HEAT AND MASS TRANSFER
IN UNSATURATED SOILS
DURING FREEZING

A Thesis
Submitted to the Faculty of Graduate Studies and Research
in Partial Fulfillment of the Requirements
For the Degree of
Master of Science
in the
Department of Civil Engineering
University of Saskatchewan
Saskatoon, Canada

by

Greg P. Newman

Fall, 1995

ABSTRACT

Experimental and field data have shown that large amounts of water can be redistributed from warmer soils to and behind an advancing freezing front. The mechanisms by which this occurs are becoming more understood, but the most appropriate method for analysing these mechanisms is not yet known. Various researchers have developed soil freezing models, but they are all limited to some extent and are not practical tools from a design or predictive modelling perspective. The objective of this research program is to develop unsaturated soil freezing theory from a geotechnical engineering perspective, and to verify the theory by modifying an existing non-freezing soil heat and mass transfer model.

In this study the SoilCover (MEND, 1993) model is modified to verify the theory and numerical solution. SoilCover (MEND, 1993) is a one-dimensional soil heat flow and mass transfer computer model used for designing protective covers over waste rock and tailings. These covers, if they remain saturated, significantly reduce oxygen infiltration into the waste material where it can combine with water to produce acid mine drainage. SoilCover (MEND, 1993) is not capable of modelling through the winter months when upper regions of the covers become subjected to freezing temperatures.

Unique to the modified soil freezing model is the method by which the coupled heat and mass equations are combined and solved. The numerical model uses a single, unique expression which describes the heat flow, mass transfer, and phase change phenomenon in the frozen or partially frozen soil zones. To derive the modified equation, the dependent suction variable in the mass transfer equation is re-written as a function of freezing point depression temperature using a Clapeyron type relationship that is obtained by combining soil freezing curve data with soil water characteristic curve data. The mass transfer

equation is then re-written as a function of change in ice content and substituted into the ice content term of the heat transfer equation. The result is a single combined heat and mass transfer equation with one unknown variable, i.e., temperature. Once new temperatures are solved for over the current time step, suctions and ice contents are computed using back-substitution.

The revised model was verified using laboratory freezing test data collected at the University of Saskatchewan in 1977. During laboratory data modelling of three freezing tests, the average percent difference between measured and computed frost front positions was approximately 6%. The average difference between measured and computed ice contents was approximately 7%, and the average difference between measured and computed liquid water contents was approximately 14%. These discrepancies were primarily due to errors in the estimated and measured soil thermal and hydraulic property functions.

Results of the laboratory data simulations suggest that the permeability versus suction relationship for an unsaturated soil also applies as soil pore-water freezes. This finding is contrary to the findings of other researchers who had to introduce an arbitrary ice impedance factor to make computed and measured ice contents agree. The ice impedance factor has the effect of reducing the permeability by several orders of magnitude as the volumetric pore-ice content increases. In this study, good agreement between computed and measured ice contents was obtained without the use of an impedance factor.

To demonstrate an application of the revised model, a simulation of freezing and thawing in a soil cover system was carried out and compared to field data collected during the winter of 1993 / 1994 at a silver mine near Houston, B.C. For comparisons between the field data and simulations, the soil surface temperature beneath the snow pack had to be estimated as the numerical model does not account for heat and mass flux through snow layers. Daily infiltration during the spring thaw was also estimated based on averaged meteorological data provided by Equity Mine.

ACKNOWLEDGMENTS

I am grateful for the assistance I received from many people during my time as a graduate student. I would like to thank the laboratory technician, Alex Kozlow, and the secretaries Lynn Bantle, Heather Larson and Debbie Pfeifer for helping me with various tasks over the past two years and for being friendly and outgoing. I am grateful for the help I received from Jamshid Yazdani in the geotechnical laboratory. I owe a debt of gratitude to Mike O'kane, a project engineer in the Unsaturated Soils Group, for helping introduce me to the world of geotechnical engineering and for being a good friend. I also thank Placer Dome Canada Inc. for providing the funding necessary to carry out this research.

I would like to thank my supervisor, Professor Ward Wilson, for giving me this project and for encouraging me to carry on when the going was tough. I am also grateful for the advice I received from Professors Lee Barbour and Del Fredlund, and from the members of my examining committee: Professors Don Gray, Bob Besant, and Dennis Pufahl.

I deeply appreciate the love and support I have received throughout my life from my parents, Paul and Edith Newman. Finally, I will be eternally grateful to my wife and colleague, Lori. She sacrificed spending time with our newborn son, Ben, to return as a full time funded graduate student so that we could both afford this education.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 Background

Researchers have been interested in analysing freezing processes in soils throughout this century. In the 1970's oil companies proposed construction of chilled gas pipelines stretching from northern permafrost regions to populated areas in the south. Oil companies were concerned about the effects of frost damage on the pipelines and it was this interest, and the accompanying influx of research dollars, which facilitated means for predicting frost heave phenomenon.

Early research focused exclusively on thermal processes in freezing soils, but in the 1970's it became clear that freezing and thawing analyses must include both heat flow and mass transfer. Throughout the 1970's and 1980's researchers developed numerous heat flow and mass transfer models for freezing soils. While engineers were interested in frost heave predictions, soil scientists focused on predicting temporal winter temperature and moisture content profiles in agricultural soil. Various theoretical approaches were tried with varying degrees of success. A practical model was never developed for general use by practicing engineers or soil scientists.

During this period it became clear that certain problems were common to all the proposed models. These problems related to the relationship between pore-water pressures and temperature in different soil types; the method by which the unfrozen water content versus temperature function was obtained; and the applicability of the water permeability versus suction (or water content) function for water in partially frozen soils. The Clapeyron equation relates a change in pressure between any two phases of a substance (i.e., liquid - vapour, or liquid - solid) to the change in temperature of the system. It has been used with limited success to couple the heat flow and mass transfer equations for freezing soils, but research has shown that the Clapeyron equation only applies in certain circumstances (i.e., if the soil water is wholly capillary or, wholly adsorptive). The Clapeyron equation has also been used as a tool to predict the amount of unfrozen water in a frozen soil as a function of temperature below freezing. Finally, it is known that the water coefficient of permeability in a partially frozen soil is reduced by pore-ice build up, but there has been disagreement about how to predict the new permeability in the frozen zones. As a result, some researchers have used an empirical impedance factor to calibrate the permeability functions in their models.

1.2 Objectives of Research Program

The two primary objectives of this research program are as follows:

1. to develop unsaturated soil freezing theory from a geotechnical engineering perspective, and
2. to devise a numerical technique for verifying this theory by modifying an existing non-freezing soil heat flow and mass transfer model.

In this study, the existing numerical model SoilCover (MEND, 1993) was modified to verify the theory and numerical solution technique. The non-freezing SoilCover (MEND, 1993) model was developed as a design tool for engineers working on soil cover systems over acid generating mine waste rock and tailings. SoilCover (MEND, 1993) uses Darcy's and Fick's laws to describe the flow of liquid water and vapour in the soil below the soil - atmosphere boundary. A modified Penman formulation (Wilson, 1990) computes the actual evaporation from the soil surface and thus couples the soil model with the atmosphere. The unmodified version of SoilCover (MEND, 1993) is not capable of modelling soil temperature and suction profiles if the ground surface temperature drops to 0°C or colder. Therefore, long term predictive modelling is difficult because there is no continuity between summer and winter seasons.

1.3 Organization of Thesis

Problems associated with freezing soil analyses are discussed in detail in the literature review chapter (i.e., Chapter 2). Chapter 2 also discusses the mechanisms of heat flow and mass transfer in freezing and non-freezing soils, and the methods by which the soil thermal and hydraulic properties can be computed. Heat flow and mass transfer processes in unsaturated soils are fairly well understood for the non-freezing case, but analysis of the phenomenon is more complex if part of the pore space is occupied by ice. Where most of the soil thermal and hydraulic properties are functions of changing water contents in the unfrozen case, they become functions of changing water and changing ice contents if the soil is frozen. A brief discussion of the various types of soil freezing models which have been developed in the last twenty years is also presented in Chapter 2.

The theoretical development chapter of this thesis (i.e., Chapter 3) presents the coupled heat flow and mass transfer equations used in the non-freezing version of SoilCover (MEND, 1993). Following this, the modified heat flow and mass transfer equations which include phase change phenomenon are derived. The modified theory is uniquely incorporated into the existing model in such a way that it is consistent with the model's finite element method formulation.

Chapter 4 introduces the laboratory data model verification program. Initial verification of the revised model was carried out using laboratory freezing data obtained by Jame (1977) at the University of Saskatchewan. During the laboratory data modelling program, the sensitivity of computed moisture and temperature profiles to slight changes in soil hydraulic properties was tested.

In Chapter 5, the results of the laboratory verification program are presented. Chapter 6 discusses the modelling results and probable reasons for any discrepancy between computed and measured results are given. In addition, general comments about the advantages and limitations of the modified soil freezing model are presented. Concluding comments are given in Chapter 7.

Appendix A presents a field application of the revised model and compares simulation results with freezing and thawing data collected in the field. The field data was obtained by O'kane (1995) from an instrumented clay - till cover over mine waste rock at a silver mine, near the town of Houston, in the interior of British Columbia. Appendices B through G include the revised computed code, support files, and sample input files usd in the field data modelling.

CHAPTER 2
LITERATURE REVIEW

2.1 Introduction

From a historical perspective, the conceptualizations of, and subsequent attempts at analyzing freezing and thawing processes in soil dates back to lectures given by Stefan and Neumann in the late 1800's (Jumikis, 1966). They presented exact solutions to predict the depth of frost penetration as a function of time. Their methods were based on questionable assumptions about the freezing temperature of pore-water, the shape of the temperature profile below the frost line, the thermal properties of soil at temperatures above and below freezing, and the significance of mass transfer mechanisms. In addition, their solutions were given for a unique set of boundary conditions which seldom exist in nature.

Relatively little research related to soil freezing was carried out in the first half of this century. Bouyoucos (1920) demonstrated that a certain amount of unfrozen water exists in frozen soils. He also concluded that the unfrozen water exists because of interactions between the mineral matrix and pore-water. In 1929, Taber showed that a major cause of volume changes in frozen soil was the formation of segregated ice. While researchers were aware that pore-water in soils freezes at temperatures below 0°C, it was Fisher (1924) who presented a detailed discussion on the freezing of water in capillary systems which indicated the dominant influence on freezing point depression was not salt concentration, but the forces by which water was held in the soil pores. Schofield (1935) presented an underived freezing point depression calculation as a function of negative pore-water pressure, and in 1935, Beskow demonstrated that negative pore-water pressures also develop within a fine-grained soil during freezing.

In the post World War II era a large increase in freeze/thaw and permafrost research was initiated. Large scale engineering projects throughout the world required an accurate means for predicting freeze / thaw behaviour and its effect on engineered structures. Kemper (1960) showed that water transfer in unsaturated soils takes place in thin liquid films which exist between adjacent soil particles. Dirksen and Miller (1966) showed that the water transfer is altered by the presence of pore-ice, and Jumikis (1966), Hoekstra (1966) and others, showed that freezing and thawing in soils involves significant mass transfer processes in addition to the more obvious heat transfers.

During the late 1960's and throughout the 1970's a considerable amount of research was carried out for freezing and thawing soils. During this period there was little consensus among researchers about how to analyse the physical processes occurring in a freezing soil. As a result, numerous theories were proposed which attempted to describe the physics of soil freezing. Numerous models were also developed which attempted to provide engineers and soil scientists with a means of mathematically analysing freeze / thaw behaviour. Works by Harlan (1973), Guymon and Luthin (1974), Jame (1977), Konrad and Morgenstern (1990), Nixon (1992) and others have tried to present mechanistic models for freezing behavior in fine grained soils with or without frost heave. Flerchinger (1989) presents a simultaneous heat and water model for a snow-residue-soil system using conventional heat flow and mass balance theory. His model includes osmotic effects, but neglects frost heave.

The bulk of this literature review will focus on research conducted from the mid 1960's to the present. This research has, in general, dealt with the following:

- understanding the physical processes which take place during freezing,

- quantifying the amount of unfrozen water in frozen soils as a function of temperature or suction,
- determining the thermal and hydraulic properties of unsaturated soils subjected to freezing and thawing for use in analytical models, and
- developing numerical models to analyse the coupled heat and mass transfer processes.

These will be discussed in turn.

2.2 The Physical Processes Occurring During Soil Freezing

A soil system consists of a mineral matrix whose voids may contain air, water, water-vapour, ice, and various solute solutions. The soil itself is physically discontinuous from the atmosphere at its surface, so it is common when analyzing soil behaviour to define the surface as a natural boundary. Although the soil appears discontinuous with the atmosphere at this point, it is the energy and mass fluxes across the soil - atmosphere boundary which cause changes in the stress states of the soil.

When the temperature at the soil surface drops, a thermal gradient develops between the cold surface and the warmer soils below. A transient heat flow is initiated which includes conductive, convective, and radiative heat transfers. If the boundary temperature drops below the freezing point of the pore-water, some of the pore-water will freeze and release latent heat into the system. As the pore-water freezes the amount of water remaining in the liquid state is reduced and negative pore-water pressures (or suctions) are induced, or increase in the case of an initially unsaturated soil. In response to the pressure and temperature gradient, moisture, if available, flows in its liquid and vapour phase from lower regions in the soil to and beyond the advancing freezing front, where it freezes and releases more latent heat. If the conditions are right (i.e., soil type, overburden pressure, water content etc.), ice lenses may form and result in heave of the soil surface. The frost front advance continues until the frozen zone can no longer remove all the latent energy released by the phase change at the frost front. Liquid flux will continue throughout the system as long as the water permeability is sufficiently high or until the pores become completely blocked by ice, at which point liquid flux will be shut down behind the frost front.

Most researchers today would agree that the previous paragraph describes the basic processes which take place during soil freezing. The problem with this description of the physical processes is that it is too simplified. A rigorous analysis of this type of coupled heat and mass transfer problem requires a deeper understanding of the individual physical processes and their significance on the final analytical solutions.

2.2.1 Moisture Transfer Mechanisms

As discussed in the previous section, moisture transfer in freezing / thawing soils can occur in the liquid or vapour phase. The type of liquid transfer depends on the soil geometry, degree of saturation, and availability of water. While researchers have been aware of vapour transfer processes in non-freezing soils for the past 50 years (Smith, 1939), the significance of vapour transfer in freezing soils subjected to different types of boundary conditions is still under debate.

2.2.1.1 Vapour Transfer

The driving force for vapour transfer is the gradient between the partial vapour pressure of pore-water in the warmer soils at depth and the partial vapour pressure of pore-water in the upper regions of soil, just below the deepest point of ice formation. Thus, vapour transfer takes place towards the colder regions, or along the drop of the thermal gradient. Jumikis (1966) mentions the importance of vapour transport in soils with relatively large void sizes, especially in the case where there is no continuous liquid phase. He acknowledges that vapour diffusion also takes place in soils with particles coated by moisture films, but he adds that it is very difficult to analyse this type of diffusion due to the difficulty in expressing the geometry of the voids and surface topography of the soil particles.

The relevance of vapour transfer in well graded soils is considered by Gray et al., (1985). While studying over-winter soil moisture changes, Gray observed post-winter field moisture conditions which suggested a significant amount of vapour transfer had occurred. By comparing the energy and mass balances between two different sites on the Canadian Prairies, Gray was able to back calculate what appeared to be a substantial vapour transfer event.

Harlan (1973) was one of the first investigators to model coupled heat and mass transfer in freezing soils. His model was developed on the assumption that vapour transfer has a negligible effect on the net mass transfer. The assumption made by Harlan (1973) may have some validity for the tests he conducted using Yoho Clay soil (with relatively low porosity), but his modelling of freezing in Del Monte Sand most likely had error introduced by omission of vapour transport. Harlan (1973) was not able to present a comparison of laboratory data and analytical results so it is hard to make quantitative comments about the validity of his assumptions. Many subsequent researchers followed Harlan's lead and omitted the vapour transport mechanism. in their freezing models.

Philip and de Vries (1957) presented heat and mass transfer equations for porous materials which included vapour transfer under non-freezing conditions. Their approach was limited from a geotechnical engineering perspective. They assumed volume change did not occur, they neglected liquid flow resulting from changes in total stress, and they assumed liquid flow was in response to changes in volumetric water content, and not hydraulic head.

Dakshanamurthy and Fredlund (1981) presented un-coupled simultaneous heat and mass transfer equations which included air, water, vapour and heat flow in non-freezing, unsaturated swelling soils. Wilson (1990) used simplified forms of these relationships and presented two coupled heat and mass transfer partial differential equations with hydraulic head and temperature as dependent variables. Wilson's equations are the basis for this research program which expands the heat and mass transfer model to include freezing and thawing in soils. Wilson's (1990) coupled heat and mass transfer equations are discussed in detail in Chapter 3.

2.2.1.2 Liquid Transfer

Soil physicists describe the transport of water through soils as bulk flow or film-capillary flow (Jumikis, 1966). In a saturated soil, all of the voids are filled with water and the flow is considered bulk fluid flow. In unsaturated soils, water is transported by film-capillary flow mechanisms. In either case, and for saturated or unsaturated conditions, water flows in response to changes in hydraulic head.

The flow of liquid water through saturated soils is commonly described using Darcy's law, where the rate of flow through a soil mass is proportional to the hydraulic head gradient. The coefficient of proportionality between the flow rate and the hydraulic head gradient is called the coefficient of permeability. This coefficient is relatively constant for any specific saturated soil. Darcy's law also applies to flow of water through unsaturated soils. In this case the coefficient of permeability is not constant, but a function of water content or matric suction. Matric suction is defined as the difference between pore-air and pore-water pressure (Fredlund and Rhardjo, 1993). The coefficient of permeability is generally a function of any two of the following three volume-mass soil properties: the degree of saturation, the void ratio or the water content (Fredlund and Rhardjo, 1993).

Water can be considered to flow only through pores that contain water. As a result, the air filled pores are non-conductive channels to the flow of water (Fredlund and Rhardjo, 1993). Childs (1969) speculated that the air-filled pores in an unsaturated soil behave similarly to the solids which make up the soil. This is similar to the assumption made by Harlan (1973) that the suction - permeability relationship of a partially frozen soil is the same as that of the unfrozen, unsaturated soil because the frozen water is treated as part of the solid matrix of the soil. Experiments have been conducted to verify Darcy's law for water flow through unsaturated soils, but experimental data supporting the Harlan (1973) assumption has been lacking. Results of numerical modelling by Jame (1977), Taylor and Luthin (1978) and others did not verify the Harlan (1973) hypothesis. However, results of this current study suggest his assumptions may have some validity. This will be discussed in section 2.4.3 and in Chapter 6.

2.2.1.3 Moisture Zones in Frozen Soils

During the 1960's, Jumikis (1966), Dirksen and Miller (1966), Hoekstra (1966) and others imagined that a freezing soil consisted of three zones: a frozen zone, a frost front, and unfrozen soil. The frost front is the point of farthest frost advance into the soil. Miller (1972) first made reference to a thin zone of low permeability frozen soil which lies between an ice lens (if present) and the unfrozen soil called the frozen fringe. Harlan (1973) makes reference to an additional zone "in close proximity to the freezing front" where a large redistribution of water takes place. Figure 2.1 illustrates these zones.

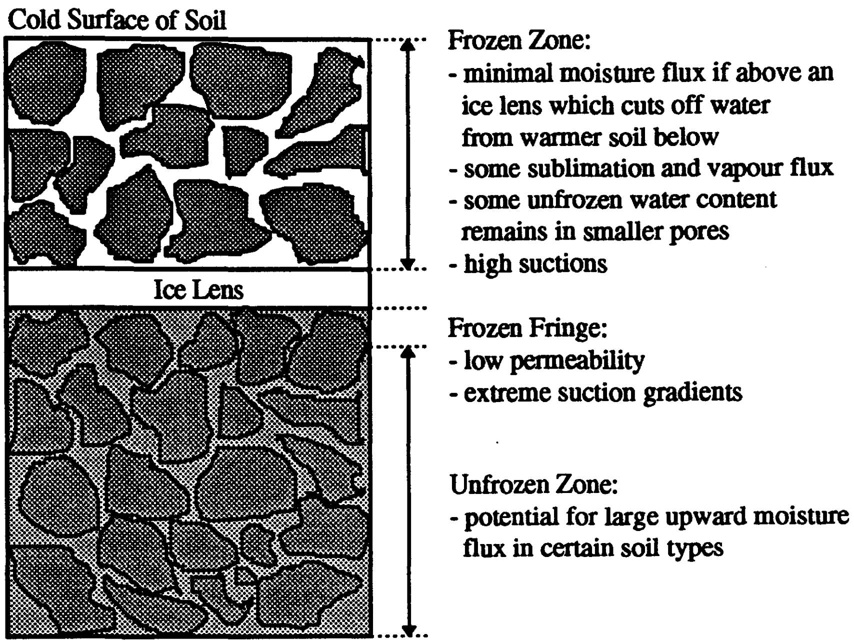


Figure 2.1 Definition of Various Zones in a Frozen Soil Based on Miller (1972) and Harlan (1973)

Ice lenses are thin plates of ice segregated from the soil matrix. The process by which they are formed has been the topic of great debate over the years. Ice lensing results in frost heave and considerable damage to surface structures. Ice lensing and heaving was initially studied by Taber (1929) and Beskow (1935) earlier in the century. More recently, Miller (1978), Konrad and Morgenstern (1980) and Nixon (1991) discuss the process in some detail and how it can be analysed. Ice lensing is significant because it can cause damage to surface structures and because it affects the moisture flow conditions in the frozen soil above the lens. It is generally agreed that during freezing, liquid moisture flux in the frozen zone above an ice lens is negligible because the ice lens acts as a barrier to liquid flow from below. Ice lensing is not considered in the computer model developed in Chapter 3.

Dirksen and Miller (1966) and Hoekstra (1966) conducted experiments which revealed a large moisture flux from the unfrozen soil to the freezing front. They also noted that the ice content behind the frost front increased with time. In order to explain this phenomenon, vapour transfer analyses were conducted. From these studies, it became evident that vapour flow alone could not explain the quantity of water in the frozen soil. Hoekstra (1966) observed that, for his test conditions, the magnitude of moisture flow in the frozen and unfrozen zones were similar. The significance of this finding is that it becomes evident liquid flow may continue in the frozen zone as long as a liquid source is available or until the permeability becomes sufficiently low to effectively prevent further moisture movement (Konrad and Morgenstern, 1980).

2.2.2 Heat Transfer Mechanisms

The three mechanisms for heat flux in soils are conduction, convection, and radiation. Latent heat of phase change introduces sources or sinks for heat flux. Heat conduction occurs through the soil particles, pore-water, ice, vapour, and air. In air and vapour it occurs by a process of collision between molecules and subsequent increase in kinetic energy. A similar process occurs in water, but additional heat is transferred through making and breaking hydrogen bonds. In the soil solid and in ice, conduction is the most efficient form of heat transfer as energy is transferred through vibration of adjacent atoms in the solid lattice structure.

Convection occurs both by molecular motion (diffusion) and heat transfer through bulk fluid flow (advection). Free convection is a mass transport phenomenon which results from density gradients that are often accompanied by temperature gradients. Thus, fluid of higher density flow towards the lower density fluid, advecting heat energy as it moves. Forced convection results from pressure differences similar to those present in groundwater.

Thermal radiation is energy emitted by matter that is at a finite temperature. It is transported by electromagnetic waves and does not require the presence of a material medium. Heat transfer by radiation in the soil pores is a function of temperature, pore geometry and water content. The effect of this type of heat transfer decreases rapidly with decreasing void size, increasing water content, and decreasing temperature (Lunardini, 1991). Thus, if radiation is only relevant to low water content, large pore size, and high temperature soils, then it can be neglected in the case of freezing soils.

Latent heat is released or absorbed when water changes phase. The latent heat of fusion (i.e., water to ice) is equal to 334 kJ/kg of water, the latent heat of sublimation is equal to 2709 kJ/kg, and the latent heat of vapourization is 2375 kJ/kg. The latent heat input or removed from a system has a significant effect on the rate of penetration of the freezing or thawing front. Convective heat transfer is considered in the next section.

2.2.2.1 The Relevance of Convective Heat Transfer

A majority of analytical soil freezing models make the assumption that heat transfer by convection is negligible. Harlan (1973) and Guymon and Luthin (1974) included the convective component in their analysis, while Nixon (1975) and Taylor and Luthin (1978) found that heat transfer by convection was two to three orders of magnitude lower than that due to conduction and they omitted it. Jame and Norum (1980) used the Harlan (1973) approach without the convection term and obtained quite reasonable results modelling freezing of a fine silica flour over a period of 72 hours. Flerchinger (1987) included the convective term in his field modelling of winter freezing and thawing. Tao and Gray (1994) also included convection in their model predicting snow melt infiltration into frozen soils. It appears that the inclusion or omission of the convective term depends on the boundary conditions of the system being modeled and on the permeability of the soil. In general, convective heat transfer should be included when modelling high permeability soils, especially where there is potential for large moisture fluxes.

