Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
       :=False, Transpose:=False
    Sheets("S3").Select
    Range("A1").Select

    Sheets("Sheet1").Select
    Range("A7:C32000").Select
    Selection.Copy
    Range("A1").Select
    Sheets("S4").Select
    Range("A9").Select
    ActiveSheet.Paste
    Range("C9").Select
    Application.CutCopyMode = False
    Range("A8:C32002").AdvancedFilter Action:=xlFilterCopy, CriteriaRange:= _
    Range("A4:C5"), CopyToRange:=Range("E8:G32002"), Unique:=False
    Sheets("Sheet1").Select
    NoXNodes = Range("J2").Value
    Sheets("S4").Select
    Range("F9:G" & NoXNodes + 8).Select
    Selection.Interior.ColorIndex = 40
    Selection.Copy
    Sheets("S4.dat").Select
    Range("A1").Select
    Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
       :=False, Transpose:=False
    Sheets("S4").Select
    Range("A1").Select
    Application.CutCopyMode = False
    Sheets("Sheet1").Select
    Range("A1").Select

End Sub
```

8.3.2 Elements.xls

Module: Elements

```
Sub Elements()
'
'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Sheets("Sheet1").Select

NoXElements = Range("E3").Value
NoYElements = Range("E4").Value
NoXNodes = Range("I3").Value
NoYNodes = Range("I4").Value
Range("A8").Select

For ycount = 1 To NoYElements
        For xcount = 1 To NoXElements
                i = 1 + (2 * (xcount - 1)) + ((2 * NoXNodes - NoXElements) * (ycount - 1))
                j = i + 1
                k = j + 1
                l = k + (NoXNodes - xcount)
                m = k + (2 * NoXNodes - NoXElements)
                n = m - 1
                o = n - 1
                p = l - 1
                ActiveCell.Value = i
                ActiveCell.Offset(0, 1).Range("A1").Select
                ActiveCell.Value = j
                ActiveCell.Offset(0, 1).Range("A1").Select
                ActiveCell.Value = k
                ActiveCell.Offset(0, 1).Range("A1").Select
                ActiveCell.Value = l
                ActiveCell.Offset(0, 1).Range("A1").Select
                ActiveCell.Value = m
                ActiveCell.Offset(0, 1).Range("A1").Select
                ActiveCell.Value = n
                ActiveCell.Offset(0, 1).Range("A1").Select
                ActiveCell.Value = o
                ActiveCell.Offset(0, 1).Range("A1").Select
                ActiveCell.Value = p
                ActiveCell.Offset(1, -7).Range("A1").Select
        Next xcount
Next ycount
```

```
Range("A8:H" & (NoXElements * NoYElements) + 7).Select
Selection.Interior.ColorIndex = 40
Selection.Copy
Sheets("NodesXElements.dat").Select
Range("A1").Select
   Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
   :=False, Transpose:=False
Application.CutCopyMode = False

Sheets("Sheet1").Select
Range("A1").Select

End Sub
```

Module: ElementCoordinates

```
Sub ElementCoordinates()
'
'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

   Sheets("Sheet1").Select
   Dim NoElements As Integer
   NoElements = Range("E5").Value

   For counter = 1 To NoElements
         Sheets("Sheet1").Select
         Range("A8").Select
         ActiveCell.Offset((counter - 1), 0).Range("A1:H1").Select
         Selection.Copy
         Sheets("Sheet2").Select
         Range("D4").Select
         ActiveCell.Offset((counter - 1) * 8, 0).Range("A1").Select
         Selection.PasteSpecial Paste:=xlAll, Operation:=xlNone, SkipBlanks:=False _
            , Transpose:=True
   Next counter

   Range("B4:C" & (NoElements * 8) + 3).Select
   Selection.Interior.ColorIndex = 40
   Selection.Copy
   Sheets("ElementCoord.dat").Select
   Range("A1").Select
   Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
   :=False, Transpose:=False

   Sheets("Sheet2").Select
```

```
    Range("D4:D10000").Select
    Selection.Interior.ColorIndex = 2

    Range("A1").Select
    Sheets("Sheet1").Select
    Range("A1").Select

End Sub
```

Module: BoundaryElements

```
Sub BoundaryElements()
'
'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