2.3 Unfrozen Water in Frozen Soils

Since Bouyoucous (1920) first showed that some part of water in a clay-water mixture remains unfrozen at temperatures as low as -78°C, many researchers have tried to explain, in physical and theoretical terms, the processes by which this phenomenon occurs (Williams, 1964; Williams, 1966; Nersesova and Tsytovich, 1966; Miller, 1966; Takagi, 1966; Low et al, 1968; Jame, 1972; Tice et al, 1976; Black and Tice, 1989). Since direct measurement of freezing point depression is difficult, the overall objective of their research was to develop a tool for predicting the freezing temperatures based on some easily measured soil properties. Regardless of the experimental methods used to obtain this data, it is common practice to report the freezing temperature as a function of the unfrozen water content. The converse of this relationship is the unfrozen water content versus sub-zero temperature function, which is a very useful function to have when modelling freezing in soils.

There appear to be two theories about the validity of unfrozen water content data. Nersesova (1966), Tice et al, (1966), Jame (1972) and others indicate that the unfrozen water content data for a given soil is independent of initial water content (or degree of saturation) and mainly a function of sub-zero temperature. Therefore, one soil freezing curve is valid for the entire range of water contents that a particular soil could have. Yong (1965), and Lange and McKim (1963), however, suggest that the unfrozen water content is dependent on both the initial degree of saturation and sub-zero temperature. Figure 2.2 shows soil freezing curves developed by Yong (1965), and Figure 2.3 shows soil freezing curves developed by Nersesova and Tsytovich (1965).

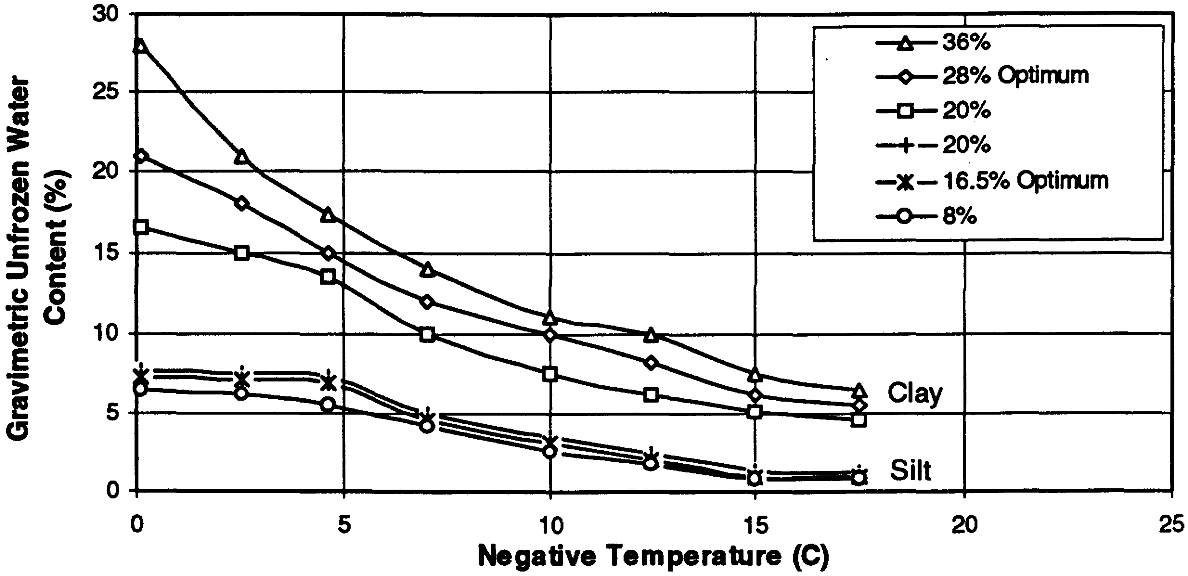


Figure 2.2 Temperature and Unfrozen Water Content of Silt and Clay for Various Initial Water Contents (after Yong, 1965)

In Figure 2.2 the experimental soil freezing curves are given for both silt and clay samples <u>prepared</u> at different initial moisture contents. Thus, each of the six curves in Figure 2.2 represent the soil freezing curve of a <u>unique</u> soil sample. Observation of Figure 2.2 reveals that samples prepared at higher water contents retain more unfrozen water at lower freezing temperatures than samples prepared at lower water contents.. In Figure 2.3, experimental soil freezing curves are given for five different materials ranging from clay to quartz sand. Although no mention is made of the initial moisture conditions (i.e., at which the samples were prepared), it is obvious that each curve is only valid for the specified soil type with a similar stress history. The apparent contradiction between the two figures is a result of how they are interpreted. In Figure 2.2, initial water content refers to the water content at which the sample was compacted. While in Figure 2.3, the initial water content is not the water content at which the material was compacted, but the water content present in the soil when freezing first begins. In fact, both figures show valid soil freezing curves.

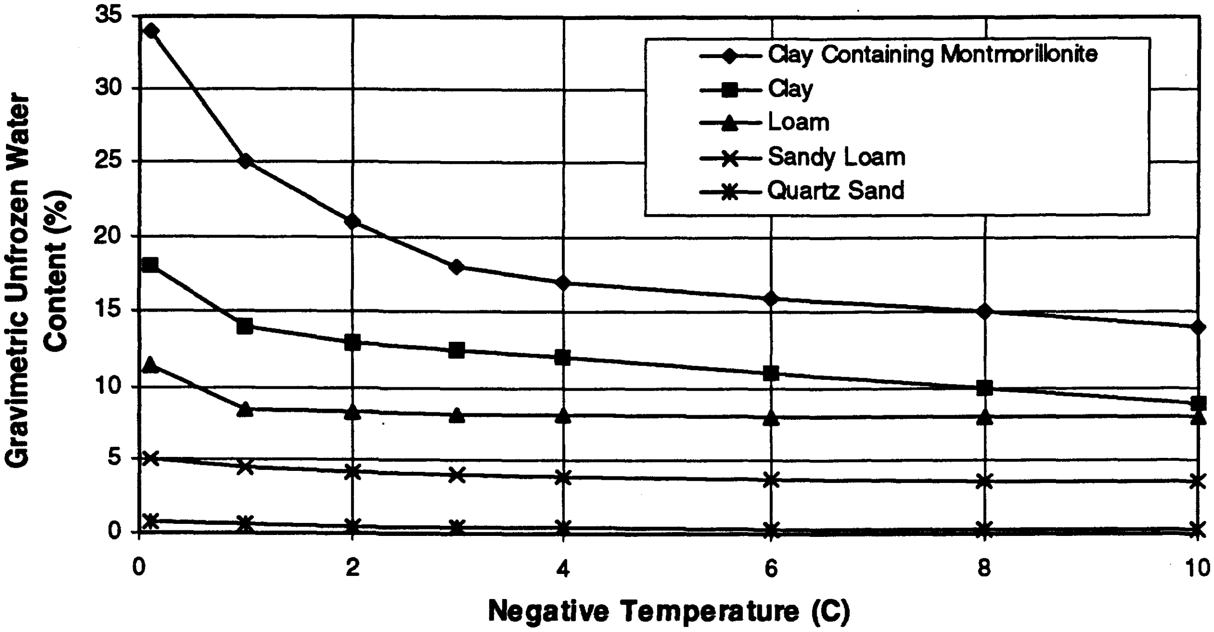


Figure 2.3 Unfrozen Water Contents in Typical Non-saline Soils (after Nersesova and Tsytovich, 1965)

Jame (1972) conducted laboratory experiments on fine silica flour to determine the relationship between unfrozen water content and freezing point depression at different freezing temperatures. One test involved gradually reducing the soil temperature until ice began to nucleate. This was done for soils at various water contents. The second test involved calorimetric determination of unfrozen water content for various sub-zero temperatures. The results of his findings are illustrated in Figure 2.4. Essentially, it was shown that the relationship between freezing point depression and water content is the same as the relationship between sub-zero temperature and unfrozen water content. This finding is significant because it permits the use of the freezing point depression curve for predicting unfrozen water content at different sub-zero temperatures. It will also allow correlation to be made between water content, suction (as given by soil water characteristic curves) and soil temperature. These are discussed shortly.

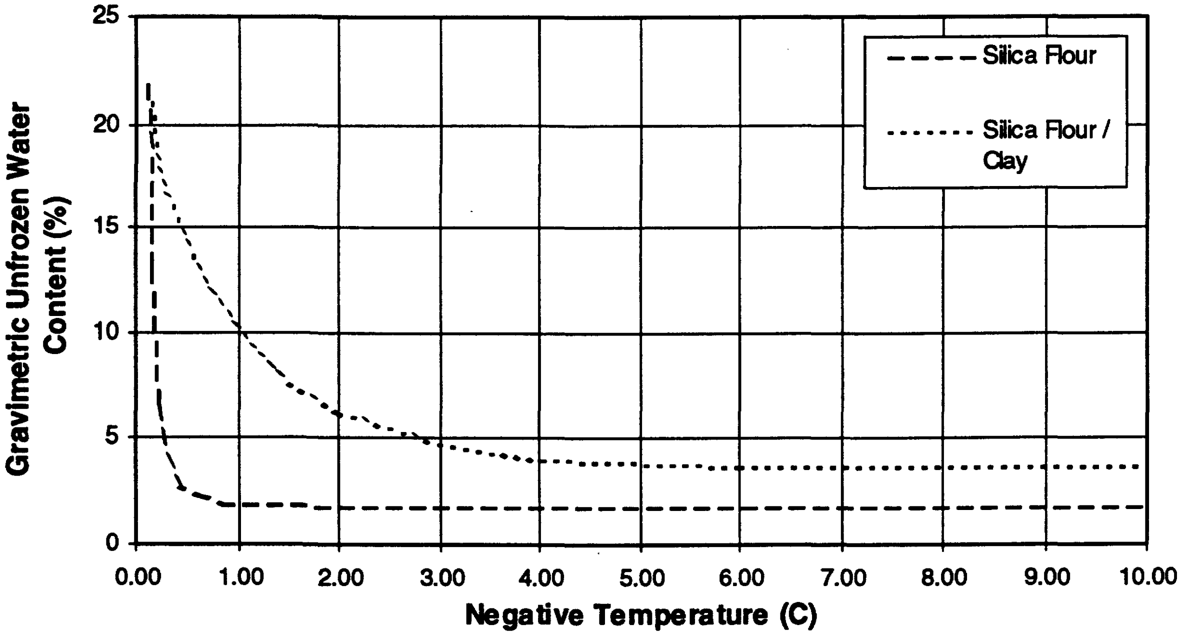


Figure 2.4 Relationship Between Freezing Point Depression and Water Content, or Between Unfrozen Water Content and Sub-zero temperature (after Jame, 1972)

2.3.1 Thermodynamic Equilibrium

The theoretical basis for soil water freezing relationships mentioned above comes from thermodynamic phase equilibrium theory. Therefore, it is important to discuss this theory as it applies to partially frozen, unsaturated soils. This information is also necessary for developing the computer model used in this study, as it is the basis for a correlation between the soil freezing curve (and freezing point depression curve) and the soil water characteristic curve.

Initial studies of unfrozen water content in frozen soils concentrated on the influence of salts on the freezing temperature of pore-water. Fisher (1924) indicated that the phenomenon is due partially to the presence of salts, and also to the way which the water is held within the soil. Fisher studied the capillary forces acting in pore-water in a freezing soil. Taber, in the 1920's, and Edlefsen and Anderson in 1943, suggested that water in fine grained soils is under the influence of adsorption forces. Miller (1965, 1973) distinguishes between "capillary" water forces in coarser grained, non-colloidal soils (i.e., sands and coarse silts), and "adsorptive" water forces in finer grained, wholly colloidal soils (i.e., fine silts and clays). The significance of these soil classifications becomes apparent when thermal equilibrium analysis is applied to water in frozen soils.

Analysis of the Gibbs free energy for any two phases in equilibrium can be used to derive the Clapeyron equation which relates how the equilibrium pressure changes with a change in temperature. The basic form of the Clapeyron equation is as follows:

$$\frac{dP}{dT} = \frac{\Delta h}{T\Delta V} \qquad [2.1]$$

where,

P = equilibrium pressure (kPa),

T = temperature of the system (K),

h = specific enthalpy difference between phases (kJ/kg), and

V = specific volume difference between phases (m^3/kg).

Various forms of the Clapeyron equation have been used for different purposes in the study of freezing soils. Schofield (1935), Takagi (1963), Low et al. (1968), and Jame (1972) presented equilibrium thermodynamic relationships which related the freezing point depression of soil water to soil suction. Figure 2.5 shows the results of calculations made by Jame (1972) of suction as a function of sub-zero temperature for a coarse grained soil. The figure shows that very high suctions develop within a relatively small sub-zero temperature range.

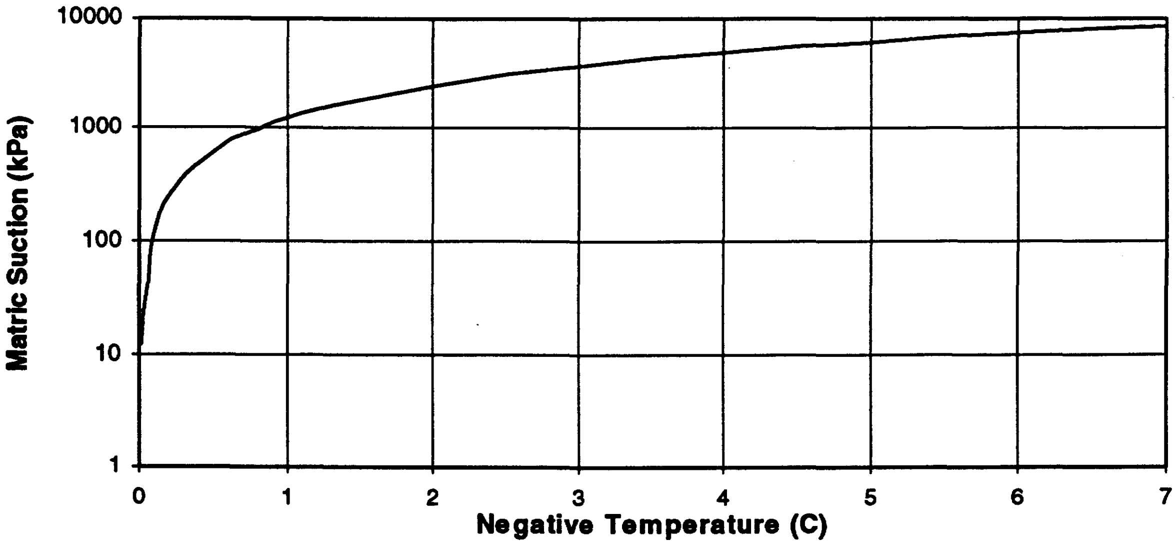


Figure 2.5 Theoretical Relationship Between Matric Suction and Sub-zero temperature for a Coarse Grained Soil (after Jame, 1972)

In the 1960's, researchers like Everett (1961) , Penner (1967), and Williams (1967) used a form of the Clapeyron equation as the basis for capillary models of ice segregation and frost heave. In the 1970's and 1980's, Harlan (1973), Taylor and Luthin (1978), Jame and Norum (1980) and others presented heat and mass transfer relationships which were inherently coupled in the frozen zone by soil freezing curves and the Clapeyron equation (i.e., the soil freezing curve related water content to temperature, and the Clapeyron equation related temperature to suction for use in computing hydraulic gradients). Miller (1978), Konrad and Morgenstern (1980), and Nixon (1991, 1992) used Clapeyron type equations to relate pore-water pressures to ice pressures beneath a growing ice lens in their studies of frost heaving mechanisms. Koopmans and Miller (1966), and Black and

Tice (1989) tried to derive the soil freezing curve from the soil water characteristic curves using equilibrium thermodynamics. The relationships between the soil water characteristic curves and soil freezing curves are discussed in the next section.

2.3.2 Relating Soil Freezing Curves to Soil Water Characteristic Curves

In a freezing soil, three possible phase interfaces exist: ice-water, air-water, and ice-air. If pore-ice pressure is assumed to be constant (which is apparently reasonable for normal hexagonal ice - see Jumikis, 1966; Jame 1972) and pore-air pressure is assumed constant (which is most often the case), then the two interfaces of significance are the air-water and ice-water interfaces. Figure 2.6 shows a schematic of a partially frozen unsaturated soil and the three phase stress state pressure variables. The difference between pore-air pressure and pore-water pressure in an unsaturated soil is called the matric suction. A soil water characteristic curve is used to show the variation of volumetric (liquid) water content with respect to changes in matric suction.

Williams (1964) first presented data on the relationship between a soil water characteristic curve measured at room temperature, and a soil freezing curve for the same material. The experimental data presented by Williams (1964) relating suction and temperature agreed closely with theoretical relationships given by the appropriate form of the Clapeyron equation. This finding was significant because it suggested that a theoretical soil freezing curve could be constructed using a measured soil water characteristic curve and the Clapeyron equation. In effect, the Clapeyron equation would provide the freezing temperatures corresponding to each water content or suction.

Miller (1973) related the moisture states achieved in freezing/thawing soils to drying/wetting processes in unsaturated soils. His theory was presented for soils that were either wholly colloidal (i.e., fine grained) or wholly non-colloidal (i.e, coarse grained). Soil types that fell between this range (i.e., were a combination of coarse and fine material) were not included in the Miller (1973) theoretical development due to the difficulty in applying the Clapeyron equation to soils dominated by both capillary and adsorptive forces. Miller found that for soils wholly dominated by adsorptive forces, a correlation exists between soil water characteristic data and soil freezing data. This agrees with the earlier experimental findings of Williams (1964), and Koopmans and Miller (1966). Results obtained by Koopmans and Miller (1966) are shown in Figure 2.7. In this figure it is clear that the freezing curve is similar to the drying curve, and the thawing curve is similar to the wetting curve. A best fit line has been added to the Figure to help with the interpretation.

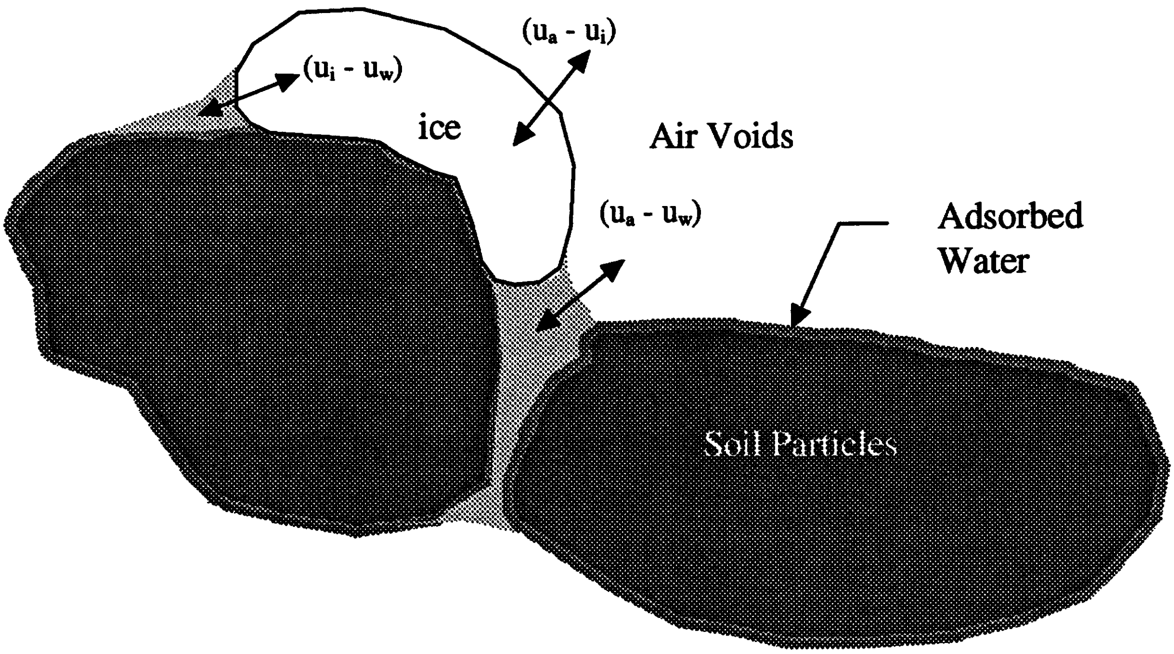


Figure 2.6 Schematic of Partially Frozen Soil Showing Relevant Stress State Variables (After Miller 1973)

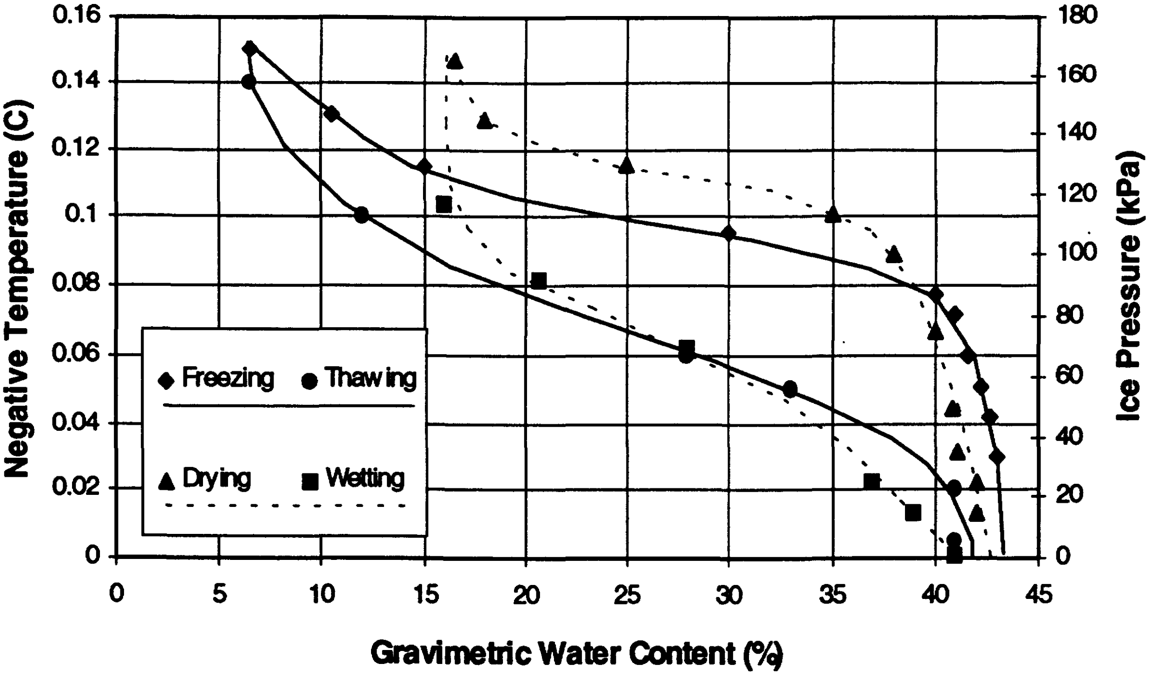


Figure 2.7 Experimental Relationships Between Soil Freezing Curves and Soil Water Characteristic Curves Including Hysteresis Effects (after Koopmans and Miller, 1966)

Application of the Kelvin equation to the ice-water and ice-air interface shows that a correction must be made to the relationship between soil water characteristic data and soil freezing data for soils wholly dominated by capillary forces. For the ice-water interface:

$$(u_i - u_w) = \frac{\sigma_{iw}}{r_{iw}} \qquad [2.2]$$

where,

u_i = ice pressure (kPa),

u_w = water pressure (kPa),

σ_{iw} = the surface tension between ice and water (kN), and

r_{iw} = the radius of curvature of the interface (m).

For the ice-air interface:

$$(u_i - u_a) = \frac{2\sigma_{ai}}{r_{ai}} \qquad [2.3]$$

where,

u_a = the air pressure (kPa),

σ_{ai} = the surface tension between air and ice (kN), and

r_{ai} = the mean radius of curvature of the interface (m).

Koopmans and Miller (1966) experimentally determined the air-water, ice-water ratio (i.e., σ_{aw}:σ_{iw}) to be 2.2 : 1; and Miller (1973) calculated the ice-water, air-ice ratio (i.e., σ_{iw}:σ_{ai}) to be 1: 3.2 by analysing the contact angle between interfaces.

Black and Tice (1989) correlated experimental soil water characteristic curve data to measured soil freezing data. They also presented unique power curve relationships which simultaneously represented both the soil freezing curve data and the soil water characteristic curve data for initially saturated soils composed of either coarse or fine material. An example of their experimental and computed results for a fine grained soil are shown in Figure 2.8 for the freezing / drying and thawing / wetting cases.

The general form of the Clapeyron equation used by Black and Tice (1989) is as follows:

$$u_w - \frac{u_i}{\gamma_i} = \frac{L_f}{273.15} T^* \qquad [2.4]$$

where,

γ_i = the specific gravity of ice (dec.),

L_f = the volumetric latent heat of fusion (kJ/kg),

T^* = the freezing point depression of pore water (°C).

Equation 2.4 in its current form does not relate freezing point depression, T^*, to matric suction, $(u_a - u_w)$. To make this connection, Black and Tice (1989) relate $(u_i - u_w)$ to $(u_a - u_w)$, using the ratios of surface tensions, $\sigma_{aw}:\sigma_{iw}$ discussed earlier. These correlations are presented below for both a coarse grained and fine grained soil.

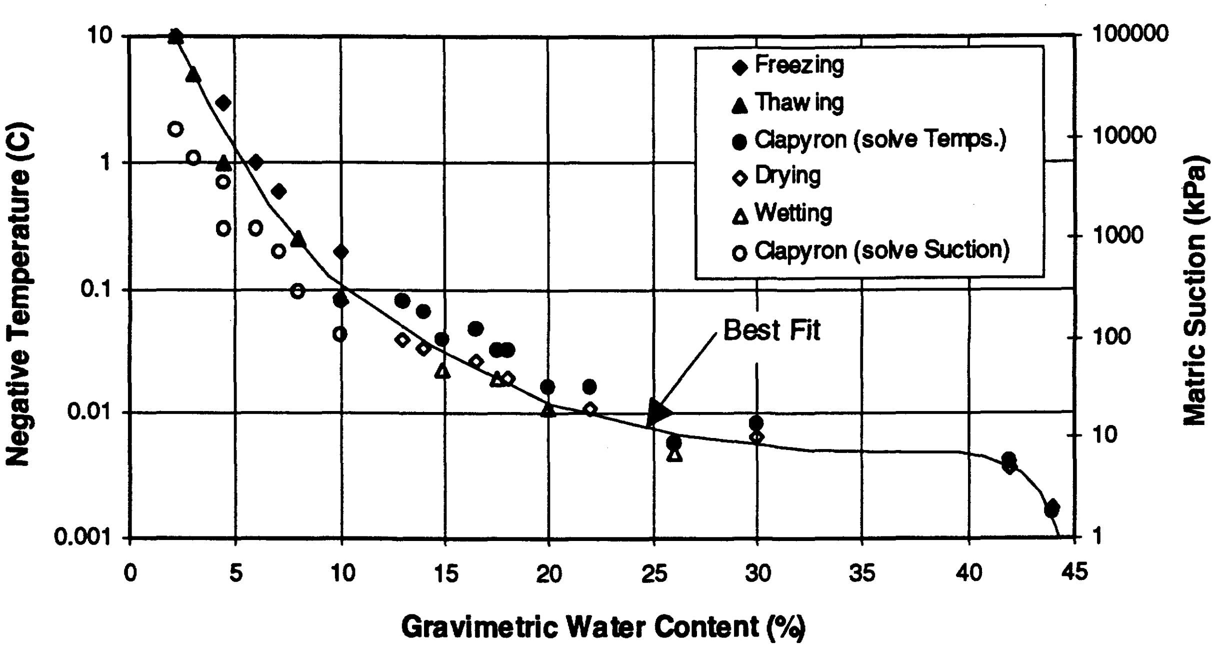


Figure 2.8 Combination of Experimental and Theoretical Soil Water Characteristic and Soil Freezing Curves for Windsor Sandy Loam (after Black and Tice, 1989)

When adsorptive forces << capillary forces (i.e., coarse grained):

$$(u_a - u_w) = \frac{\sigma_{aw}}{\sigma_{iw}}(u_i - u_w), \text{ or} \qquad [2.5]$$

$$(u_a - u_w) = 2.2(1110)\, T^* \qquad [2.6]$$

When capillary forces << adsorptive forces (i.e., fine grained):

$$(u_a - u_w) = (u_i - u_w), \text{ or} \quad [2.7]$$

$$(u_a - u_w) = 1110\,T^* \quad [2.8]$$

The constant value equal to 1110 kPa/°C combines the latent heat of fusion value, specific volume, and the conversion between the freezing temperature of water in Kelvins and degrees Celsius. The constant value of 2.2 is the ratio of surface tensions between air - water, and ice - water in a coarse grained soil.

The above relationships may be useful for obtaining soil freezing curve data which can not be measured easily. For either soil type, the soil freezing data can be calculated using measured soil water characteristic curve data and the appropriate form of the Clapeyron equation as presented by Black and Tice (1989). Currently, no Clapeyron type formulation exists for soils which contain a mixture of capillary and adsorptive water forces.

The preceding discussions are significant to the development of the current soil freezing model because a Clapeyron type equation can couple the temperature and suction states in pore-water undergoing a phase change. Even though no theoretical Clapeyron equation exists for soils containing capillary and adsorptive water, it may be possible to obtain a relationship between suction and temperature using a measured soil freezing curve and a measured soil water characteristic curve. If both curves are known, then the freezing temperatures from the soil freezing curve should be inherently linked to the soil water suctions through the unfrozen water content which is common to both curves. This will be discussed again in Chapter 3.

Other empirical methods have been developed for predicting the unfrozen water content versus temperature or freezing point depression curve. Low et al. (1968) present a detailed thermodynamic technique and its instructions for use. Their method is quite complicated to apply and is most accurate in predicting large freezing point depressions.

Tice et al. (1976) present a method which shows how to predict the unfrozen water content in soils from liquid limit data. Their method predicts the empirical constants, α and β for use in the simple power curve relationship:

$$w_u = \alpha\left(T^*\right)^\beta \qquad [2.9]$$

where,

w_u = the unfrozen gravimetric water content (dec.).

Their method gives very good correlation between measured and calculated values for liquid limits less than 100 and for relatively salt free soils. This type of power curve can also be used with values of α and β which are determined experimentally. Table 2.1 shows some published unfrozen water content parameters for use in the above equation.

Anderson et al. (1973) recognize the applicability of the power curve shown in equation 2.9, but they discuss the drawbacks of such a curve when used on clay-water systems. They suggest using a combination of two power curves. After testing eleven soil samples with specific surface areas ranging from 0.02 to 800 m^2/g they were able to regress values of α and β against specific surface area, S, as follows:

$$\ln(-\beta) = -0.2640 \ln(S) + 0.3711 \qquad [2.10]$$

$$\ln(\alpha) = 0.5519 \ln(S) + 0.2618 \qquad [2.11]$$

Combining equations 2.10 and 2.11 with equation 2.9 results in an equation by which the gravimetric unfrozen water content can be determined for salt free soils at freezing temperatures when only the specific surface area is known. This equation is given below.

$$\ln(w_u) = 0.2618 + 0.5519 \ln(S) - 1.449 S^{-0.264} \ln(T^*) \qquad [2.12]$$

The following list summarizes the significant points presented in this section of the literature review.

1. Water in soils freezes below 0°C.
2. Unfrozen water exists in frozen soils primarily due to negative pore-water pressures.
3. The unfrozen water content versus sub-zero temperature relationship applies regardless of the water content present when the soil first starts to freeze.
4. The Clapeyron and Kelvin equations provide a relationship between suction and temperature for soils dominated either by capillary or adsorptive water forces.
5. A relationship between temperature and suction states may be determined for a soil regardless of soil type if both the soil freezing, and soil water characteristic curves are known.
6. The soil freezing curve should ideally be measured in a laboratory, but other empirical methods of computing the necessary data have been developed.

Table 2.1 Unfrozen Water Content Parameters For Use In Equation 2.9

Soil Type	Specific Surface Area	α	β
Morin Clay	60	0.096	-0.406
Morin Clay	60	0.131	-0.505
Caen Silt	-	0.095	-0.227
Calgary Silt	-	0.096	-0.364
Manchester Silt	-	0.058	-0.425
Kaolin	-	0.104	-0.245
Allendale Clay	-	0.157	-0.187
Inuvik Clay	-	0.145	-0.254
Tomokomai Clay	54	0.195	-0.305
Suffield Clay	140	0.139	-0.315
Fairbanks Silt	40	0.048	-0.326
Illite	50.6	0.332	-0.273
Fairbanks Silt	-	0.074	-0.384
Undisturbed Fairbanks Silt	-	0.058	-0.439
Chena Silt	6	0.014	-1.460
Japanese Clay (45%)	-	0.128	-0.402
West Lebanon Gravel	15	0.021	-0.408
Manchester Silt	18	0.025	-0.515
Kaolinite (kGa-1)	23	0.058	-0.864
Chena Silt	40	0.032	-0.531
Leda Clay	58	0.108	-0.649
Morin Clay	60	0.095	-0.479
O'Brien Clay	61	0.104	-0.484
Goodrich Clay	68	0.0864	-0.456
Tuto Clay	78	0.128	-0.603
Sweden 478 Clay	113	0.271	-0.472
Suffield Silty Clay	148	0.111	-0.254
Frederick Clay	159	0.140	-0.279
Elleworth Clay	184	0.112	-0.293
Regina Clay	291	0.211	-0.238
Niagara Silt	37	0.066	-0.410
Norway LE-1 Clay	52	0.099	-0.523
Kaolinite No. 7	72	0.198	-0.689
Athena Silt Loam	83	0.060	-0.301
Sweden 201 Clay	106	0.197	-0.492
Hectorite	419	0.384	-0.369
Volcanic Ash	474	0.031	-0.097

2.4 Thermal and Hydraulic Properties of Frozen Soils

Modelling of transient freeze / thaw processes in soil requires that the thermal and hydraulic properties of the soils be known for both the freezing and non-freezing zones. From a heat transfer perspective it is necessary to know the thermal conductivity, volumetric specific heat capacity, and latent heat of fusion of water for the soil. From a mass transfer perspective, it is necessary to know the coefficient of permeability, and vapour diffusion coefficient. The unfrozen water content versus temperature relationship has been discussed in the previous section.

In the unfrozen zone, changes in thermal conductivity, volumetric specific heat, coefficient of permeability and vapour diffusion coefficient are a function of changes in water content. The relationships between these soil properties and water content are not valid in the frozen zone because the presence of ice changes the solid and liquid matrix of the soil. Excluding the coefficient of water permeability in frozen soils, there are generally well accepted methods of determining the required soil properties for both the freezing and non-freezing cases. These are discussed below.

2.4.1 Thermal Conductivity

Thermal conductivity is the amount of heat passing a unit cross-sectional area of soil, per unit time, under a unit temperature gradient. In equation form, it can be represented by:

$$\lambda = \frac{q\,L}{A(T_2 - T_1)} \qquad [2.13]$$

where,

λ = the thermal conductivity (W/mK),

q = the heat flux per unit time (W/m^2),

L = the length of flow (m)

A = the cross-sectional area (m^2), and

$T_{1,2}$ = the temperatures at each end of length, L (K).

The thermal conductivity of the soil system is a function of the thermal conductivites and quantities of each individual component in the soil, and of the combination of soil components (i.e., soil density and porosity).

Farouki (1981) discusses and compares various methods for calculating the thermal conductivity of frozen and unfrozen soils. He discusses the sensitivity of each method with respect to soil type (fine or coarse), degree of saturation, mineral composition, and phase state (i.e., frozen or unfrozen). Farouki concludes that the method provided by Johansen (1975) gives the best results for frozen or unfrozen, coarse or fine soils, at various degrees of saturation above 0.1. He adds that the method proposed by de Vries (1952) is more accurate for unfrozen coarse soils when the degree of saturation is between 0.1 and 0.2. Below a saturation of 0.1, none of the methods give good predictions. The method proposed by Kersten (1949) gives good results for frozen fine soils at a saturation below 0.9, but this method does not apply for any coarse grained soil (frozen or unfrozen) with either a high or low quartz content. In saturated soils, several methods appear to compare favorably, but Farouki (1981) is of the opinion that the method proposed by Johansen (1975) is easiest to use.

The method given by Johansen (1975) expresses the thermal conductivity of an unsaturated soil as a function of the thermal conductivity in the dry and saturated states at the same dry density. The expressions listed below enable the thermal conductivity to be calculated for various cases. The main equation used by Johansen (1975) is:

$$\lambda = (\lambda_{sat} - \lambda_{dry})\lambda_e + \lambda_{dry} \qquad [2.14]$$

where,

λ_{sat} = the saturated thermal conductivity (W/m°C),

= $0.75^n \lambda_s^{(1-n)}$ for the unfrozen case,

= $2.2^n \lambda_s^{(1-n)} 0.269^{W_u}$ for the frozen case,

w_u	=	the unfrozen volumetric water content (dec.),
n	=	the porosity of the soil,
λ_s	=	the effective solids thermal conductivity (W/m°C),
	=	$7.7^q\, 2.0^{1-q}$ if $q > 0.20$,
	=	$7.7^q\, 3.0^{1-q}$ if $q < 0.20$,
q	=	the quartz content as a fraction of total solids content(dec.),
λ_{dry}	=	the thermal conductivity of the soil matrix in the dry state (W/m°C),
	=	$\frac{0.137\gamma_d + 64.7}{2700 - 0.947\gamma_d}$ if the soil is in a natural state,
	=	$0.39\, n^{-2.2}$ if the soil is crushed,
γ_d	=	the dry density of the soil (kg/m^3),
λ_e	=	the Kersten number (dec.),
	=	$0.7 \log S_r + 1.0$ for a coarse, unfrozen soil,
	=	$\log S_r + 1.0$ for a fine, unfrozen soil,
	=	S_r for any frozen soil,
S_r	=	the degree of saturation (dec.), and
	=	$(\theta_i + \theta_u) / n$.

In the equations listed in above, the thermal conductivity of ice is assumed constant at 2.2 W/m°C, and that of water is 0.57 W/m°C.

The de Vries (1952) method was used by Harlan (1973) and subsequently Guymon and Luthin (1974), Jame (1977), Taylor and Luthin (1978), Guymon et al. (1980), and Flerchinger (1987). The equation is of the form as follows:

$$\lambda = \frac{\sum_{j=1}^{n} F_j \theta_j \lambda_j}{\sum_{j=1}^{n} F_j \theta_j} \qquad [2.15]$$

where,

θ_j = the volumetric content of the j^{th} component (m^3/m^3),

F_j = the weighting factor of the j^{th} component (dec.),

$$= \frac{1}{3}\sum_{n=1}^{3}\{1 + \left(\frac{\lambda_j}{\lambda_{air}} - 1\right)g_n\}^{-1}$$

g_n = $g_1 + g_2 + g_3 = 1$, a depolarization factor depending on the shape of the component (dec.), and

λ_{air} = $\lambda_a + \lambda_v$, the thermal conductivity of air and vapour (W/m°C).

In the Jame (1977) study, the thermal conductivity of the soil solid, λ_j, was taken as that of pure quartz (i.e., 8.54 W/m°C), and the thermal conductivity of air, λ_a, was taken as 0.025 W/m°C. For volumetric water contents above 0.2, the vapour phase thermal conductivity, λ_v, was 0.736 W/m°C and for water contents below 0.2, the vapour thermal conductivity varied linearly from 0.0736 W/m°C to zero at oven dryness. Water has a thermal conductivity of 0.573 W/m°C and ice has a thermal conductivity of 2.176 W/m°C. Values of g_n for the soil solid particles were chosen as 0.125, 0.125, and 0.75; which corresponds to particles having a shape of an ellipsoid of revolution. The values of g_1 and g_2 for the air were assumed to decrease linearly from 0.333 in water saturated soils to 0.105 at a volumetric water content of 0.2. Below this water content they varied linearly from a value of 0.105 to 0.015 at oven dryness. Values of g_n for ice were chosen to be the same as the soil solid particles. Prior to using de Vries (1952) method of calculating thermal conductivity, Jame (1977) compared experiment values with theoretical values for the non-freezing case. The results of this comparison are illustrated in Figure 2.9.

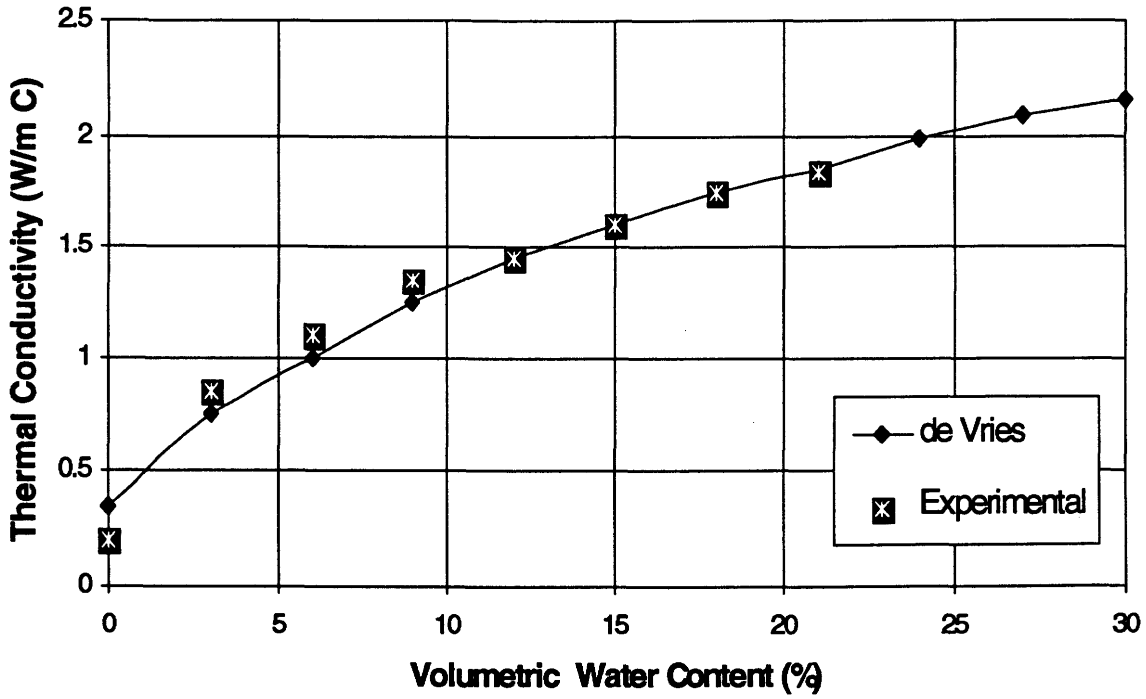


Figure 2.9 Comparison of Experimental and de Vries (1952) Method Thermal Conductivity for Non-Freezing Case (after Jame, 1977)

2.4.2 Volumetric Specific Heat and Apparent Specific Heat

If the temperature of a soil subjected to thermal gradients changes with time, then some of the heat is either stored or removed from the soil. The volumetric heat capacity of a soil is the heat energy required to raise the temperature of a unit volume of soil by 1 °C. The volumetric heat capacity is the product of the mass specific heat, c and the density, ρ (Farouki, 1981). The volumetric specific heat of a soil-liquid-ice mixture can be estimated by the expression :

$$c\rho = \sum_{j=1}^{n} (c\rho)_j \theta_j \qquad [2.16]$$

where,

$(c\rho)_j$ = the volumetric specific heat of the j^{th} component (J/m^3 °C).

If the dry density of the soil is known, equation 2.16 can be expressed as:

$$c\rho = \gamma_d (c_s + 4184 w_u + 2100 w_i) \quad [2.17]$$

where,

γ_d = the dry density of the soil solid (kg/m^3),

c_s = the mass specific heat of the soil (J/kg °C), and

w_i = the gravimetric content of ice (dec.).

The mass specific heat of water and ice are 4184 J/kg°C and 2100 J/kg °C respectively. In the experiments performed by Jame (1972), a mass specific heat of silica flour solid was calorimetrically determined to be 837 J/kg°C. This is the same silica flour that is used for verifying the computer model developed in this current study.

In frozen soils the phase change is a gradual process, thus the term specific heat capacity is not strictly applicable (Anderson et al, 1973). In its place it is possible to use the term apparent volumetric specific heat capacity, $\overline{\rho c}$, originally defined by Williams (1964) as:

$$\overline{\rho c} = \rho c + L_f \frac{\partial \theta_u}{\partial T} \quad [2.18]$$

where,

$\overline{\rho c}$ = the apparent volumetric specific heat (J/m^3 °C),

L_f = the latent heat of fusion of water (J/kg), and

θ_u = the volumetric unfrozen water content, (m^3/m^3).

The apparent volumetric specific heat incorporates the latent heat of fusion and the change in unfrozen water content with change in sub-zero temperature.

The latent heat of fusion is known to vary with temperature and unfrozen water content. In their model, Guymon and Luthin (1974) use the following relationship to correct for the change in latent heat as follows:

$$L_f = \rho_u L_f \left(\frac{\theta_u}{1-\theta_s} \right) \qquad [2.19]$$

where,

θ_s = the saturated volumetric water content (m^3/m^3).

Anderson et al. (1973) report that corrections to the latent heat of fusion are only necessary at temperatures below approximately -20°C. In this current study the latent heat of fusion is assumed constant at 344 kJ/kg.

2.4.3 Water Coefficient of Permeability

It was discussed in section 2.1.2 that there is a large change in water permeability near and behind the freezing front in the frozen soil. Harlan (1973) makes the assumption that the coefficient of permeability versus water content (or suction) function is the same in the frozen and unfrozen zones at any given liquid water content. Harlan (1973), however, was unable to conclude that his hypothesis was valid, as his own numerical results showed that too much water and ice accumulated behind the freezing front and that the water content decreased too rapidly at the freezing front. Numerical modelling carried out by Jame (1977) also suggests that the assumption made by Harlan (1973) is invalid. Jame (1977) suggests that the presence of ice probably disrupts the established flow paths and hence reduces the flow rate.

To account for the reduced flow, Jame (1977) introduced an impedance factor of the form:

$$k = k_h \, 10^{-E\theta_i}$$ [2.20]

where,

k = the actual coefficient of permeability (cm/s),

k_h = the unfrozen coefficient of permeability (cm/s), and

E = an empirical constant.

Jame (1977) calibrated his computer model by adjusting the empirical constant, E, in the above equation and by custom fitting a diffusivity versus water content function using data gathered from his initial tests. The permeability relationships then appeared to work well and give good results in other freezing simulations using the same material. Taylor and Luthin (1978), Hromadka (1987), Gosnik et al. (1988) and others have used the Jame (1977) approach and also obtained good computed results. Gosnik et al. (1988) report that typical values of 'E' are about 8 for fine sands and silts to 20 - 30 for coarser gravely soils. Black (1991) is very critical of the 'impedance factor' approach, stating that it is a "potent and wholly arbitrary correction function" for determining permeability. Results of the current study tend to support Black's opinion. This is discussed in Chapter 6.

Anderson et al. (1973) introduces a method for determining the coefficient of permeability in frozen soils. The Anderson et al. (1973) equation is of the form:

$$k = \frac{k_o}{-T^{\alpha}}$$ [2.21]

where,

k_o = the coefficient of permeability at -1°C, (cm/s), and

α = the slope of the K vs. -T on a log - log plot.

The α term is approximately equal to -5β, where β is the exponent parameter used in the power curve relationship given by Anderson et al. (1973) for unfrozen water content versus temperature functions (see Table 2.1 for some published values of the "β" parameter). The coefficient of permeability at -1°C is obtained from special tests described by Anderson et al. (1973).

Nixon (1992) compiled some coefficient of permeability values for frozen soils and plotted them on a log-log plot. These are shown in Figure 2.10.

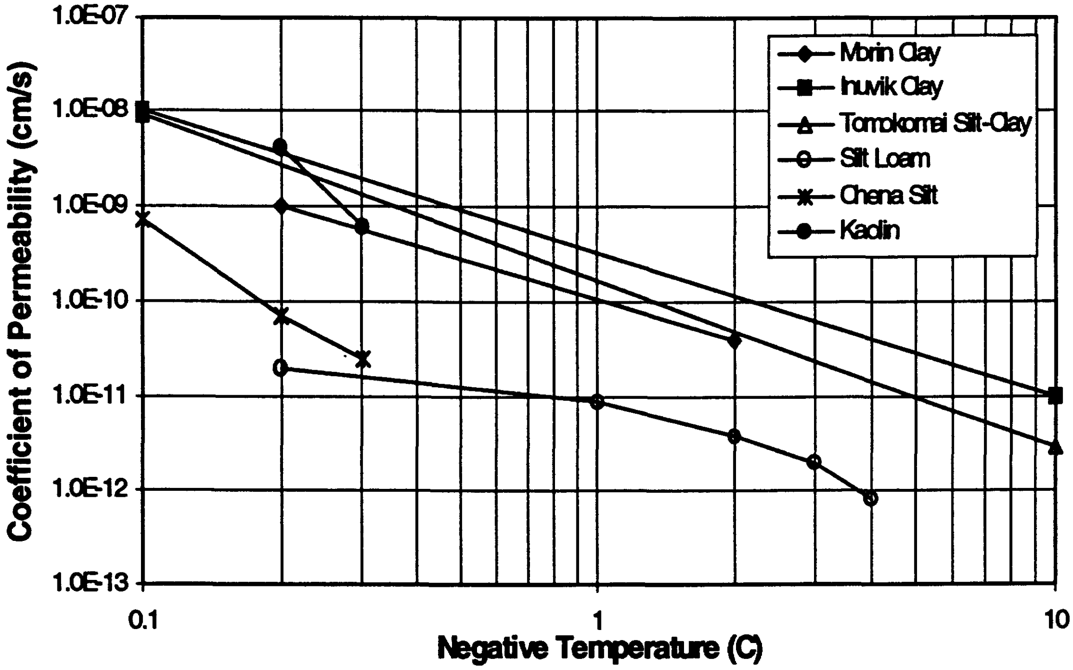


Figure 2.10 Hydraulic Conductivity of Frozen Soils (after Nixon, 1992)

Various other methods have been tried over the years. Tao (1994) used a power curve relationship first presented by Mualem (1976) that uses the normalized liquid saturation of the soil which is, itself, a function of volumetric ice content. Konrad and Morgenstern (1980) and Oliphant et al. (1982) present equations for determining the coefficient of permeability in the frozen zone which require parameters based on complex laboratory testing. Konrad and Morgenstern (1980) base their calculated coefficient of permeability on the soils segregation temperature (i.e., the temperature at which an ice lens starts to form). Nakano et al. (1982) base their calculation on the temperature gradient and the assumption that a form of the Clapeyron equation is valid for relating the pressure potential to temperature when both ice and liquid water phases are present.