    Windows("Elements.xls").Activate
    Sheets("S1").Select
    IntRow = Range("D1").Value
    Sheets("S1").Select
    Range("A5:C5").Select
    Selection.AutoFill Destination:=Range("A5:C" & IntRow + 3), Type:=xlFillDefault
    Range("A1").Select
    Range("G5:I5").Select
    Selection.Copy
        For i = 6 To IntRow + 3
            Range("G" & i).Select
            ActiveSheet.Paste
        Next i
    Range("A1").Select
    Range("G4:I" & IntRow + 3).Select
    Selection.Copy
    Sheets("NodesXElementsS1.dat").Select
    Range("A1").Select
    Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
    :=False, Transpose:=False
    Sheets("S1").Select
    Range("A1").Select

    Windows("Elements.xls").Activate
    Sheets("S2").Select
    IntRow = Range("D1").Value
    Sheets("S2").Select
    Range("A5:C5").Select
    Selection.AutoFill Destination:=Range("A5:C" & IntRow + 3), Type:=xlFillDefault
    Range("A1").Select
```

```
Range("G5:I5").Select
Selection.Copy
    For i = 6 To IntRow + 3
        Range("G" & i).Select
        ActiveSheet.Paste
    Next i
Range("A1").Select
Range("G4:I" & IntRow + 3).Select
Selection.Copy
Sheets("NodesXElementsS2.dat").Select
Range("A1").Select
Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
:=False, Transpose:=False
Sheets("S2").Select
Range("A1").Select

Windows("Elements.xls").Activate
Sheets("S3").Select
IntRow = Range("D1").Value
Sheets("S3").Select
Range("A5:C5").Select
Selection.AutoFill Destination:=Range("A5:C" & IntRow + 3), Type:=xlFillDefault
Range("A1").Select
Range("G5:I5").Select
Selection.Copy
    For i = 6 To IntRow + 3
        Range("G" & i).Select
        ActiveSheet.Paste
    Next i
Range("A1").Select
Range("G4:I" & IntRow + 3).Select
Selection.Copy
Sheets("NodesXElementsS3.dat").Select
Range("A1").Select
Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
:=False, Transpose:=False
Sheets("S3").Select
Range("A1").Select

Windows("Elements.xls").Activate
Sheets("S4").Select
IntRow = Range("D1").Value
Sheets("S4").Select
Range("A5:C5").Select
Selection.AutoFill Destination:=Range("A5:C" & IntRow + 3), Type:=xlFillDefault
Range("A1").Select
```

```
    Range("G5:I5").Select
    Selection.Copy
        For i = 6 To IntRow + 3
            Range("G" & i).Select
            ActiveSheet.Paste
        Next i
    Application.CutCopyMode = False
    Range("A1").Select
    Range("G4:I" & IntRow + 3).Select
    Selection.Copy
    Sheets("NodesXElementsS4.dat").Select
    Range("A1").Select
    Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
    :=False, Transpose:=False
    Sheets("S4").Select
    Range("A1").Select
    Application.CutCopyMode = False