Numerous studies have made no attempt to correct for the coefficient of permeability in frozen soils. Guymon and Luthin (1974) apply a form of the Gardner relationship for permeability in unsaturated soils to their freezing models. Flerchinger (1987) bases the permeability on suctions determined from a Brooks and Corey (1964) type calculation of matric potential. Flerchinger (1987) used his model to predict year round field moisture conditions with fairly good agreement between predicted and measured results.

To date, the literature shows that no single, acceptable method exists for determining the coefficient of permeability in a partially frozen, unsaturated soil. This is a major downfall to modelling freeze / thaw behaviour in soils. Until a suitable method of obtaining the frozen zone permeability is available, it is necessary to choose one of the approaches discussed above. One must either conduct extensive laboratory testing on freezing soils, or one must "calibrate" a model using measured data. This current study will compare results obtained with and without a permeability function correction for pore-ice blockage in the frozen zone.

2.5 Numerical Models of Freezing / Thawing Soils

Numerical models of soil freezing can be divided into two groups: those that assume the moisture content is static (i.e., considering only the thermal regime), and those that consider simultaneous coupled heat and mass transfer relationships. Extensive laboratory and field testing has shown that the former analysis is not sufficient. Thermal analysis does not account for any moisture redistribution and it does not give accurate temperature profiles in a partially frozen soil (Jame and Norum, 1980).

The literature shows that there are three basic approaches to modelling heat and mass transfer in freezing soils. The *capillary models* (Penner, 1959; Williams, 1967) credit capillary suction at the ice / water interface for moving water toward a growing ice lens. Evidence has shown that capillary suction effects do not explain continued ice segregation under high overburden pressures (Smith, 1985).

The *hydrodynamic models* (Harlan, 1973; Guymon and Luthin, 1974; Taylor and Luthin, 1978; Jame, 1977; Jame and Norum, 1980; and others) use coupled heat and mass transfer relationships to model the complete soil regime above and below the frost line - with or without frost heave. Hydrodynamic models use some capillary theory and they require accurate predictions of coefficient of permeability in the both frozen and unfrozen zones. As discussed in the previous section, an arbitrary 'impedance factor' has been introduced to calibrate the hydraulic conductivity in these models.

The secondary frost heave approach (Miller, 1978) was developed with the objective of predicting ice lens formation and frost heave. It builds on the previously mentioned models and assumes that the criterion for the initiation of a new ice lens within the frozen fringe is the same as the criterion for initiation of an air-filled crack in unsaturated, unfrozen soils. This approach led to the *rigid ice model* (Miller, 1978) of coupled heat and mass transfer for a freezing front descending through air-free, solute-free incompressible soil. According to Black (1991), the rigid ice model has inherent computational difficulties which make it difficult for use in practical problems.

Other models, based on the hydrodynamic approach, have also been developed with the sole purpose of explaining and predicting frost heave phenomenon (Gilpin, 1980; Konrad and Morgenstern, 1980; Nixon, 1991). This latter group incorporate segregation pressures and temperatures, and special calculations for water permeability in the frozen zone which are functions of the uniquely defined segregation temperatures.

The immediate application of the model proposed in Chapter 3 is to predict the thermal and moisture regime in a soil cover subjected to small overburden pressures and neglect any ice lensing. As a result, some form of a hydrodynamic model is most appropriate. The hydrodynamic model is discussed in more detail below.

2.5.1 Hydrodynamic Soil Freezing Models

The hydrodynamic model recognizes the coupled heat and mass transfers occurring simultaneously in freezing soils. The equations used in this approach are adapted from those used in unfrozen soils and are linked by the unfrozen water content versus temperature relationship and the Clapeyron equation.

The mass transfer equation given by Harlan (1973) is as follows:

$$\frac{\partial}{\partial z}\left(k\frac{\partial \psi}{\partial z}\right)=\frac{\partial \theta_u}{\partial t}+\frac{\rho_i}{\rho_u}\frac{\partial \theta_i}{\partial t} \qquad [2.22]$$

where,

k = the hydraulic conductivity (m/s),

ψ = the soil water suction (m),

θ_u = the unfrozen volumetric water content (m^3/m^3),

θ_i = the volumetric ice content (m^3/m^3),

ρ_u = the density of liquid water (kg/m^3),

ρ_i = the density of ice (kg/m^3), and

t = time (s).

The heat transfer equation given by Harlan (1973) is as follows:

$$\frac{\partial}{\partial z}\left(\lambda\frac{\partial T}{\partial z}\right)-\rho_u c_u V_z\frac{\partial T}{\partial z}=\rho c\frac{\partial T}{\partial t}-L_f\rho_i\frac{\partial \theta_i}{\partial t} \qquad [2.23]$$

where,

z	=	vertical position (m),
λ	=	the thermal conductivity of the soil (W/m °C),
T	=	temperature (°C),
c_u	=	the mass specific heat of liquid water (kJ/kg °C),
V_z	=	the fluid flow velocity in the z-direction (m/s),
ρc	=	the volumetric heat capacity of the soil (kJ/kg °C), and
L_f	=	the latent heat of fusion of water (kJ/kg).

The second term on the left side of equation 2.23 is the convective heat transfer term which is often neglected (see comments in section 2.2). The two terms on the right side of the equation can be combined as:

$$\frac{\partial}{\partial z}\left(\lambda \frac{\partial T}{\partial z}\right) - \rho_u c_u V_z \frac{\partial T}{\partial z} = \overline{\rho c}\frac{\partial T}{\partial t} \qquad [2.24]$$

where,

$\overline{\rho c}$ = the apparent volumetric specific heat (kJ/m^3 ^{0}C),

= $\rho c - L_f \rho_i \dfrac{\partial \theta_i}{\partial T}$ (Harlan, 1973; Jame, 1977), and

= $\rho c + L_f \dfrac{\partial \theta_u}{\partial T}$ (Anderson et al, 1973; Smith, 1985).

The apparent volumetric specific heat takes into account the latent heat of phase change during freezing or thawing. Therefore, the only modification to the heat transfer equation for freezing soils is the substitution of apparent volumetric specific heat for volumetric specific heat.

It should be noted that the apparent volumetric specific heat is a function of either the unfrozen volumetric water content or the volumetric ice content multiplied by ice density.

This former version of the equation is convenient because the partial derivative term is simply the slope of the soil freezing curve. However, this form of the equation was derived for use in freezing analysis which neglects mass transfer.

Observation of equations 2.22 and 2.23 reveals that the heat and mass transfer equations are coupled by the change in ice content per change in temperature. The system of coupled equations can be solved either by finite difference or finite element methods assuming appropriate boundary conditions are applied. The solution strategies must be flexible enough to handle small time steps during the early stages of frost penetration. The grid spacings are also fairly small to increase the stability of the model. A detailed comparison of the advantages and disadvantages of each of various solution strategies and numerical procedures is beyond the scope of this literature review.

2.6 Summary

The physical processes taking place during soil freezing are not well understood and as a result the methods used to analyse freezing and thawing in soils are varied. Evidence clearly shows that both heat and mass transfer processes occur simultaneously and are coupled in both the frozen and unfrozen soil zones. Unfrozen water has been shown to exist in frozen soils at various freezing temperatures and it is this unfrozen water (along with some vapour flux in certain circumstances) which facilitates mass transfer behind the freezing front. Numerous analytical methods are available for predicting soil thermal properties in the frozen and unfrozen zones, but no generally accepted method exists to predict water permeability in the frozen zone. Various types of computer models have been proposed to predict heat and mass transfer in freezing soils and the complexity of these models increases when analysis of ice segregation is attempted. As the complexity of some of these models increase, the applicability of the models for practical use by engineers decreases.

In this research program coupled heat and mass transfer equations will be derived for freezing and thawing unsaturated soils. The SoilCover (MEND 1993) model will be modified to verify proposed theory. Additional observations will be made about the necessity of using an ice impedance factor for calibrating the permeability function in the frozen zones. The revised numerical model will neglect ice lensing and convective heat transfer. Soil properties will be measured where possible or computed using the most appropriate and acceptable methods discussed previously in this chapter. The model will be verified using laboratory data collected by Jame (1977) and then applied to field data collected by O'kane (1995) from a waste rock cover at a mine site in the interior of British Columbia.

CHAPTER 3
THEORETICAL DEVELOPMENT

3.1 Introduction

The primary objectives of this research program are to develop freezing theory for unsaturated soils and to devise a numerical solution technique for implementing theory into existing non-freezing soil heat and mass transfer models. This chapter describes the theoretical development and the numerical solution technique as it is implemented into the non-freezing SoilCover (MEND, 1993) model. Changes are made to the coupled heat and mass transfer equations, but, in order to model soil freezing, changes must also be made in the way soil thermal and hydraulic properties are determined. The background for the soil property changes was presented in the literature review chapter. The revised model verification program, the presentation of modelling results, and a discussion of the modelling results are presented in Chapter 4, Chapter 5 and Chapter 6 respectively.

The SoilCover (MEND, 1993) model is a one dimensional finite element program which models transient water transport and heat flow in a soil profile. The model uses a physically based method for predicting the exchange of water and energy between the atmosphere and soil surface. Darcy's and Fick's laws are used to describe the flow of liquid water and vapour in the soil. Fourier's law for heat conduction and the latent heat of phase change between liquid and vapour phases describe the heat flow regime below the soil / atmosphere boundary. Evaporative fluxes from a saturated or unsaturated soil surface are predicted based on atmospheric conditions, vegetation cover and soil conditions. The modified Penman formulation (Wilson, 1990) is used to compute the actual rate of evaporation from the soil surface.

3.2 Existing Heat and Mass Transfer Equations

The heat and mass transfer equations used in SoilCover were derived by Wilson (1990) for the non-freezing case. The mass transfer equation is as follows:

$$\frac{\partial h_w}{\partial t} = c_w^1 \frac{\partial}{\partial z}\left(k_w \frac{\partial h_w}{\partial z}\right) + c_w^2 \frac{\partial}{\partial z}\left(D_v \frac{\partial P_v}{\partial z}\right) \qquad [3.1]$$

where,

h_w = total head (m),

t = time (s),

c_w^1 = $\frac{1}{\rho_u g m_2^w}$; the modulus of volume change with respect to the liquid phase,

ρ_u = density of water (kg/m^3),

g = acceleration due to gravity (m/s^2),

z = position (m),

k_w = the coefficient of permeability (m/s),

c_w^2 = $\frac{P+P_v}{P(\rho_u)^2 g m_2^w}$; the modulus of volume change with respect to the vapour phase,

m_2^w = slope of the soil water characteristic curve (1 / kPa),

P = total gas pressure in the air phase (kPa),

P_v = partial pressure due to water vapour (kPa),

D_v = diffusion coefficient of water vapour through soil (kg m / kN s),

= $\alpha\beta\left(D_{vap}\frac{W_v}{RT}\right)$,

α = tortuosity factor of soil (dec.),

= $\beta^{2/3}$ (dec.),

β	=	cross sectional area available for vapour flow (dec.),
D_{vap}	=	molecular diffusivity of water vapour in air (m^2/s),
T	=	temperature (K),
W_v	=	molecular weight of water (0.18 kg/kmole), and
R	=	universal gas constant (8,314 J/mole/K).

The heat transfer equation given by Wilson (1990) is:

$$C_h \frac{\partial T}{\partial t} = \frac{\partial}{\partial z}\left(\lambda \frac{\partial T}{\partial z}\right) - L_v\left(\frac{P+P_v}{P}\right)\frac{\partial}{\partial z}\left(D_v \frac{\partial P_v}{\partial z}\right) \qquad [3.2]$$

where,

C_h	=	volumetric specific heat of the soil as a function of water content ($J/m^3/°C$),
λ	=	thermal conductivity of the soil (W/m/°C), and
L_v	=	latent heat of vapourization of water (J/kg).

Equations 3.1 and 3.2 are not in a form that can easily be applied to a finite element formulation since the coupling variable, P_v, is not one of the dependent variables (i.e., T, h_w). Joshi (1993) replaced the total head term in the mass transfer equation and the vapour pressure terms in both equations with an equivalent water pressure term, ψ. The resulting mass transfer equation given by Joshi (1993) for non-freezing soils is:

$$m_2^w \frac{\partial \psi}{\partial t} = \frac{\partial}{\partial z}\left(k_w \frac{\partial}{\partial z}\left(\frac{\psi}{\rho_u g} + z\right)\right) + \frac{1}{\rho_u}\frac{\partial}{\partial z}\left(D_1 \frac{\partial \psi}{\partial z} + D_2 \frac{\partial T}{\partial z}\right) \qquad [3.3]$$

where,

D_1	=	$(1/\rho_u)D_v d_1$ (m^3 s/kg),
D_2	=	$(1/\rho_u)D_v d_2$ ($m^2/°C$ s),
d_1	=	$\frac{P_v W}{\rho_u RT}$ (dec.),

d_2 = $\frac{\partial P_{vs}}{\partial T} h_r - \frac{P_v \psi W}{\rho_u R T^2}$ (kg/m s^2°C)

ψ = soil matric suction (kPa),

= $-u_w$ when the pore-air pressure is assumed atmospheric, and

u_w = the pore-water pressure (kPa).

The heat transfer equation for non-freezing soils was modified by Joshi (1993) as follows:

$$C_h \frac{\partial T}{\partial t} = \frac{\partial}{\partial z}\left(\lambda \frac{\partial T}{\partial z}\right) - L_v \frac{\partial}{\partial z}\left(D_1 \frac{\partial \psi}{\partial z} + D_2 \frac{\partial T}{\partial z}\right). \qquad [3.4]$$

3.3 Derivation of the Modified Heat and Mass Transfer Equations for Freezing Soils

Equations 3.3 and 3.4 represent the transient thermal and water pressure stress states in a soil for non-freezing conditions. In order to illustrate how these are modified for freezing conditions it is advantageous to begin with the water phase continuity equation for a partially frozen soil.

$$\theta_w = \theta_u + \frac{\rho_i}{\rho_u} \theta_i \qquad [3.5]$$

where,

θ_w = the total volumetric moisture content in the soil (m^3/m^3),

θ_u = the total volumetric liquid water content in the soil (m^3/m^3),

θ_i = the total volumetric ice content in the soil (m^3/m^3), and

ρ_i = the density of ice (kg/m^3).

The change in storage of total water content in a elemental volume of soil over time is equal to the magnitude of the flux terms on the right side of the mass transfer continuity relationship (i.e., Eq. 3.3). Substituting the liquid and vapour flux terms into the time derivative of the water phase continuity equation (i.e,. Eq. 3.5) results in:

$$\frac{\partial \theta_u}{\partial t}+\frac{\rho_i}{\rho_u}\frac{\partial \theta_i}{\partial t}=\frac{\partial}{\partial z}\left(k_w\frac{\partial}{\partial z}\left(\frac{\psi}{\rho_u g}+z\right)\right)+\frac{1}{\rho_u}\frac{\partial}{\partial z}\left(D_1\frac{\partial \psi}{\partial z}+D_2\frac{\partial T}{\partial z}\right). \qquad [3.6]$$

If no freezing has taken place in the soil then the change in storage of unfrozen water is a function of the change in matric suction, and equation 3.6 reduces to equation 3.3. If freezing has taken place then the unfrozen water content is primarily a function of change in sub-zero temperature and its value is known from the soil freezing curve. For the freezing case, the change in unfrozen water content can be obtained using the slope of the soil freezing curve and the change in sub-zero temperature. Thus, equation 3.6 can be written as:

$$m_2^i\frac{\partial T}{\partial t}+\frac{\rho_i}{\rho_u}\frac{\partial \theta_i}{\partial t}=\frac{\partial}{\partial z}\left(k_w\frac{\partial}{\partial z}\left(\frac{\psi}{\rho_u g}+z\right)\right)+\frac{1}{\rho_u}\frac{\partial}{\partial z}\left(D_1\frac{\partial \psi}{\partial z}+D_2\frac{\partial T}{\partial z}\right) \qquad [3.7]$$

where,

m_2^i = the slope of the soil freezing curve (1 / °C).

Equation 3.7 is the mass transfer equation for regions in a soil where freezing is occurring, or, where ice already exists.

The heat transfer equation is modified to include the latent heat of the phase change between liquid and solid phases by adding the appropriate term on the right side of equation 3.4 as follows:

$$C_h \frac{\partial T}{\partial t} = \frac{\partial}{\partial z}\left(\lambda \frac{\partial T}{\partial z}\right) - L_v \frac{\partial}{\partial z}\left(D_1 \frac{\partial \psi}{\partial z} + D_2 \frac{\partial T}{\partial z}\right) + L_f \rho_u \frac{\partial \theta_i}{\partial t} \qquad [3.8]$$

where,

L_f = the latent heat of fusion of water (334 kJ/kg °C).

Equations 3.7 and 3.8 are the heat flow and mass transfer continuity relationships for a freezing or partially frozen soil. The objective now, is to rearrange these equations such that they are solvable within the existing SoilCover program finite element formulation. Observation of the modified heat and mass transfer equations reveals two points. First, they are coupled by the partial vapour pressure variable and by the volumetric ice content variable. The assumption can be made that the primary coupling variable in a freezing soil is the volumetric ice content. This appears to be a reasonable assumption for the soil regions near the freezing front when one considers the relatively large volume of liquid water which changes phase to ice at this point compared with the volume of vapour which changes phase to liquid water. This may be a questionable assumption for regions well behind the freezing front where the primary mode of moisture transport is in the vapour phase. The second observation about the revised heat and mass equations is that there are three unknown variables (i.e., T, ψ, and θ_i) and only two equations. Thus, the system of equations appears indeterminate.

Three steps need to be taken to render the system of equations determinate. First, the modified mass transfer equation (i.e., Eq. 3.7) is written such that the volumetric ice content term is isolated on the left side as follows:

$$\frac{\rho_i}{\rho_u} \frac{\partial \theta_i}{\partial t} = \frac{\partial}{\partial z}\left(k_w \frac{\partial}{\partial z}\left(\frac{\psi}{\rho_u g} + z\right)\right) + \frac{1}{\rho_u} \frac{\partial}{\partial z}\left(D_1 \frac{\partial \psi}{\partial z} + D_2 \frac{\partial T}{\partial z}\right) - m_2^i \frac{\partial T}{\partial t}. \qquad [3.9]$$

Second, the right hand side of equation 3.9 is substituted into the volumetric ice content term in the modified freezing heat transfer equation (i.e., Eq. 3.8) as follows:

$$C_h \frac{\partial T}{\partial t} = \frac{\partial}{\partial z}\left(\lambda \frac{\partial T}{\partial z}\right) - L_v \frac{\partial}{\partial z}\left(D_1 \frac{\partial \psi}{\partial z} + D_2 \frac{\partial T}{\partial z}\right) + L_f \frac{\rho_u^2}{\rho_i} \frac{\partial}{\partial z}\left(k_w \frac{\partial}{\partial z}\left(\frac{\psi}{\rho_u g} + z\right)\right)$$
$$+ L_f \frac{\rho_u}{\rho_i} \frac{\partial}{\partial z}\left(D_1 \frac{\partial \psi}{\partial z} + D_2 \frac{\partial T}{\partial z}\right) - L_f \frac{\rho_u^2}{\rho_i} m_2^i \frac{\partial T}{\partial t}. \qquad [3.10]$$

Equation 3.10 is now the modified heat <u>and</u> mass transfer for the freezing zone in a soil. The above equation now contains two unknowns, Ψ and T; and it is still not solvable in this form as there are two unknown variables and only one equation.

Thermodynamic phase equilibrium in a freezing soil was considered in Chapter 2. At that point, the Clapeyron equation was introduced and shown to be useful (in certain circumstances) for relating suctions to temperatures in freezing soils. Various forms of the Clapeyron equation work adequately for soils that are wholly dominated by either adsorptive water forces <u>or</u> capillary water forces, but these relationships fail for soils containing a combination of capillary and adsorptive water forces. The general form of the Clapeyron equation for equilibrium between any two phases (Equation 2.1) is repeated again as follows:

$$\frac{d\psi}{dT} = \frac{\Delta H}{T \Delta V} \qquad [3.11]$$

where,

ΔH = change in internal energy between phases (kJ/kg), and

ΔV = change in specific volume between phases (m^3/kg).

Equation 3.11 clearly shows there is a unique relationship between suctions and temperatures in a material undergoing a phase change. The problem with using this type of relationship in freezing soils is that the equation is not clearly defined for all soil types (Black and Tice, 1989; Koopmans and Miller, 1966). Theoretically, it is possible to avoid this problem by combining data from the soil water characteristic curve and the soil freezing curve.

If the soil water characteristic curve and soil freezing curve are known, then for regions of soil where freezing is occurring, a unique relationship exits between suction and temperature. The slope of the soil freezing curve can divided by the slope of the soil water characteristic curve to give a value for the right side of equation 3.11 as follows:

$$\frac{\partial \psi}{\partial T} = \frac{m_2^i}{m_2^w} = \frac{\partial \theta_u}{\partial T}\frac{\partial \psi}{\partial \theta_u} = G \qquad [3.12]$$

where,

G = the ratio between change in suction and change in temperature for a given unfrozen water content in a freezing soil (kPa / °C).

Equation 3.12 can now be used to eliminate the suction variable in the combined heat and mass transfer equation (i.e., Eq. 3.10) so that only one equation with one unknown remains. Making this substitution and grouping like terms results in:

$$\left(C_h + L_f\frac{\rho_u^2}{\rho_i}m_2^i\right)\frac{\partial T}{\partial t} = \frac{\partial}{\partial z}\left(\lambda\frac{\partial T}{\partial z}\right) - (L_v - L_f\frac{\rho_u}{\rho_i})\frac{\partial}{\partial z}\left(D_1 G\frac{\partial T}{\partial z} + D_2\frac{\partial T}{\partial z}\right)$$
$$+ L_f\frac{\rho_u^2}{\rho_i}\frac{\partial}{\partial z}\left(k_w G\frac{\partial}{\partial z}\left(\frac{T}{\rho_u g} + z\right)\right). \qquad [3.13]$$

It is interesting to note that the term ($C_h + L_f\ \rho_u^2/\rho_i\ m_2^i$) is the same as the "apparent specific heat" term used commonly in freeze / thaw analysis of soils. The first term on the right side of Eq. 3.13 is the conductive heat transfer term; the second term accounts for the net latent heat removed from the system due to phase changes from vapour to liquid and liquid to solid phases [1]; and the final term on the right side accounts for the liquid flowing into the system that changes phase and releases latent heat.

3.4 Solution Strategy

Equation 3.13 can be solved to give the soil temperature profile in the freezing or frozen soil zones. The suction profile in the freezing zone is obtained by looking up the suction which corresponds to the new unfrozen water contents given by the soil freezing curve for each newly solved temperature.. The modified numerical model uses the combined heat / mass transfer equation in the following way:

1. Using the previous time step suctions, the program computes the unfrozen water content from the soil water characteristic curve for every node in the finite element mesh. It then uses this unfrozen water content and the soil freezing curve to determine the freezing point depression temperature for every Gauss point. If a new Gauss point temperature is below the freezing point temperature, or if ice already exists at a Gauss point, then ice will, or may continue to form at the Gauss point during the next time step.
2. The program then computes the "apparent specific heat" and latent heat "ice flux" term constant values based on average thermal and hydraulic properties between the current and previous time steps.

---

[1] Note, this term is not a true sublimation term. It indirectly accounts for vapour - solid phase changes during freezing and solid - vapour phases changes during melting. It does not account for direct solid to vapour phase changes in a frozen soil (i.e., ice subliming to vapour without passing through a liquid phase).

3. At each Gauss point where ice forms or already exists, the mass transfer equation in the frozen zone is 'turned off' and the modified freezing heat and mass transfer equation (i.e., Eq. 3.13) is 'turned on'. The program then solves for temperatures in the frozen zone, and for temperatures and suctions in the unfrozen zone.

4. At the end of each iteration, the suctions and ice contents are calculated using back substitution for each node in the frozen zone. The suctions are determined as mentioned above, and the ice contents are obtained by back substitution into the water phase continuity equation.