    Sheets("Sheet1").Select
    Range("A1").Select

End Sub
```

9 APPENDIX B: WORKED EXAMPLE

A worked example is presented in a stepwise sequence to illustrate the functionality and procedure required to implement and obtain results from the finite element numerical model.

Example: 2000 mm x 1000 mm concrete element cast directly onto rock

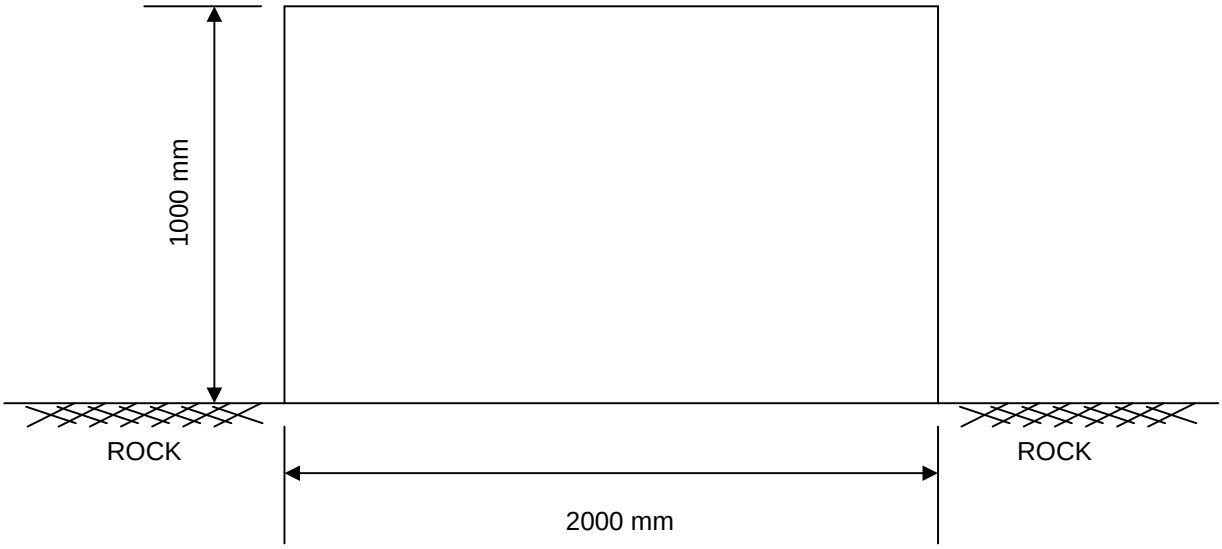


Figure B.1 Worked example: 2000 mm x 1000 mm concrete element

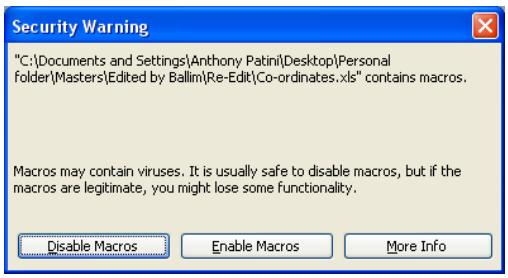
9.1 INPUT DATA GENERATION

This example assumes that the User has a general understanding of the functionality of Microsoft Excel. Microsoft Excel 2003, with Windows XP Professional as the operating system is utilized to describe this example.

All the Microsoft Excel files, data files and Matlab code can be found on the compact disc attached to this research report. It is recommended that the entire content of the compact disc be copied onto the User's PC prior to commencing with this example.

1. Open "Co-ordinate.xls" in Microsoft Excel.

2. Click “Enable Macros” and the spreadsheet will continue to open. Upon opening a security warning may be displayed depending on the PC’s security settings.



3. Click the “Run Macro” button in the Visual Basic toolbar. 
4. Highlight the “ClearCoordinates” Macro, and click on the Run button. This Macro clears all the cells in the worksheets except for the input cells as shown in Table 4.1 and various other calculation cells.

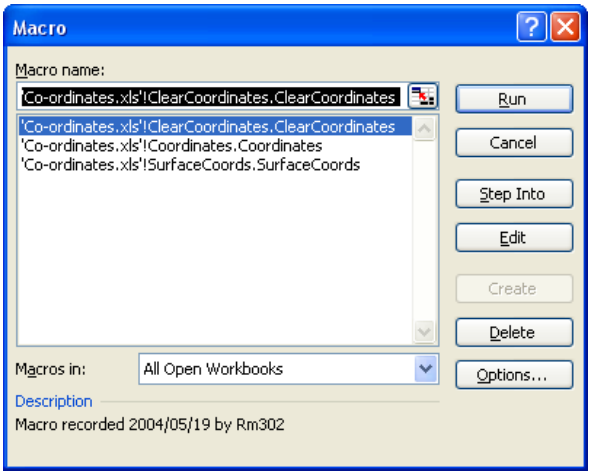
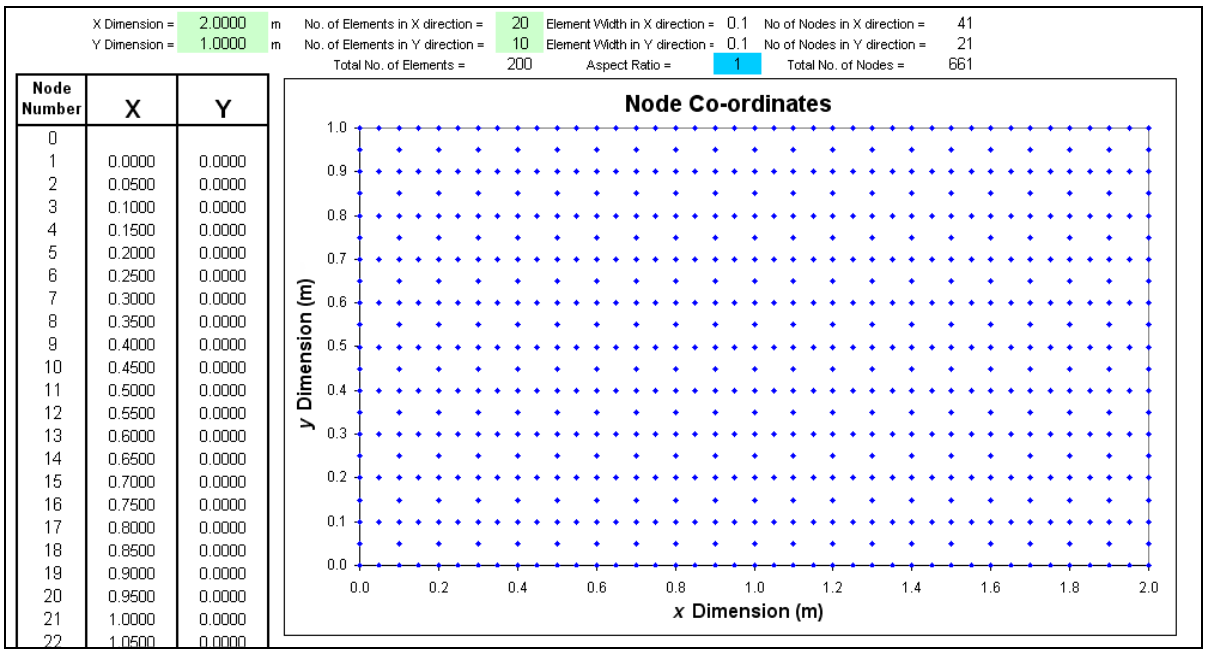


Table 4.1 Input required for the mesh generator

x Dimension =	2 m	Number of elements in the x direction =	20	Element width in the x direction (m) =	0.1	Number of nodes in the x direction =	41
y Dimension =	1 m	Number of elements in the y direction =	10	Element width in the y direction (m) =	0.1	Number of nodes in the y direction =	21
Total number of elements =			200	Aspect ratio =	1	Total number of nodes =	661