5. The iterations continue until the system converges based on temperature and suction at each node.

6. Ice contents at each node are stored in a global array to be used in soil thermal and hydraulic property calculations at the next time step. They are also used in checking the total water balance.

Figure 3.1 on the following page shows the flowchart algorithm for the modified numerical solution within the program's iteration subroutine where the element stiffness and mass matrices are computed. A complete listing of the revised computer code is given in the appendix.

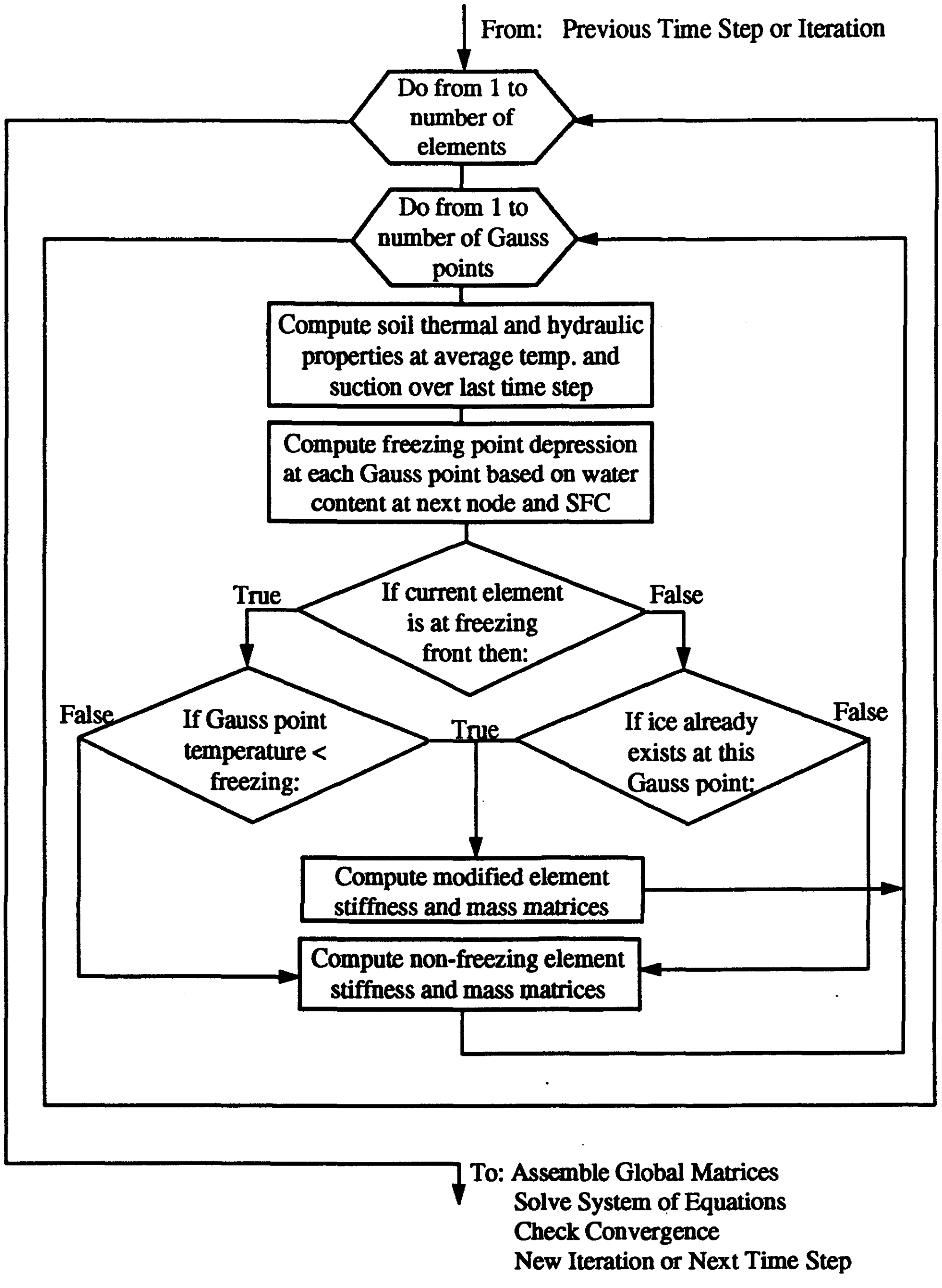


Figure 3.1 Flowchart Showing Criteria for Using Modified Soil Freezing Equation During Assembly of Element Stiffness and Mass Matrices

CHAPTER 4
REVISED MODEL VERIFICATION PROGRAM

4.1 Introduction

The revisions to the SoilCover (MEND, 1993) program for freezing analysis were verified in two ways. First, it was necessary to verify that the theoretical formulations presented in Chapter 3 produced reasonable results when compared with carefully measured laboratory data. This verified that the dependent variables, suction and temperature, were being solved accurately in the revised finite element formulation. In addition, careful comparison with the laboratory data was necessary to determine if the computed values for ice content were acceptable. Finally, laboratory verification was necessary to ensure that the revisions to the soil property calculation functions were accounted for where necessary (i.e., that soil properties were modified to a account for ice content effects). Laboratory data verification did not take into account any thawing processes.

4.2 Laboratory Data Modelling Program

Jame (1972, 1977) carried out detailed investigations of freezing phenomenon in a fine grained silica flour material. In his initial work, Jame (1972) carried out experiments to determine the relationships between the unfrozen water content, sub-zero temperature, and freezing point depression for the silica flour. His later work (Jame, 1977; Jame and

Norum, 1980) involved freezing of a horizontal column of silica flour while monitoring the temperature and total water content profiles with respect to time.

The material used by Jame (1972, 1977) was a # 40 silica flour with 72% passing the # 325 sieve. Jame (1977) prepared the silica flour at different initial moisture contents and packed it into lucite tubes, 30 cm in length and 10 cm in diameter. Jame estimated the dry density of the packed material to be 1.33 Mg/m^3. Hollow brass circulation plates were placed at both ends of the column which were then sealed with wax so that no water could flow in or out of the system. Provision was made for air movement within the column and to ensure that the air pressure remained atmospheric. The apparatus was instrumented with twelve thermocouples at 2.5 cm intervals and insulated with Styrofoam and rock wool. Moisture contents were measured using the gamma ray attenuation method through 2 mm holes in lead blocks surrounding the sample.

Each experiment began by circulating cold fluids from temperature control baths through the brass circulation plates at each end of the column. The initial uniform temperatures of the samples ranged from 20 °C to 5 °C, depending on the test. Once the uniform initial temperature was reached, the temperature at one end of the column was maintained at the initial temperature while the other end of the column was cooled rapidly to the desired cold end temperature below 0 °C. Moisture and temperature measurements were taken periodically over the 72 hour duration of each test. At the end of each test, gravimetric moisture content measurements were carried out to verify the moisture contents measured using the gamma ray method. More details of the experimental procedures and apparatus are given by Jame (1977).

The data in Table 4.1 summaries the initial conditions and boundary conditions for the three of the Jame (1977) tests.

Table 4.1 Test Conditions for Jame (1980) Experiments

Test	Initial Uniform Temperature (°C)	Initial Moisture Content (% by weight)	Cold End Temperature (°C)	Warm End Temperature (°C)
1	20	15.6	-10	20
2	5	15.0	-5.9	4.25
3	5	10.0	-5.2	5.0

The results of the freezing tests conducted by Jame (1980) verify three hypotheses regarding the freezing of a fine grained, silty material. These are as follows:

1) There is a redistribution of water from the unfrozen zone to the frozen zone where it accumulates as ice.
2) There is a clearly defined freezing front as indicated by the change in water contents.
3) The ice content will build up behind the freezing front if the advancing frost front becomes somewhat stationary and water is free to flow.

Figures 4.1 through 4.3 show measured temperature and total water content profiles for the three freezing tests reported by Jame (1980). It should be noted that the total water content consists of both ice and liquid water in the frozen zone (i.e., left side of Figure) and only of liquid water in the unfrozen zone (i.e., right side of Figure).

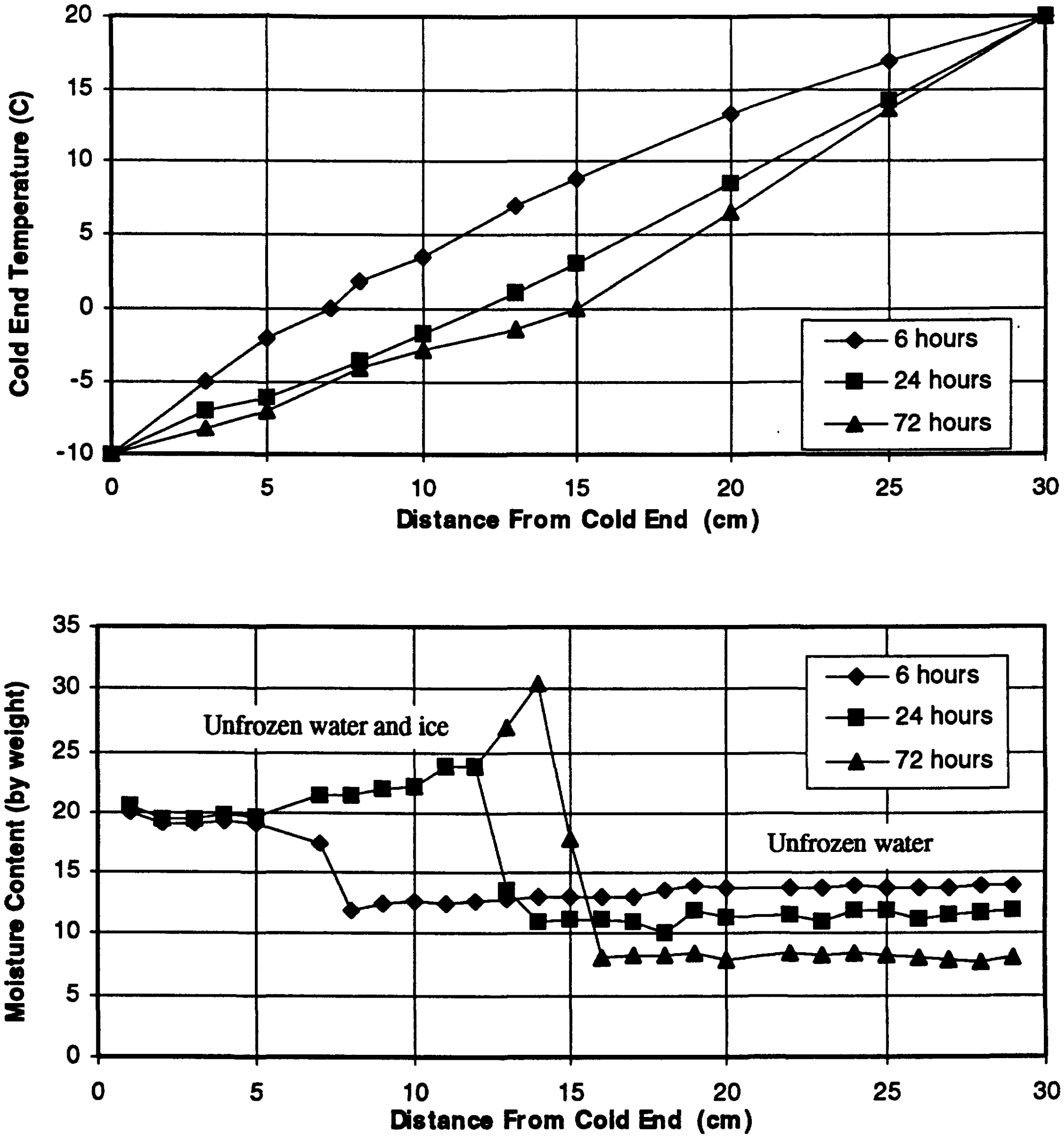


Figure 4.1 Experimental Results for Test 1 (after Jame, 1980)

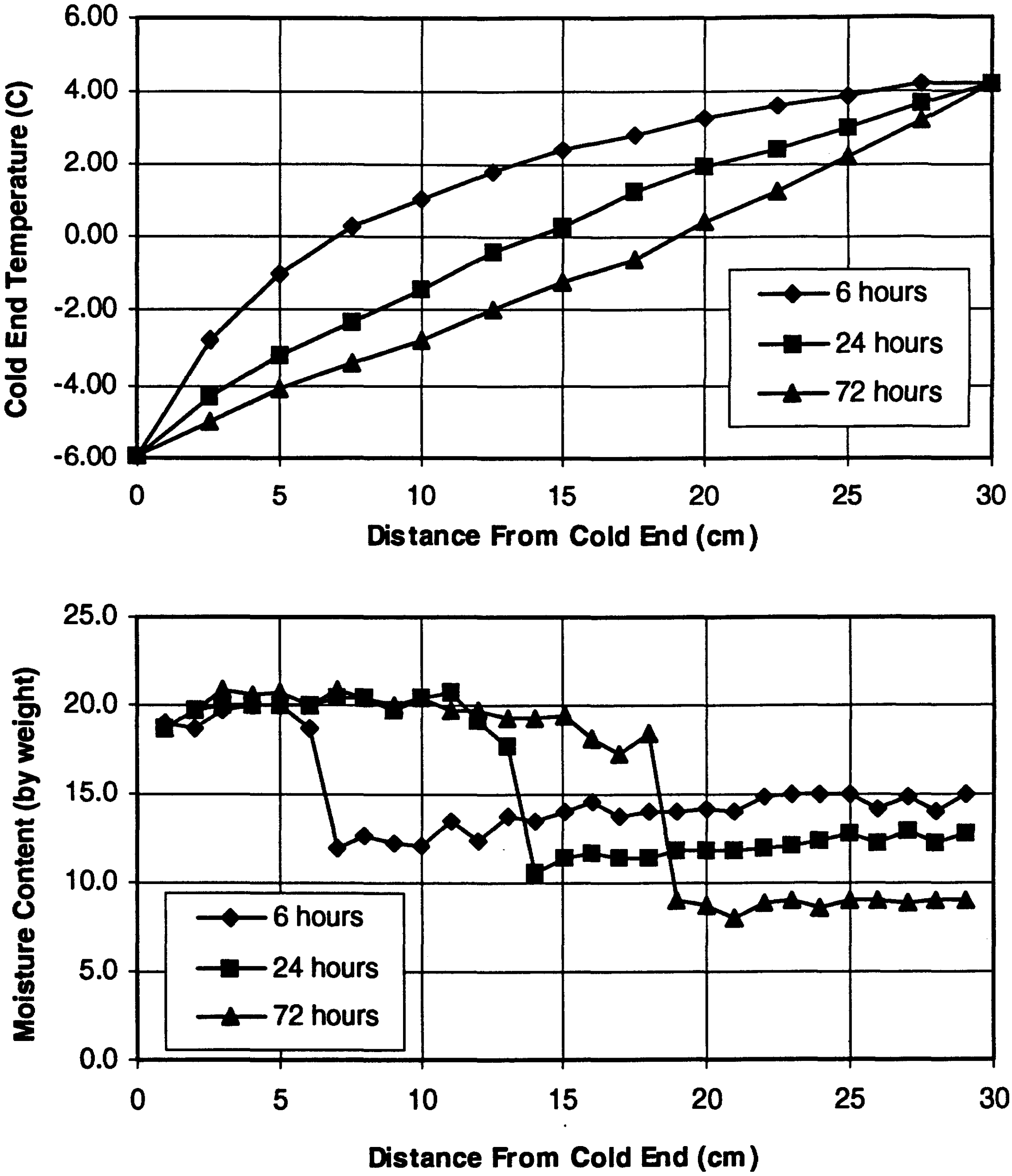


Figure 4.2 Experimental Results for Test 2 (after Jame, 1980)

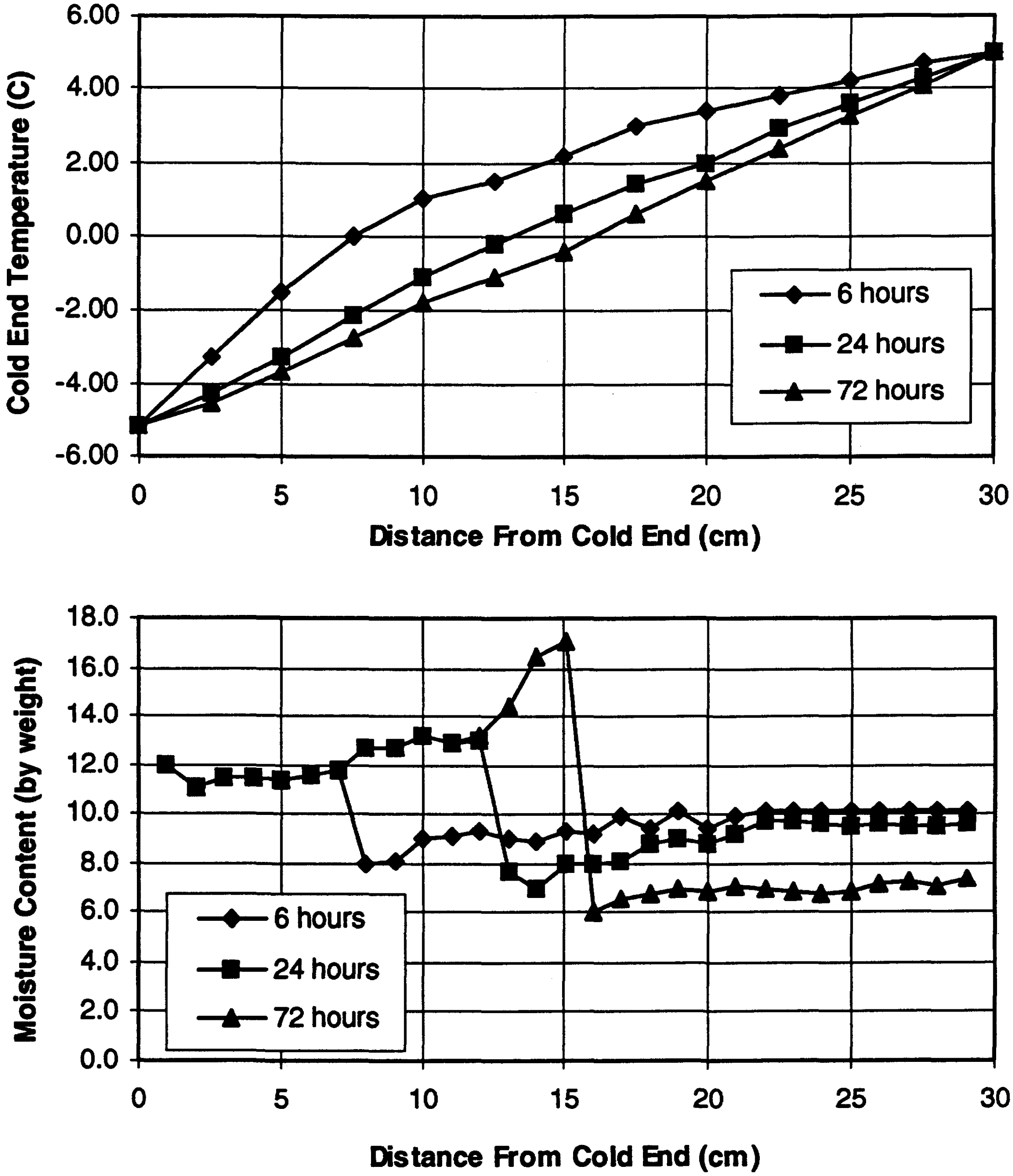


Figure 4.3 Experimental Results for Test 3 (after Jame, 1980)

4.3 Soil Properties Used in The Laboratory Data Modelling Program

In order to model the experimental data reported by Jame (1980) it was necessary to determine the thermal and hydraulic material properties required as input in SoilCover. These properties included: the soil freezing curve (i.e., unfrozen water as a function of sub-zero temperature), the soil water characteristic curve, the saturated coefficient of permeability, the coefficient of permeability as a function of matric suction, the ice impedance factor for the frozen soil, the thermal conductivity as a function of total moisture content, and the volumetric specific heat as a function of total moisture content.

Jame (1977) used a silica flour that was no longer available for purchase for this study. However, a similar material for soil property measurements was obtained. Figure 4.4 below shows the approximate grain size curve of the Jame (1972, 1977) silica flour material and the measured grain size of the silica flour used in this study. The specific gravity of the material used in this study was measured to be 2.65. Jame (1977) did not report a specific gravity of the # 40 silica flour used for that study.

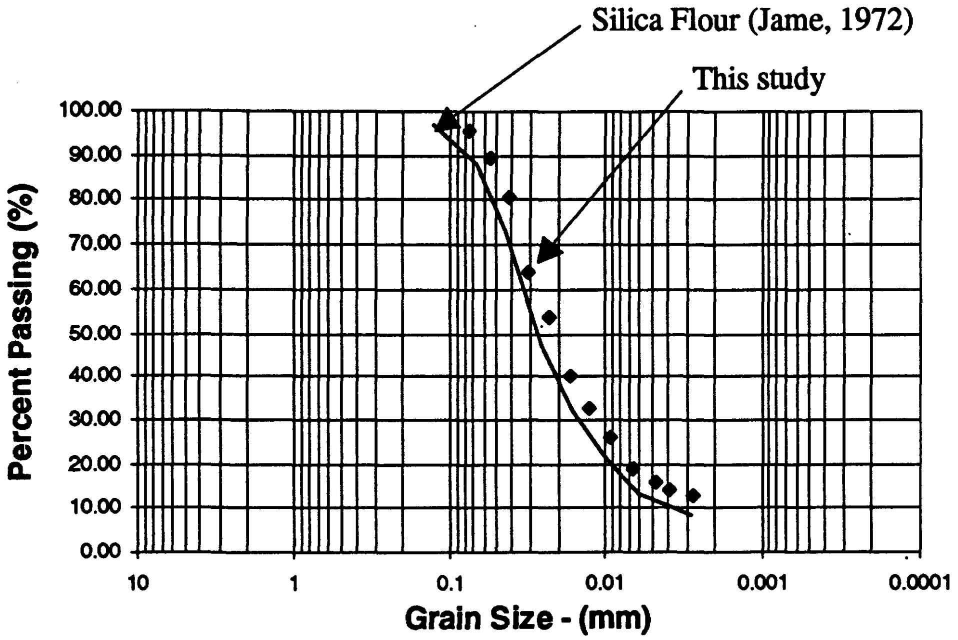


Figure 4.4 Grain Size Distribution for Silica Flour Soil Property Testing

The results of the grain size distribution test show that the material used in this study was very similar to that used by Jame (1972, 1977). As a result, it was assumed that the freezing test experimental results obtained by Jame (1977) could be simulated using material properties obtained from soil property tests conducted on the silica flour used in this study.

The soil freezing curve for the silica flour used by Jame (1977) was discussed in the literature review chapter and is presented again in Figure 4.5. A semi - log plot of the soil freezing curve is shown in Figure 4.6. In this form it is similar in shape to a soil water characteristic curve.