5. Enter the required cross-section size (x and y dimension) and the number of elements in the x and y direction. Ensure that the input data is as shown in Table 4.1 or as otherwise required. The User is requested to check and ensure that the aspect ratio is in close proximity to unity.
6. Click the "Run Macro" button 
7. Highlight the "Coordinates" Macro, and click on the Run button. This Macro generates all the nodes and corresponding Cartesian coordinates for the 8 noded quadrilateral isoparametric elements. The worksheet should look as follows:

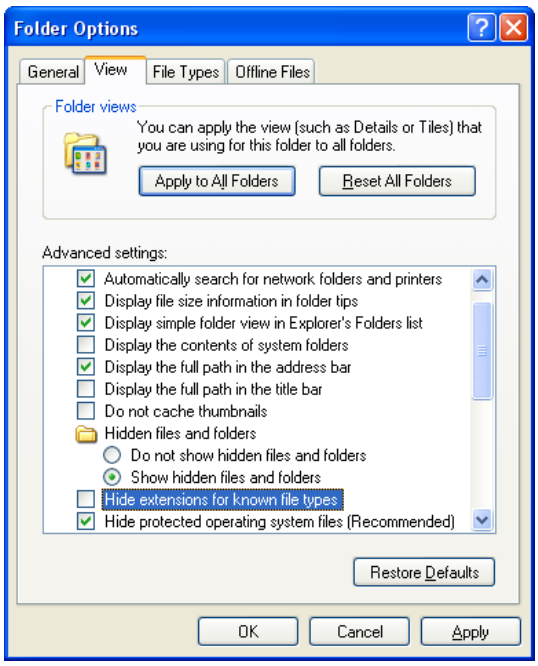


X Dimension =	2.0000	m	No. of Elements in X direction =	20	Element Width in X direction =	0.1	No of Nodes in X direction =	41			
Y Dimension =	1.0000	m	No. of Elements in Y direction =	10	Element Width in Y direction =	0.1	No of Nodes in Y direction =	21			
			Total No. of Elements =	200	Aspect Ratio =	1	Total No. of Nodes =	661			

Node Number	X	Y
0		
1	0.0000	0.0000
2	0.0500	0.0000
3	0.1000	0.0000
4	0.1500	0.0000
5	0.2000	0.0000
6	0.2500	0.0000
7	0.3000	0.0000
8	0.3500	0.0000
9	0.4000	0.0000
10	0.4500	0.0000
11	0.5000	0.0000
12	0.5500	0.0000
13	0.6000	0.0000
14	0.6500	0.0000
15	0.7000	0.0000
16	0.7500	0.0000
17	0.8000	0.0000
18	0.8500	0.0000
19	0.9000	0.0000
20	0.9500	0.0000
21	1.0000	0.0000
22	1.0500	0.0000

8. Click the "Run Macro" button 
9. Highlight the "SurfaceCoords" Macro, and click on the Run button. This Macro generates all the nodes and corresponding Cartesian coordinates for the 3 noded surface quadratic elements. The nodes are assigned the node numbers corresponding to the equivalent nodes of the 8 noded quadrilateral isoparametric elements.
10. *All the data within the worksheets that is highlighted in orange will become the input data necessary for the finite element numerical model written in Matlab.*

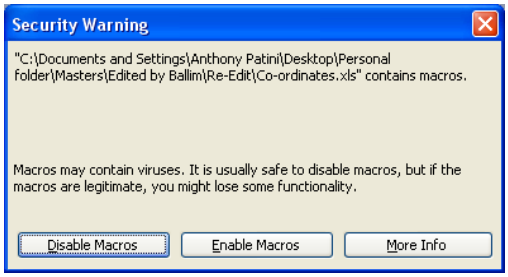
11. Convert worksheets named “S1.dat”, “S2.dat”, “S3.dat” and “S4.dat” into data files for the purpose of importing into Matlab. The process is as follows:
    - Copy the cells already selected in worksheet “S1.dat” (the executed macros select all relevant cells) and paste into a blank workbook. Save the workbook as a text file into the Matlab working folder such that the file name and extension is “S1.txt”. (Note: For Microsoft Excel 2007, save the file as a tab delimited text file.) Confirm all the information boxes during the “Save As” procedure. The Matlab working folder is described as the folder copied onto the User’s PC from the compact disc titled “Matlab”. Close the workbook just saved as a text file and ignore the information box (i.e. Click “No” on the information box requesting whether the User would like to save the file as a Microsoft Excel file).
    - Open the Matlab working folder and change the extension for the file named “S1” from “S1.txt” to “S1.dat”. Confirm the information box. If the User cannot view the file extensions in the Matlab working folder the User’s PC folder settings necessitate changing. This is done by clicking the “Tools” tab button, followed by the “Folder Options” in the dropdown menu. Click on the “View” tab. The active screen should be as shown below.



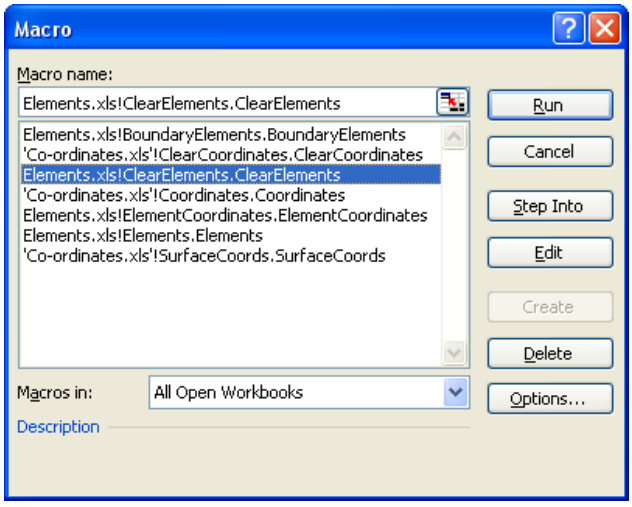
Deselect the “Hide extensions for known file types” and click on “OK”. The User can now continue with the required change to the extension.

- This process is to be repeated for worksheets “S2.dat”, “S3.dat” and “S4.dat”.

12. Open “Elements.xls” in Microsoft Excel. Do not close “Co-ordinates.xls” as links between the two Microsoft Excel files exist.
13. Click “Enable Macros” and the spreadsheet will continue to open. Upon opening a security warning may be displayed depending on the PC’s security settings.



14. Click the “Run Macro” button. 
15. Highlight the “ClearElements” Macro, and click on the Run button. This Macro clears all the cells in the worksheets except for the input cells as shown in Table 4.1 and various other calculation cells.



16. Click the “Run Macro” button 

17. Highlight the “Elements” Macro, and click on the Run button. This Macro generates all the global node numbers for each individual 8 noded quadrilateral isoparametric element from 1 to the “Total number of elements” in Table 4.1. The resulting output of the Macro describes the global assemblage process of these elements.
18. Click the “Run Macro” button 
19. Highlight the “ElementCoordinates” Macro, and click on the Run button. This Macro extracts all the global node numbers and corresponding Cartesian coordinates for each individual 8 noded quadrilateral isoparametric element from 1 to the “Total number of elements” in Table 4.1. This Macro rearranges the computed data into a simpler form for importing into Matlab.
20. Click the “Run Macro” button 
21. Highlight the “BoundaryElements” Macro, and click on the Run button. This Macro generates all the global node numbers for each individual 3 noded surface quadratic element. The resulting output of the Macro describes the global assemblage process of these elements.
22. Convert worksheets named “ElementCoord.dat”, “NodesXElements.dat”, “NodesXElementsS1.dat”, “NodesXElementsS2.dat”, “NodesXElementsS3.dat” and “NodesXElements S4.dat” into data files for the purpose of importing into Matlab. The process is as per step 11.
23. Save and close both spreadsheets.
24. Open a blank Microsoft Excel workbook for the generation of the final two data files. The first file will provide information on the maximum and minimum daily atmospheric temperatures. Column A is standard to all problems and is an incremental 24 hour period. Column B and C contain the minimum and maximum daily temperatures (in degrees Celsius) respectively for the 24 hour period concerned. These daily temperatures are easily available from the local meteorological office. Superfluous data is to be recorded within this workbook such that finite element numerical model written in Matlab does not become unstable or provide errors due to limited input data. The data in this example (as seen in the figure on page B.7) has been generated for a time duration of 1440 hours

(60 days) to ensure sufficient available ambient temperatures. Convert this worksheet into a data file titled “AmbientTemp.dat” through the method described previously.