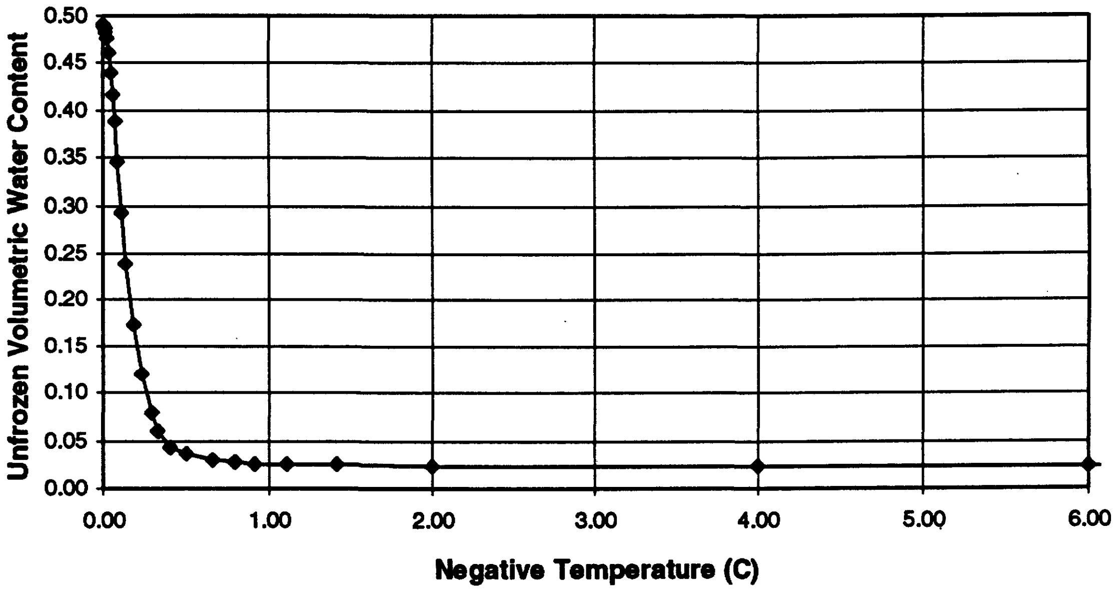


Figure 4.5 Soil Freezing Curve for Silica Flour (after Jame, 1972)

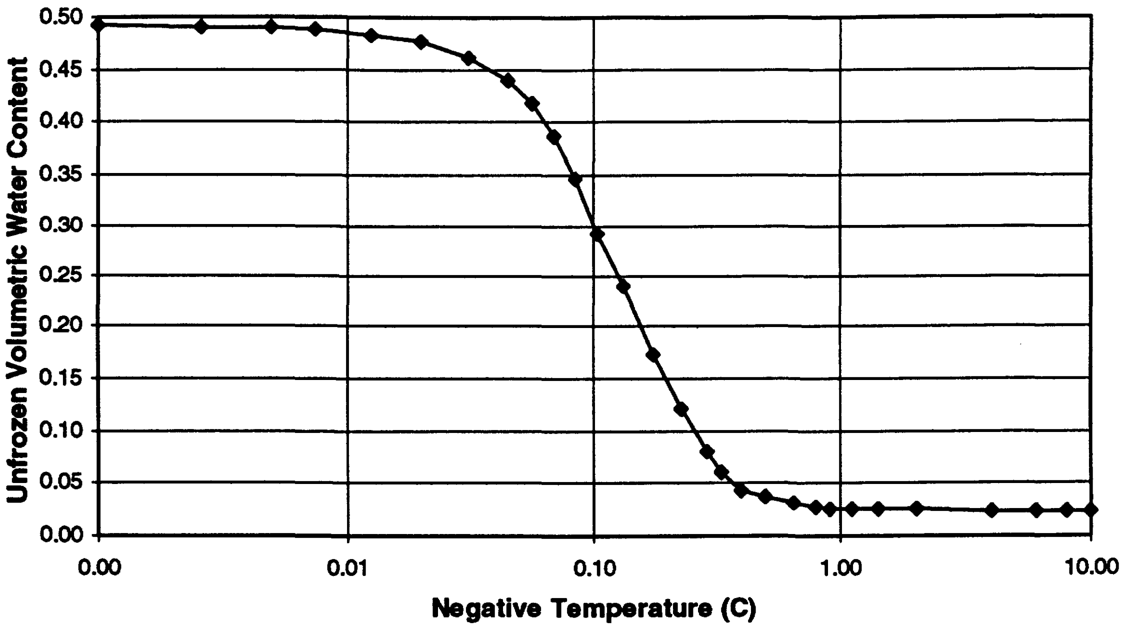


Figure 4.6 Semi - Log Plot of Soil Freezing Curve for Silica Flour

Data from the soil freezing curve given by Jame (1972) was used with a form of the Clapeyron equation given by Black and Tice (1989) to develop a theoretical soil water characteristic curve for the silica flour. This form of the Clapeyron equation relates matric suction to sub-zero temperatures in frozen soils dominated by capillary water forces. A comparison between the theoretical curve for Jame (1972) and the measured curve for this study is included in Figure 4.7.

The experimental soil water characteristic curve data was obtained using a modified Tempe cell with a 1 bar air entry disk. The 1 bar stone only permitted suctions up to 100 kPa to be applied to the sample. As a result, the suction values above 100 kPa matric suction were estimated and plotted using the Fredlund and Xing (1994) equation for the soil water characteristic curve. The estimated portion of the curve was selected such that it approximated the theoretical values and approached a zero water content at 1 million kPa matric suction. A sensitivity comparison was performed to determine the effects of changing the residual matric suctions, and the slopes of the linear portion of the curve. This is discussed later in Chapters 5 and 6.

The term 'G' was introduced into the modified heat transfer equation for a freezing soil (i.e., Eq. 3.12). The term 'G' is the ratio of change in matric suction and change in sub-zero temperature in a freezing soil and it can be computed by dividing the slope of the soil freezing curve by the slope of the soil water characteristic curve for any given unfrozen volumetric water content. As such, 'G' may be considered unique for any soil type. Figure 4.8 shows a linearized form of the 'G' term as a function of volumetric water content for the silica flour used in this study.

Fredlund et al. (1994) present an equation which predicts the permeability function for unsaturated soils using the soil water characteristic curve. The function is an integrated form of the suction versus water content relationship and can relate permeability to suctions or water contents from zero water content to saturation (or 0 kPa to 1 million kPa matric suction). Fredlund et al. (1994) verified the equation by fitting experimental data from various sources in the literature with accurate results. The equation was used in this study to predict a relative coefficient of permeability function based on the soil water characteristic curve. The relative coefficient of permeability versus matric suction relationship is shown in Figure 4.9.

The coefficient of permeability for the unsaturated soil was determined by multiplying the relative coefficient of permeability by the saturated coefficient of permeability, Ksat. A falling head permeameter was used to determine the saturated coefficient of permeability of the silica flour used in this study. The values of Ksat for the silica flour were found to range from 2.5×10^{-4} cm/s at a porosity of 0.52, to 3.0×10^{-5} cm/s at a porosity of 0.48.

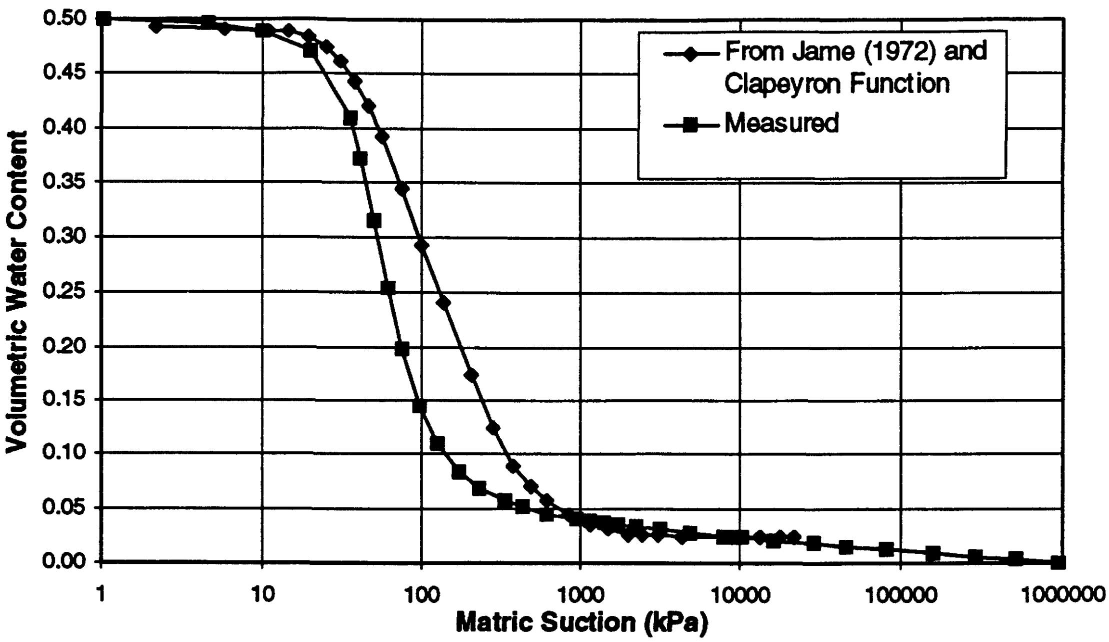


Figure 4.7 Theoretical and Experimental Soil Water Characteristic Curves (Measured Data Was Approximated at Suctions Above 100 kPa)

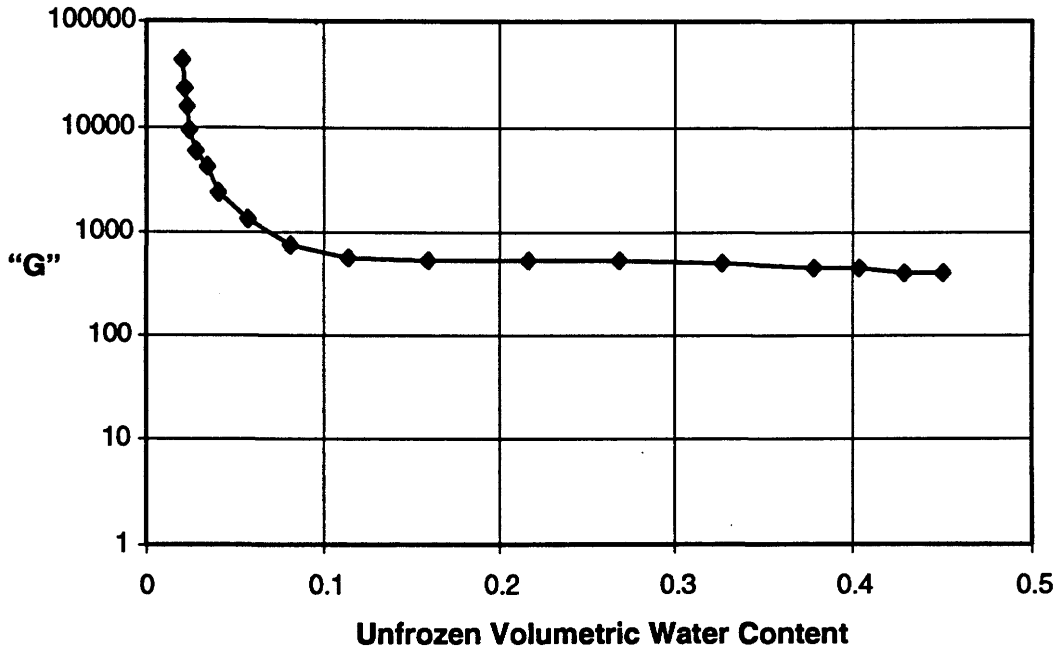


Figure 4.8 Ratio 'G' for Change in Matric Suction and Change in Sub-zero Temperature as a Function of Volumetric Water Content for Silica Flour

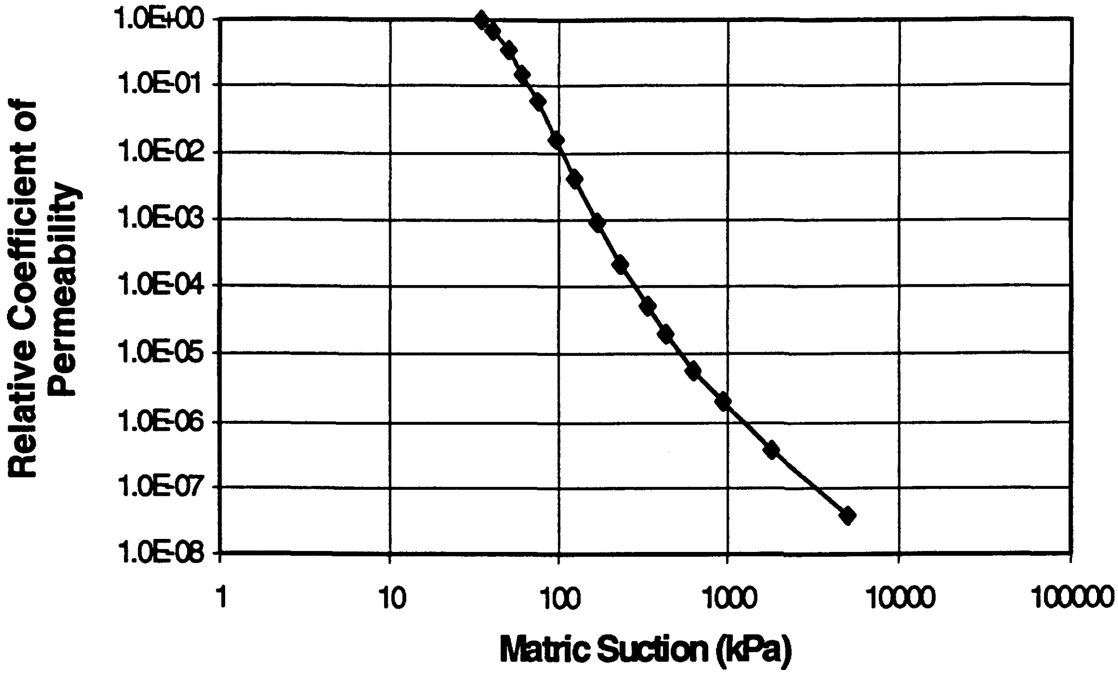


Figure 4.9 Relative Coefficient of Permeability for Silica Flour

Jame (1977) estimated the porosity of the silica flour material used to be 0.49. The soil water characteristic curve used in this study was measured at a porosity of 0.51. Because of this difference, three different saturated coefficients of permeability measured for the porosities in the range given above, were used during the computer simulations in this study. The sensitivity of the computed results with respect to small changes in the saturated coefficient of permeability for the silica flour is discussed in Chapters 5 and 6.

In numerous previous soil freezing heat flow and mass transfer models, researchers have adopted an impedance factor to account for the decreased water permeability in the frozen zone. They developed the impedance factor after initial computer simulations revealed too much ice was accumulating behind the frost front. Jame (1977) and Taylor and Luthin (1978) and others used an exponential impedance factor which was solely a function of volumetric ice content. In both these investigations, the freezing experiments performed by Jame (1977) were modelled using an impedance factor calculation of the form as follows:

$$k(\psi, \theta_i) = k(\psi) \times 10^{-(E\,\theta_i)} \qquad [4.1]$$

where,

$k(\psi)$ = the coefficient of permeability from the suction versus permeability data (m/s), and

E = an empirical constant equal to 12.

Applying this impedance factor with E = 12, has the effect of exponentially reducing the coefficient of permeability by three orders of magnitude as the volumetric ice content increases from 0.0 to 0.25. Figure 4.10 shows the exponential nature of the ice impedance factor for a range of empirical constants and material types as suggested by Gosnik et al. (1988).

There has been a great deal of criticism of the 'impedance factor' (Black and Hardenberg, 1991) as it is often considered a means of getting the model to fit the data. In this study various impedance factors were used for comparison purposes, ranging from E = 0 to E = 12. The results of this comparison are presented in Chapter 5 and discussed in Chapter 6.

The thermal conductivity versus water content relationship for an unfrozen sample of the silica flour is shown in Figure 4.11. This Figure also compares experimental results obtained by Jame (1977) with theoretical approximations obtained using the methods proposed by de Vries (1963) and Johansen (1975). The method given by Johansen (1975) is much easier than de Vries (1963) method and according to Farouki (1981) gives superior results for a wider range of soil types and water contents. As a result, the Johansen (1975) method for computing the thermal conductivity of a frozen soil was chosen in this study during computer simulations. The thermal conductivity in the unfrozen zone was obtained directly from the data of Figure 4.11. For details regarding the application of the Johansen (1975) method see Chapter 2.

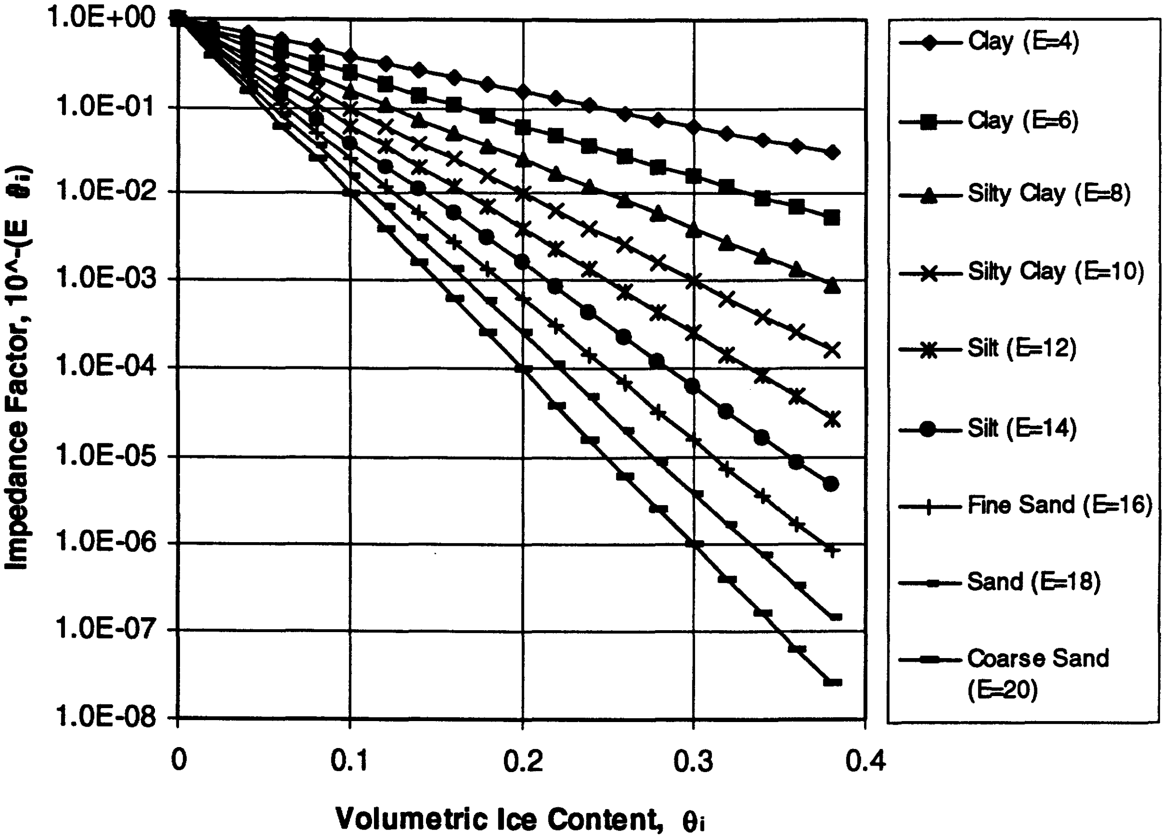


Figure 4.10 Coefficient of Permeability Ice Impedance Factors for Various Soil Types (after Gosnik et al. 1988)

Several parameters are required for the Johansen (1975) method thermal conductivity calculations. The thermal conductivity of the silica flour solid particle, λ_s, was assumed by Jame (1977) to be that of pure quartz at 8.54 W /m ^{0}C. Johansen (1975) suggested a value for λ_s of 7.7 W / m ^{0}C. In this study, a value of 8.12 W / m^{0}C seemed to give good agreement between measured and computed thermal conductivities as shown in Figure 4.11. The dry thermal conductivity of the mixture, λ_d, required in the Johansen (1975) method was measured by Jame (1977) to be approximately 0.25 W / m^{0}C.

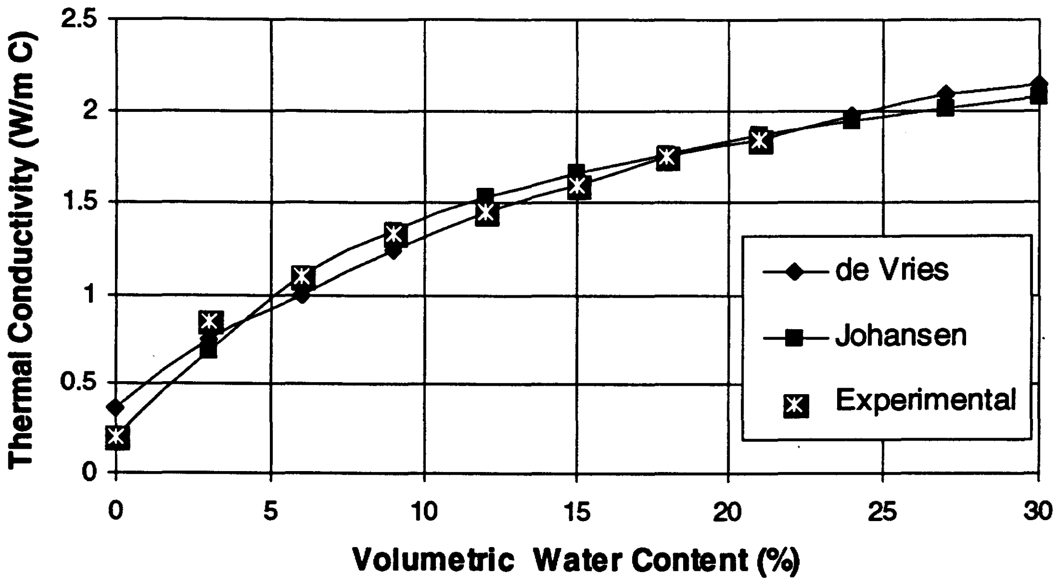


Figure 4.11 Thermal Conductivity for Unfrozen Silica Flour

Figure 4.12 shows the thermal conductivity of the frozen silica flour as a function of sub-zero temperature. The volumetric ice contents were obtained by subtracting the unfrozen water content (as given by the soil freezing curve -i.e., Figure 4.4) from an arbitrarily chosen initial water content for a range of temperatures below 0 °C. Figure 4.12 has no practical application as it was computed assuming no moisture transfer occurred. It is interesting, however, to see the wide range of thermal conductivities possible in a freezing soil, and that the thermal conductivity is influenced primarily by ice content.

The volumetric specific heat of the silica flour, liquid and ice mixture, ρc, was calculated using the following expression:

$$\rho c = \gamma_d\,(c_s + 4.184\ W_u + 2.10\ W_i) \qquad [4.2]$$

where,

ρc = the volumetric specific heat (J/m^3 C),

γ_d = the dry density of the silica flour (1330 kg/m^3),

c_s = the mass specific heat of the silica flour (0.837 J/g C),

4.184 = the mass specific heat of water (J/g C),

W_u = the gravimetric water content (dec.),

2.10 = the mass specific heat of ice (J/g C), and

W_i = the gravimetric ice content (dec.).

Figure 4.13 shows the volumetric specific heat for a range of unfrozen water contents computed using equation 4.2 without the ice content term. In the frozen zone, the computation of volumetric specific heat is obtained by including the mass specific heat of ice term in equation 4.2.

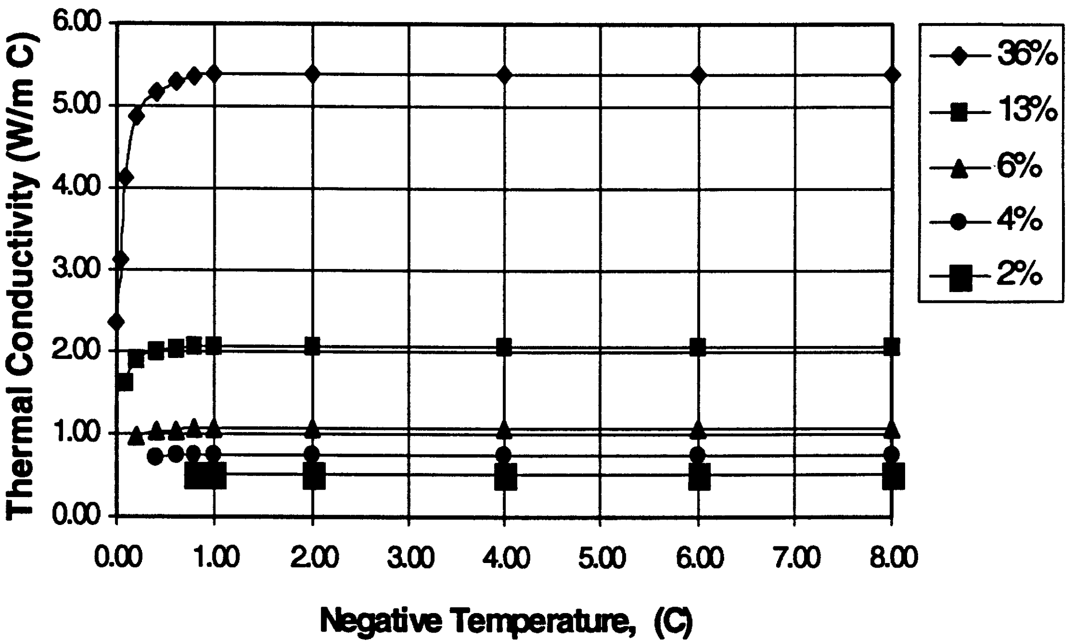


Figure 4.12 Thermal Conductivity for Frozen Silica Flour for Different Initial Gravimetric Water Contents and Neglecting Mass Transfer During Freezing

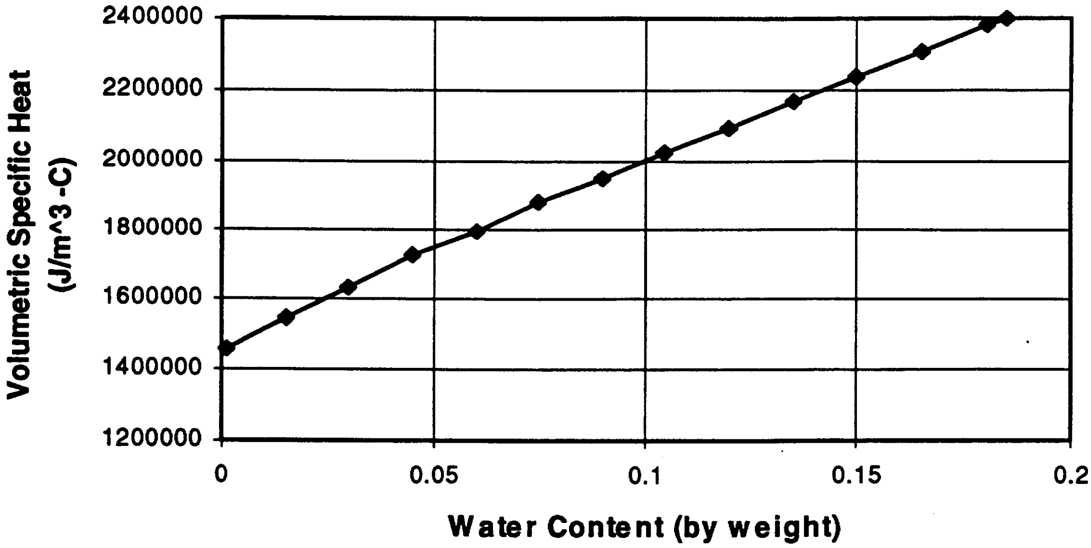


Figure 4.13 Volumetric Specific Heat in Unfrozen Soil

CHAPTER 5
PRESENTATION OF RESULTS

5.1 Introduction

This chapter presents the results of modelling the Jame (1980) laboratory freezing test data. In addition, the results obtained during calibration of the silica flour hydraulic properties are presented for discussion in Chapter 6.

5.2 Results of Modelling Jame (1980) Laboratory Data

The silica flour soil property functions and equations were incorporated into the computer program either as part of a data input file, or as part of a programming subroutine or function modification. To model the freezing tests conducted by Jame (1980) a cold end temperature boundary condition algorithm had to be added to the computer code because the temperatures at the cold end decreased from initial conditions to the prescribed cold end temperature over a period of 0.5 to 4 hours. SoilCover (MEND, 1993) presently does not allow hourly input data. The cold end temperature boundary conditions for each of the tests are given in Figure 5.1.

A finite element grid consisting of 31 nodes with even 1 cm spacings was used in all of the test verifications. A linear finite element was assumed with two Gauss points in each element. The system was considered to have converged if the suctions and temperatures did not change by more than 1% between successive iterations. Convergence was obtained at every time step in all tests. The Crank - Nicholson central difference time stepping routine scheme was used in the SoilCover program, and times steps were allowed to vary from 4 seconds to 1000 seconds. A time step control parameter limiting the change in time steps to a maximum of 4% between successive time steps was used. The average time to simulate a 72 hour freezing test was about 15 minutes. The first day took about 10 minutes to simulate, the second day took about 3 minutes to simulate, while the third day took about 1 minute to simulate. In general, the time steps became much larger as the system approached steady state. Finally, since the experiments performed by Jame (1977) were on a horizontal column of soil, the gravity term in the mass transfer equation was turned off in the computer program code.

As discussed in section 4.3, the precise saturated coefficient of permeability and ice impedance factors were not known prior to modelling. For this reason, modelling was carried out using a saturated coefficient of permeability ranging over one half an order of magnitude from 4.5×10^{-5} cm/s to 9.5×10^{-5} cm/s. Initially, an ice impedance factor was not applied. The results of the modelling using a saturated coefficient of permeability of 7.0×10^{-5} cm/s are illustrated in Figures 5.2 through 5.4 for the Jame (1980) tests 1 to 3 respectively. Results obtained during the calibration of soil hydraulic properties are presented in the next section.

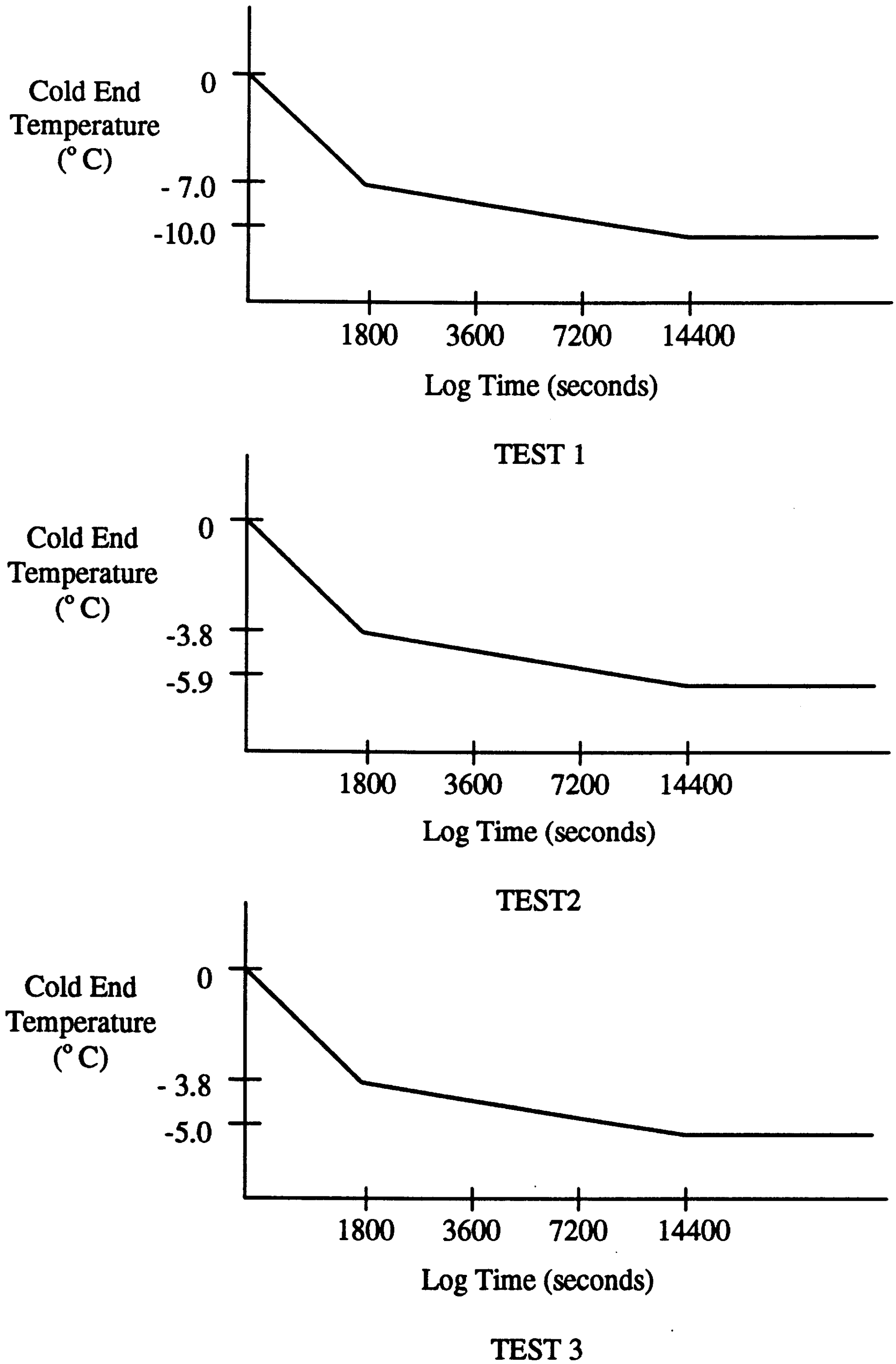
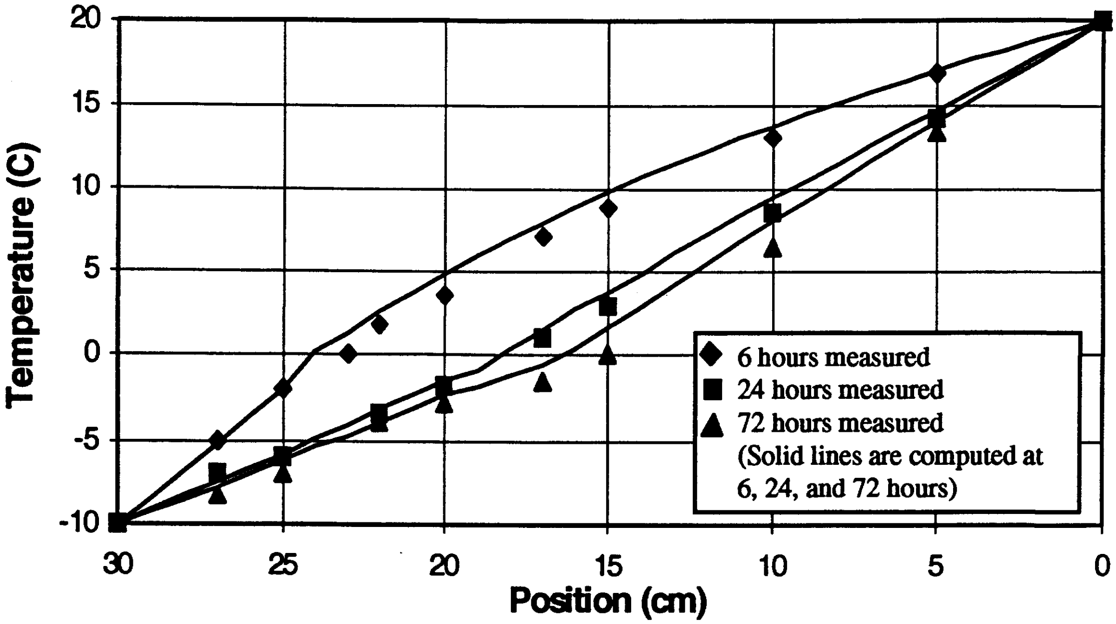


Figure 5.1 Cold End Temperature Boundary Conditions for Tests 1 - 3 (Jame, 1980)



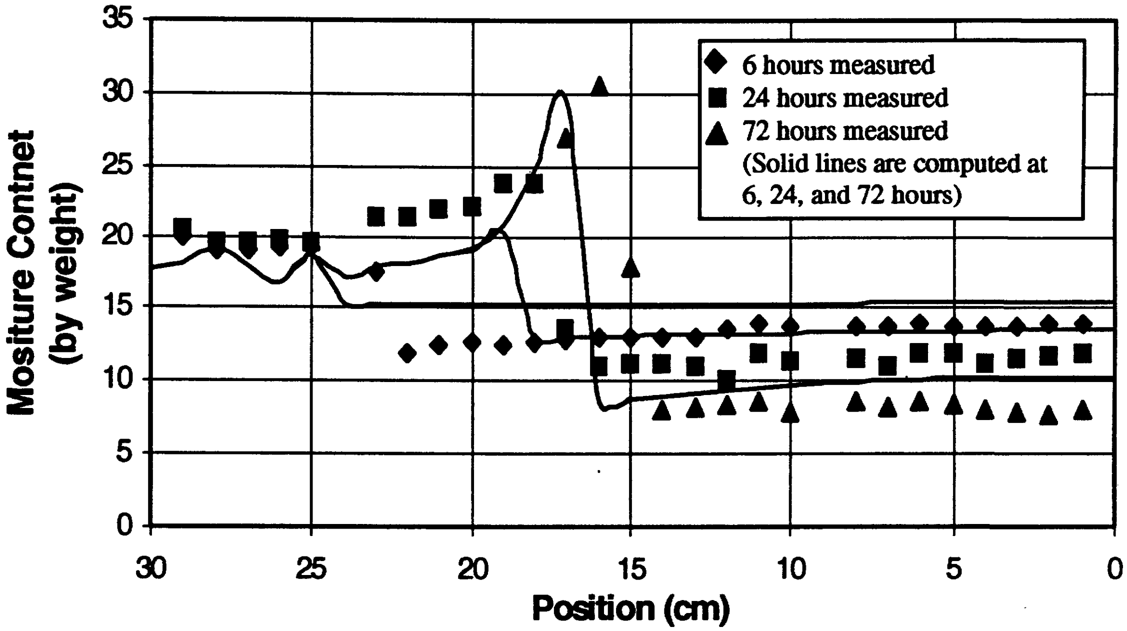


Figure 5.2 Modelling Results of Test 1

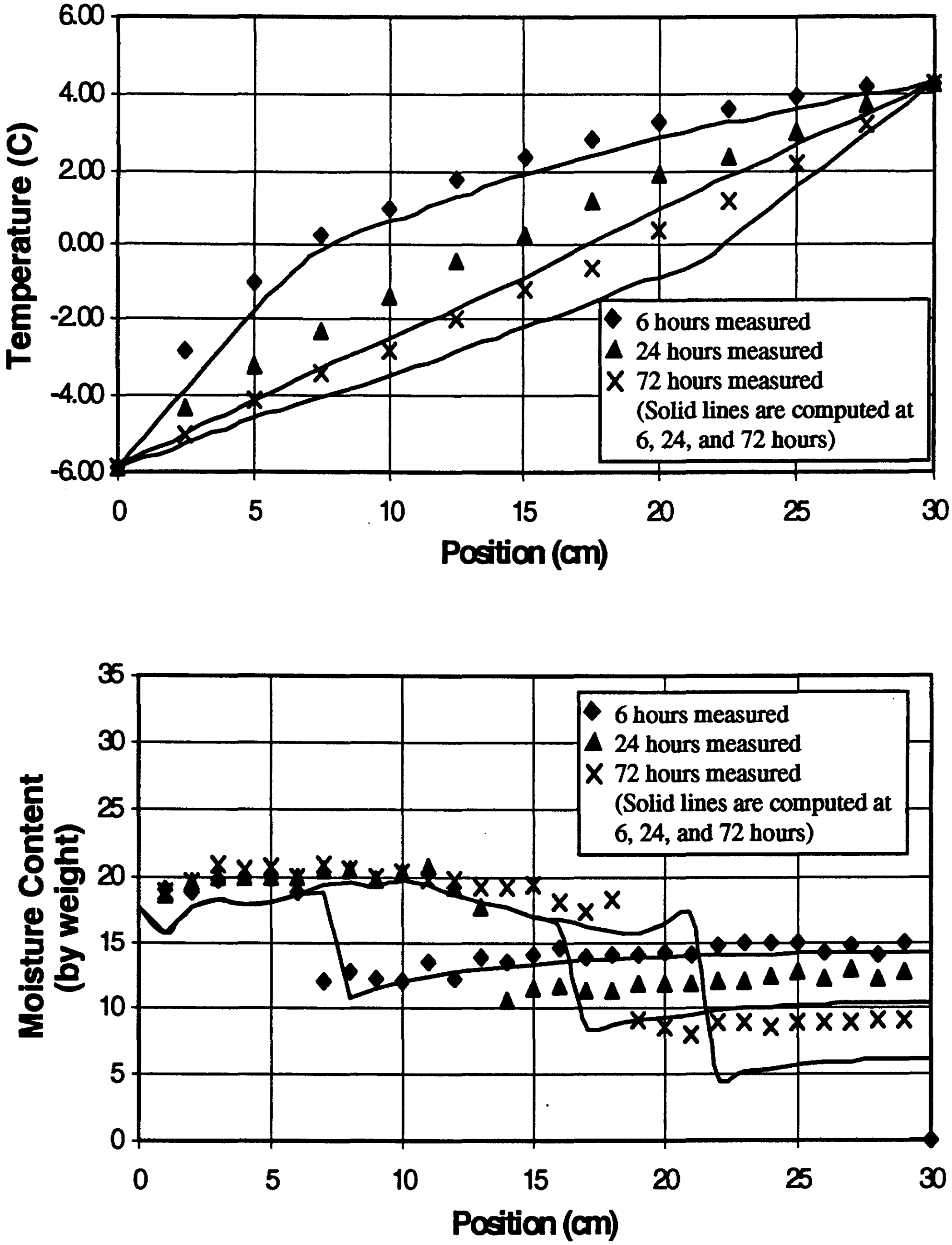


Figure 5.3 Modelling Results of Test 2

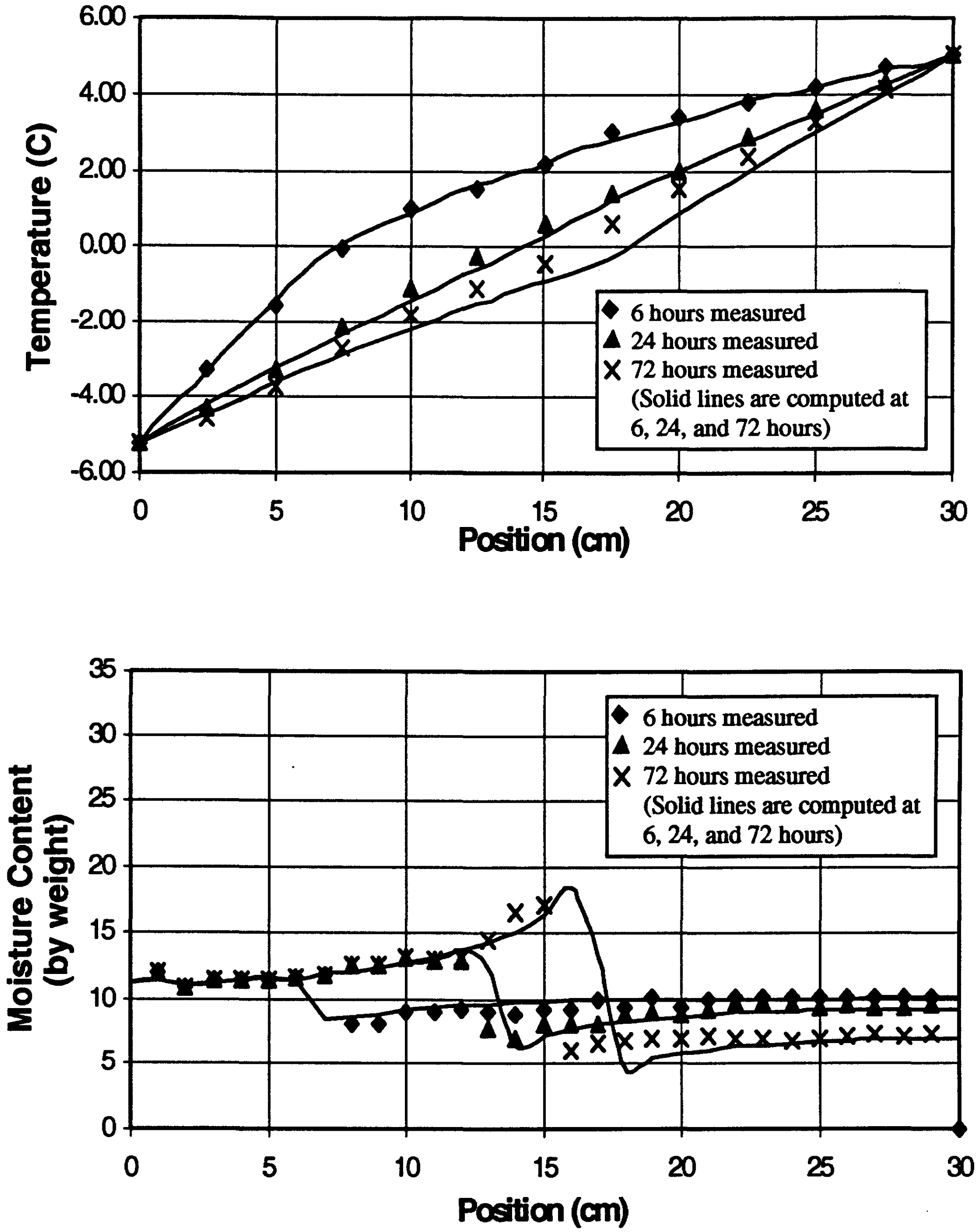


Figure 5.4 Modelling Results of Test 3

5.3 Calibration of Hydraulic Properties

In section 4.3, mention was made of the fact that the saturated coefficient of permeability was experimentally determined to be within the range of 2.5×10^{-4} cm/s at a porosity of 0.52, to 3.0×10^{-5} cm/s at a porosity of 0.48. The modified Tempe Cell test for the material used in this study indicated the porosity to be about 0.51. As a result, three different values for the saturated coefficients of permeability were selected for testing. All values were within half an order of magnitude of each other. It was also noted previously that the measured soil water characteristic curve did not account for suctions above 100 kPa. In hindsight, additional experimental testing should have been carried out to determine the volumetric water contents at higher values of suctions. Since this was not done, three different soil water characteristic curves were used for comparison purposes in this study. Finally, a range of ice impedance factors were applied to the permeability values to determine their significance and to obtain the correct empirical constant which could be used to calibrate the computer model for accurate simulation of the data presented by Jame (1980).

The saturated coefficients of permeability used in this study were 4.5×10^{-5} cm/s, 7.0×10^{-5} cm/s, and 9.5×10^{-5} cm/s. The three different soil water characteristic curves used are shown in Figure 5.5a. The corresponding relative permeability functions obtained using the Fredlund et al. (1994) equations are given in Figure 5.5b. The three different ice impedance factors are shown in Figure 5.6.

The saturated coefficients of permeability and soil water characteristic curves were chosen in such a way that they would adequately represent the measured soil property. The ice impedance factors were chosen such that they ranged from zero impedance to a three order of magnitude drop in permeability at a volumetric ice content of 0.25. Zero ice impedance would imply that the permeability given by the permeability versus suction relationship for an unfrozen soil would also apply in a frozen soil. A three order of magnitude drop in permeability impedance factor would be similar to that which Jame (1980) applied for simulating the experimental data in his study. In this study, every effort was made to avoid unreasonable adjustments of material parameters in order to obtain the desired results. Table 5.1 summarizes the numerical simulations using the various soil properties.

Table 5.1 Summary of Test Conditions Used in Calibration of Hydraulic Properites

Test Record Number	Jame Test Number	SWC Type	ksat. x 10^{-5} (cm / s)	Impedance Factor Empirical Constant
JT 102	1	1	4.5	0
JT 202	2	1	4.5	0
JT 302	3	1	4.5	0
JT 103	1	2	4.5	0
JT 203	2	2	4.5	0
JT 303	3	2	4.5	0
JT 104	1	3	4.5	0
JT 204	2	3	4.5	0
JT 304	3	3	4.5	0
JT 105	1	3	7.0	0
JT 205	2	3	7.0	0
JT 305	3	3	7.0	0
JT 306	3	3	7.0	6
JT 307	3	3	7.0	12
JT 108	1	3	9.5	0
JT 208	2	3	9.5	0
JT 308	3	3	9.5	0

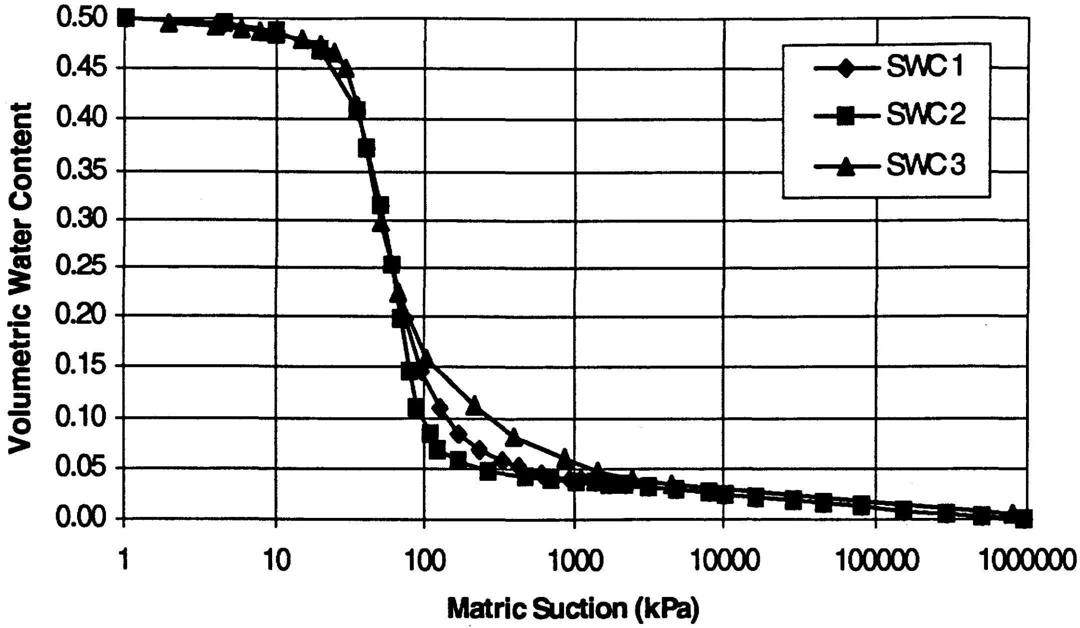


Figure 5.5a Soil Water Characteristic Curves Used In Calibrations

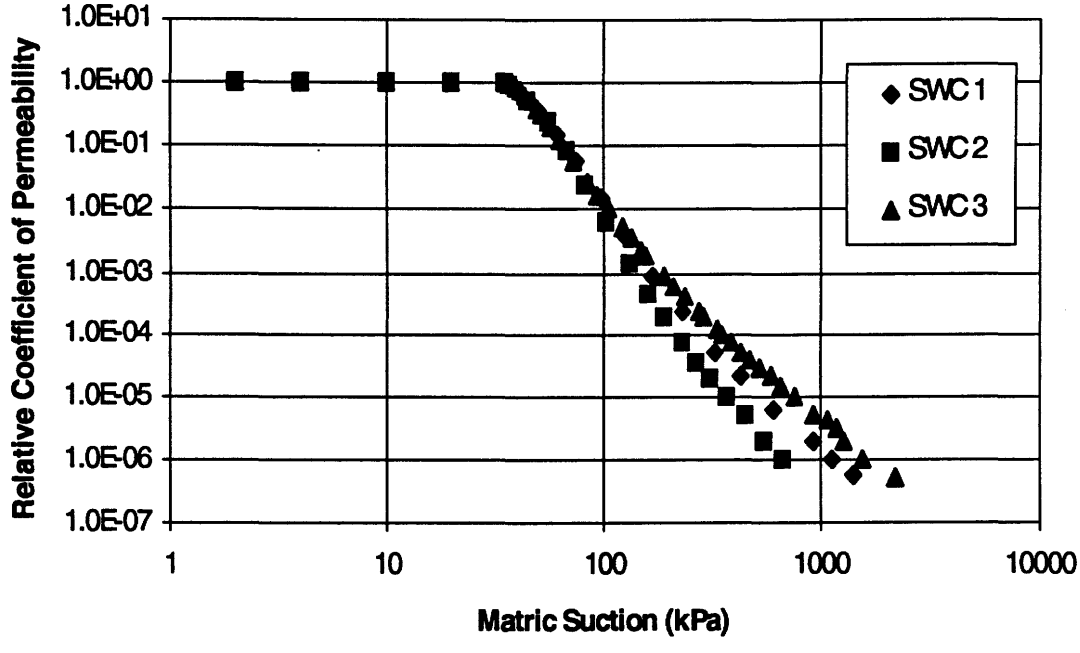


Figure 5.5b Relative Coefficient of Permeability Curves Obtained Using the Fredlund et al. (1994) Equations for the SWC Curves Used in Calibrations