	A	B	C
1	-24	3	20
2	0	2	21
3	24	3	22
4	48	6	19
5	72	6	22
6	96	7	20
7	120	7	21
8	144	7	13
9	168	6	14
10	192	6	14
11	216	2	18
12	240	3	19
13	264	2	21
14	288	3	20
15	312	2	20
16	336	5	18
17	360	5	20
18	384	5	20
19	408	4	19
20	432	3	20
21	456	5	21
22	480	4	20
23	504	5	19
24	528	4	19
25	552	6	21

AmbientTemp.dat

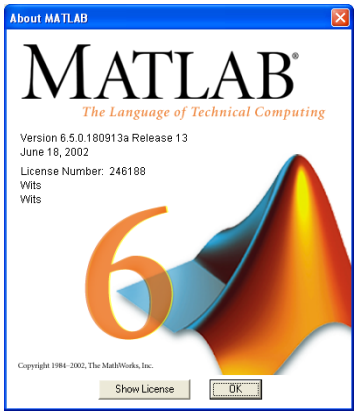
	A	B
1	0	0
2	0.160978	0.43918
3	0.322901	0.832207
4	0.486401	1.095923
5	0.651134	1.224575
6	0.817074	1.241923
7	0.984042	1.18921
8	1.151888	1.103932
9	1.320619	1.011626
10	1.490236	0.926591
11	1.660681	0.855529
12	1.831881	0.800505
13	2.003828	0.759891
14	2.176523	0.729877
15	2.349971	0.706307
16	2.524142	0.686653
17	2.698992	0.670129
18	2.874507	0.656424
19	3.05069	0.644871
20	3.227543	0.634444
21	3.405067	0.624324
22	3.583249	0.614473
23	3.762068	0.605443
24	3.941521	0.597749
25	4.121609	0.591565

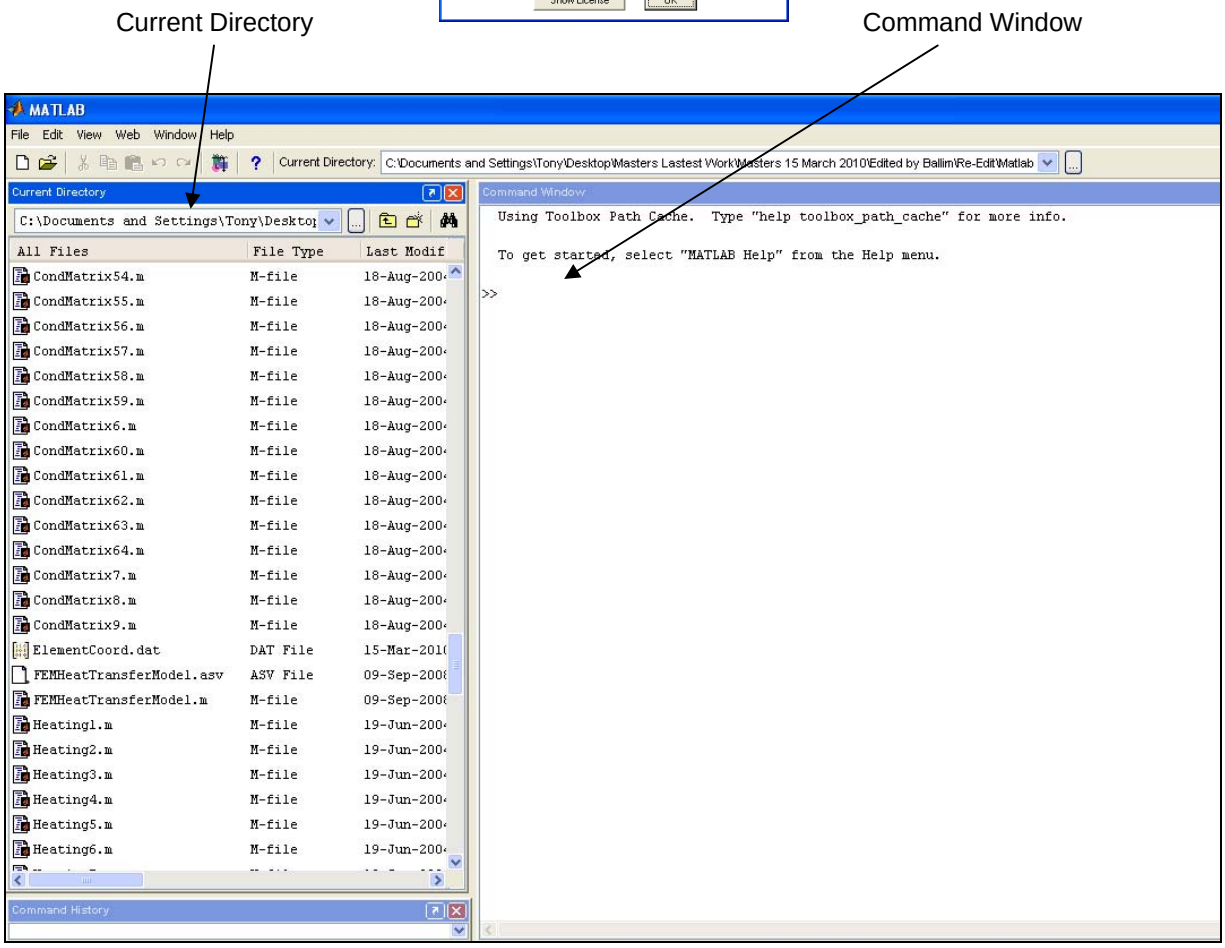
Maturity.dat

The second data file tabulates the heat rate curve as calculated using the experimental data obtained from the adiabatic calorimeter test. The values in column A and column B are the Time (t_{20} hours) and the Maturity Heat Rate (W/kg) respectively. Superfluous data is to be recorded within this workbook such that finite element numerical model written in Matlab does not become unstable or provide errors. Convert this worksheet into a data file titled “Maturity.dat” through the method described previously.

9.2 EXECUTING THE FINITE ELEMENT NUMERICAL MODEL IN MATLAB

Open Matlab and ensure that the “Current Directory” is set to the Matlab working folder. In the “Command Window” type in “FEMHeatTransferModel” followed by pressing the “Enter” key to execute the finite element numerical model. The User is required to have Matlab installed on their PC.





Enter the input data as shown below:

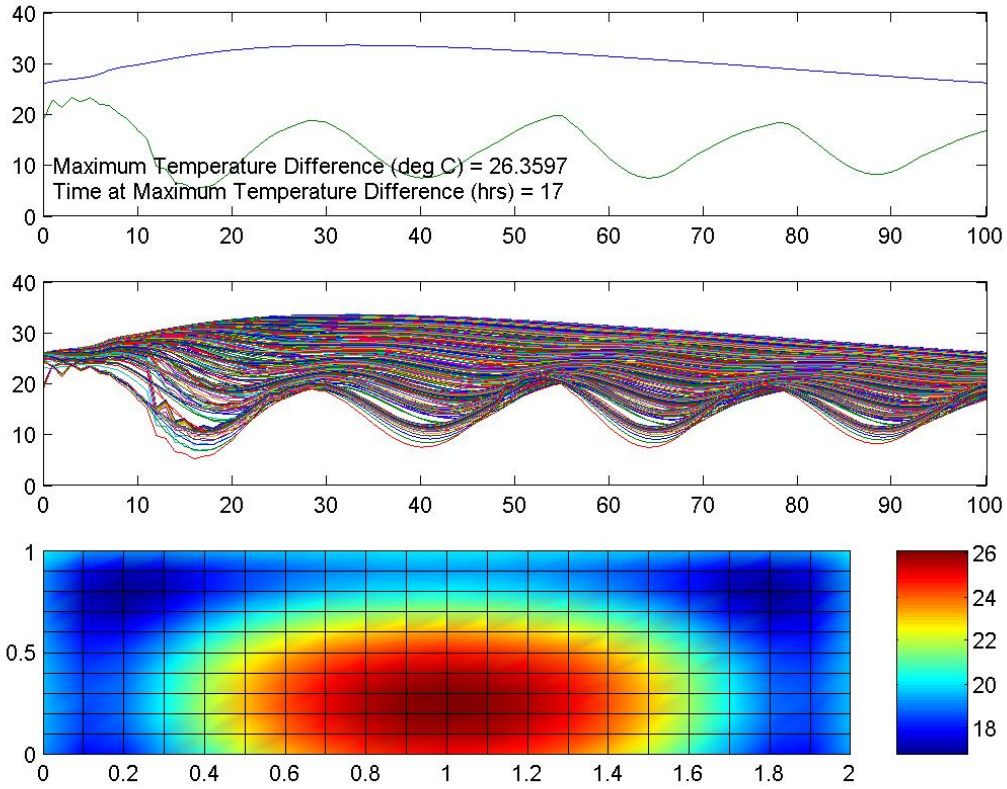
INPUT REQUIRED:

	User Input
NoOfElements = input('Total number of elements = ');	**200**
NoOfNodes = input('Total number of nodes = ');	**661**
NoOfElementsYDirection = input('Total number of elements in the y-Direction = ');	**10**
NoOfElementsXDirection = input('Total number of elements in the x-Direction = ');	**20**
InitialTemp = input('Initial concrete temperature - deg C = ');	**26**
CastTime = input('Time of day when concrete is cast - hrs = ');	**9**
k = input('Thermal conductivity of concrete - W/m.K = ');	**2.2**
rho = input('Concrete density - kg/m^3 = ');	**2500**
cp = input('Concrete specific heat - J/kg.K = ');	**1200**
Ft = input('Formwork removal time - hrs = ');	**12**
hE = input('Convective heat transfer coefficient for exposed concrete surface - W/K.m^2 = ');	**30**
hC = input('Convective heat transfer coefficient for surfaces covered with formwork-W/K.m^2 =');	**5**
kr = input('Thermal conductivity of rock - W/K.m^2 = ');	**1.2**
Sigma = input('Stefan Boltzman constant - W/K^4.m^2 = ');	**5.669e-8**
Emissivity = input('Emissivity of grey concrete surface = ');	**0.9**
tm = input('Time at which the minimum overnight temperature occurs - hrs = ');	**5**
bin = input('Binder content - kg/m^3 = ');	**220**
E = input('Apparent activation energy - kJ/mol = ');	**33.5**
R = input('Universal gas constant - kJ/mol.K = ');	**8.31e-3**
TimeIncrement = input('Time increment - hrs = ');	**1**
FinalTime = input('Time duration - hrs = ');	**100**

The “TimeIncrement” is the time step used in the model and represents the concrete age intervals at which the analysis is undertaken and temperature results reported. The “FinalTime” is the time or concrete age over which the analysis is undertaken.

The program runs automatically following the input of all the above information. The finite element numerical model can take as long as half an hour to solve depending on the speed of the User’s PC.

The results are graphically represented as shown below. The User can extract any nodal temperatures throughout the concrete element (using the Matlab function “dlmwrite”) for comparison purposes if required.



This example is now complete and the User is requested to check whether the “maximum temperature difference” is within specification.