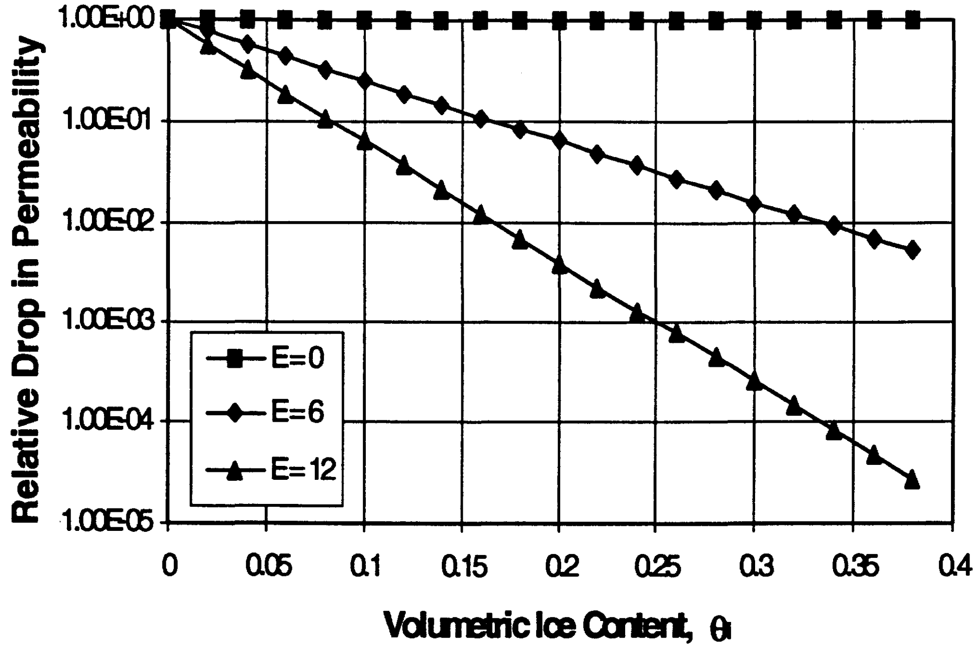


Figure 5.6 Range of Ice Impedance Factors and Relative Magnitude Applied in this Study

Simulations were carried out in the order they appear in table 5.1. Initially, the soil water characteristic curves were varied for each of the Jame (1977) freezing tests with an initial saturated coefficient of permeability of 4.5 x 10^{-5} cm/s and no impedance factor applied. These initial simulations indicated that an impedance factor was likely not necessary. After the initial nine simulations were complete (i.e., Table 5.1) the computed suction and temperature profiles showed that soil type 3 gave the most reasonable agreement with measured results when considering all three freezing tests. Using the soil water characteristic curve shown as soil type 3, six more simulations were performed to compare the effects of increasing the saturated coefficient of permeability. Once the most reasonable permeability was established at 7.0 x 10^{-5} cm/s, additional testing was done to study the effect of adding different ice impedance factors. The results of some of these tests are presented below. Comments and general observations about the results are presented in Chapter 6.

Figures 5.7 and 5.8 compare the temperatures and total moisture profiles simulated using the three slightly different soil water characteristic curves. These results were obtained for simulation of test 3, using a saturated coefficient of permeability of 4.5 x 10^{-5} cm/s and no impedance factor.

Figures 5.9 and 5.10 compare the temperature and total moisture profiles simulated with SWC 3 and three different saturated coefficients of permeability. Again, these results were obtained for simulation of test 3, using no impedance factor.

Figures 5.11 and 5.12 compare the temperature and total moisture profiles simulated with three different permeability ice impedance factors. These results were obtained for simulation of test 3, using SWC 3, and a saturated coefficient of permeability of 7.0 x 10^{-5} cm/s.

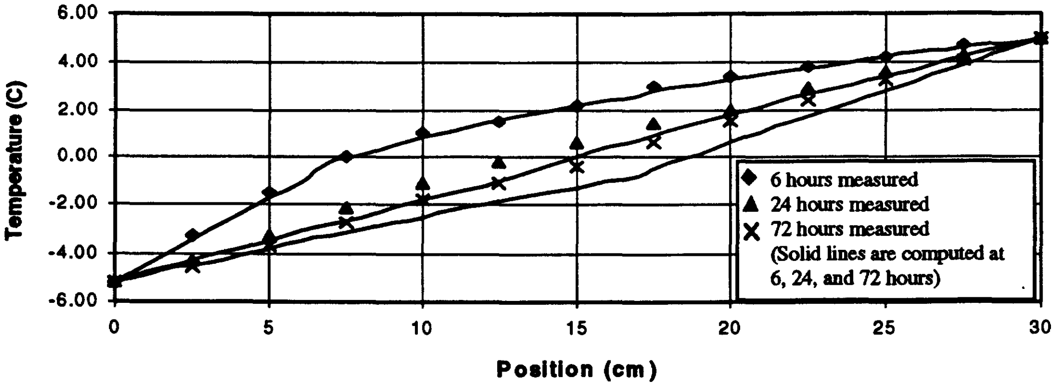


Figure 5.7a SWC 1

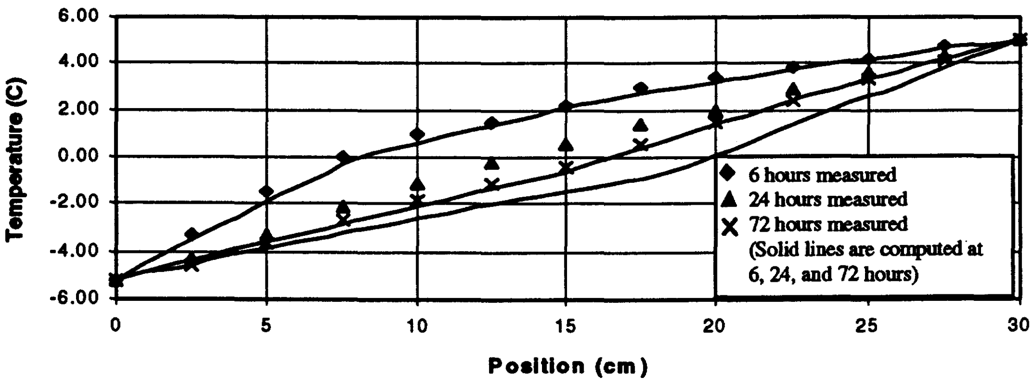


Figure 5.7b SWC 2

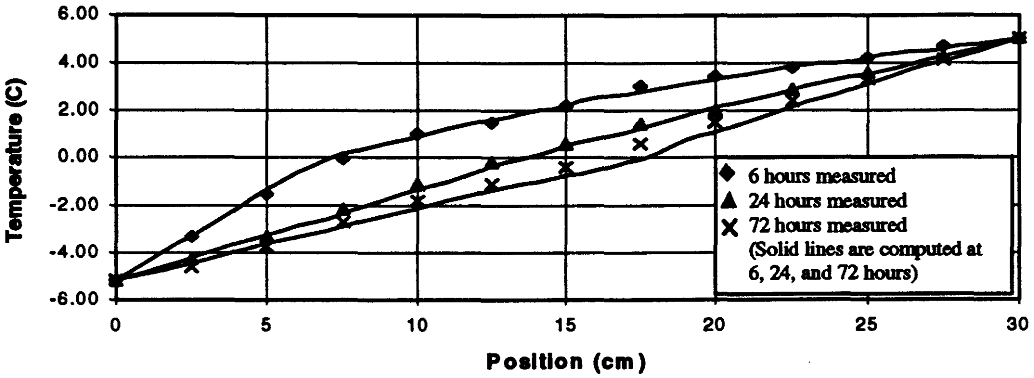


Figure 5.7c SWC 3

Figure 5.7 Comparison of Temperature Profiles for Test 3 Using Three Different Soil Water Characteristic Curves, ksat = 4.5×10^{-5} cm/s, and No Impedance Factor

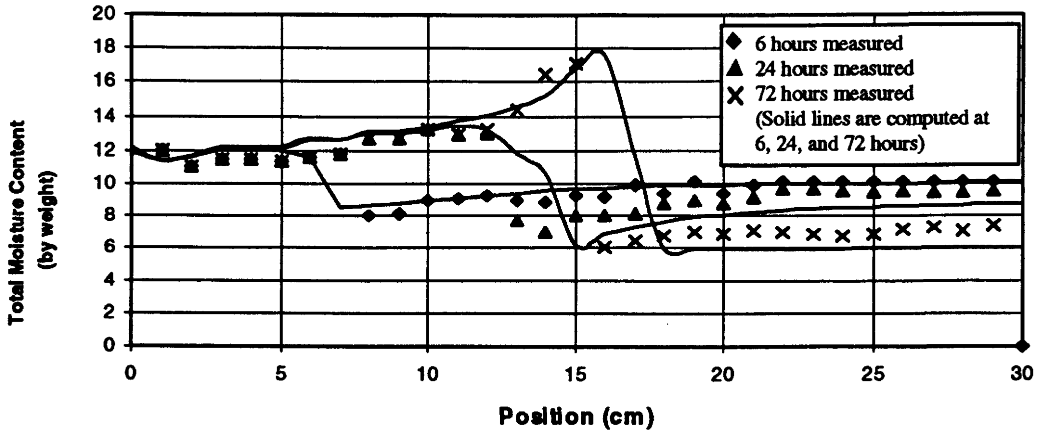


Figure 5.8a SWC 1

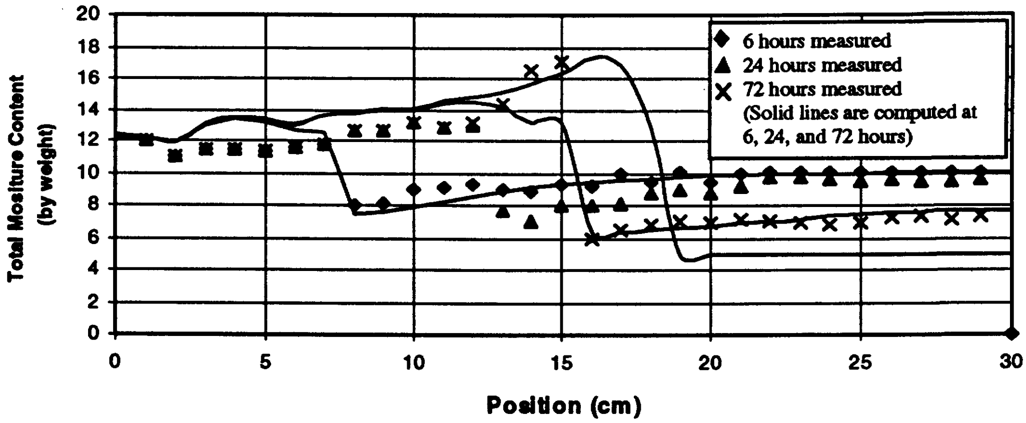


Figure 5.8b SWC 2

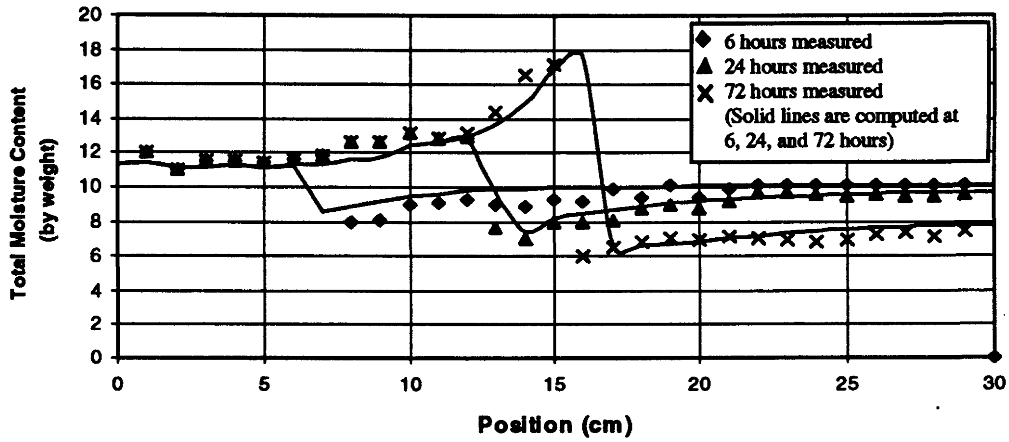


Figure 5.8c SWC 3

Figure 5.8 Comparison of Moisture Profiles for Test 3 Using Three Different Soil Water Characteristic Curves, ksat = 4.5 x 10^{-5} cm/s, and No Impedance Factor

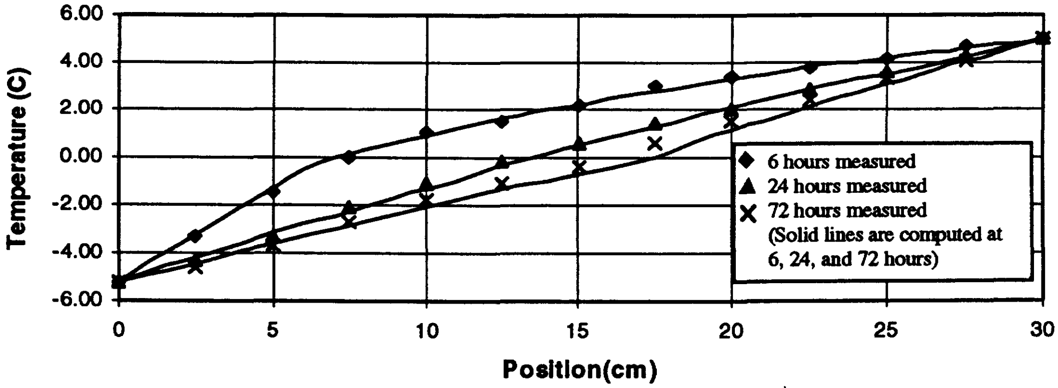


Figure 5.9a Ks = 4.5E-5

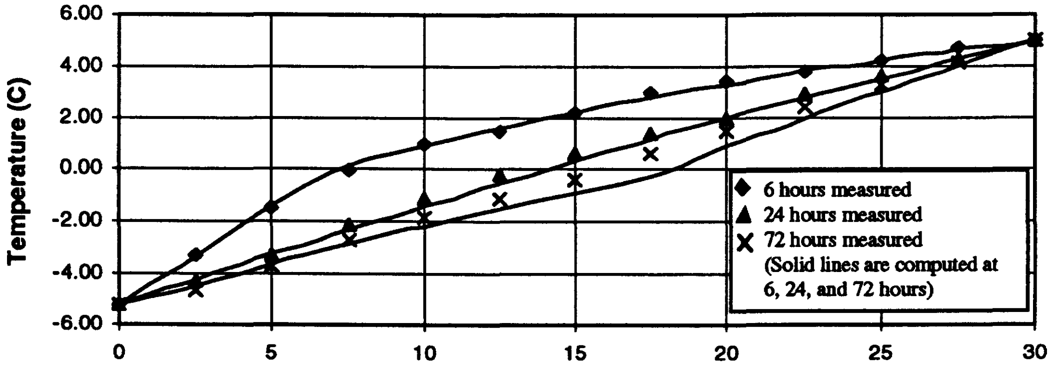


Figure 5.9b Ks = 7.0E-5

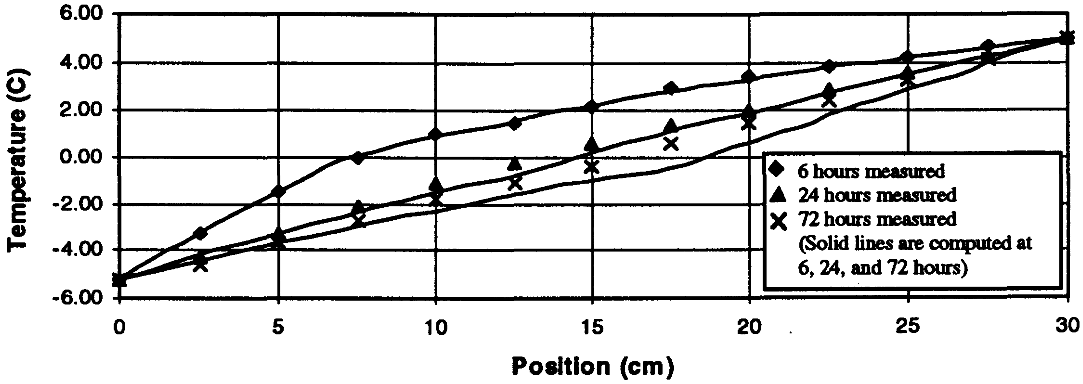


Figure 5.9c Ks = 9.5E-5

Figure 5.9 Comparison of Temperature Profiles for Test 3 Using Three Different Saturated Coefficients of Permeability, SWC 3, and No Impedance Factor

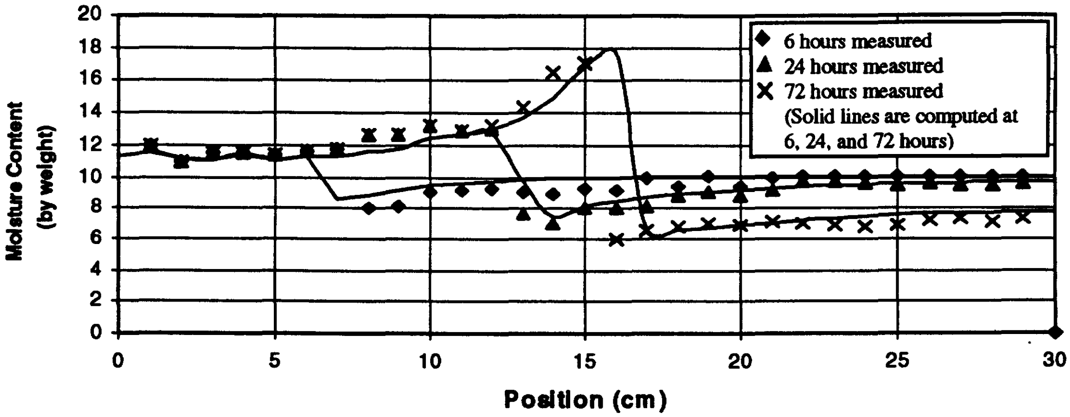


Figure 5.10a Ks = 4.5E-5

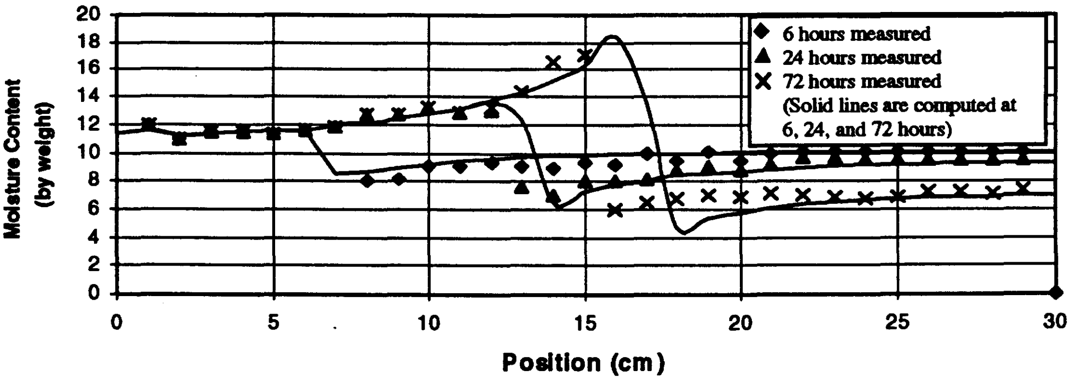


Figure 5.10b Ks = 7.0E-5

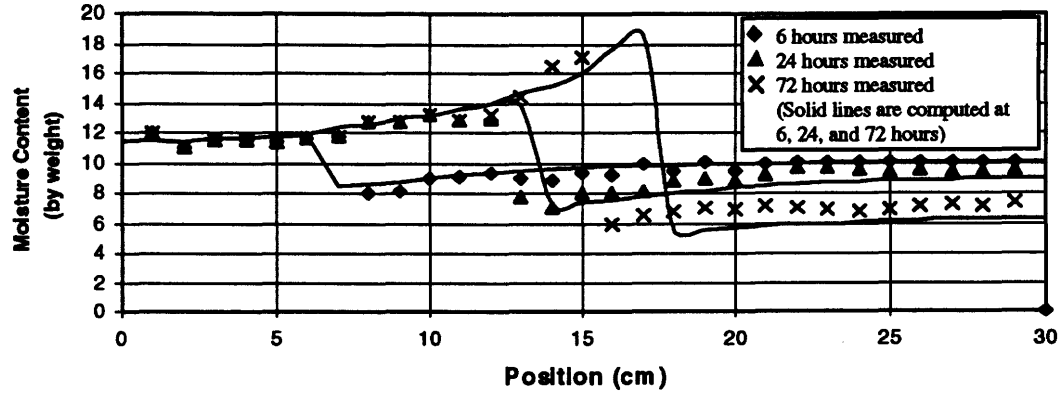
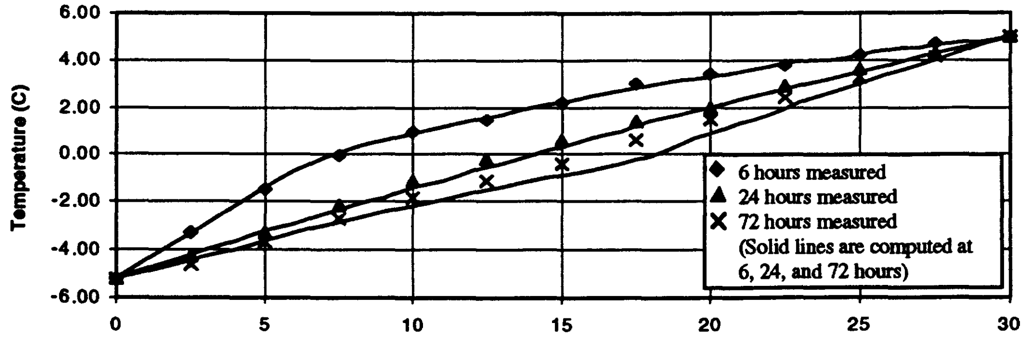
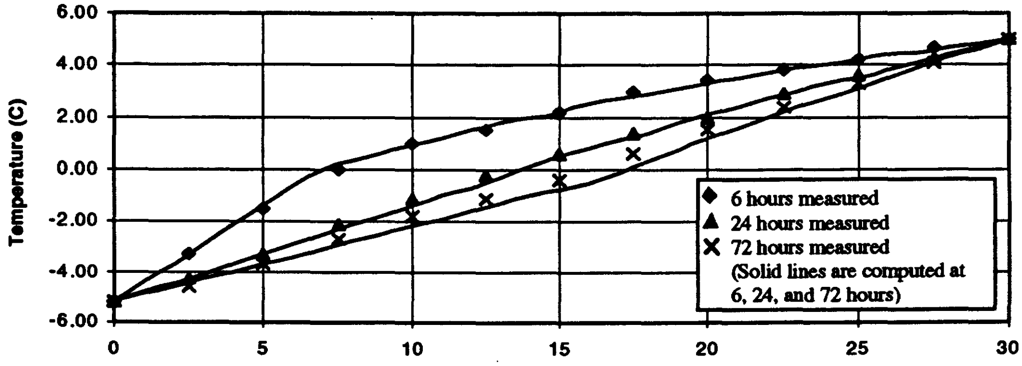


Figure 5.10c Ks = 9.5E-5

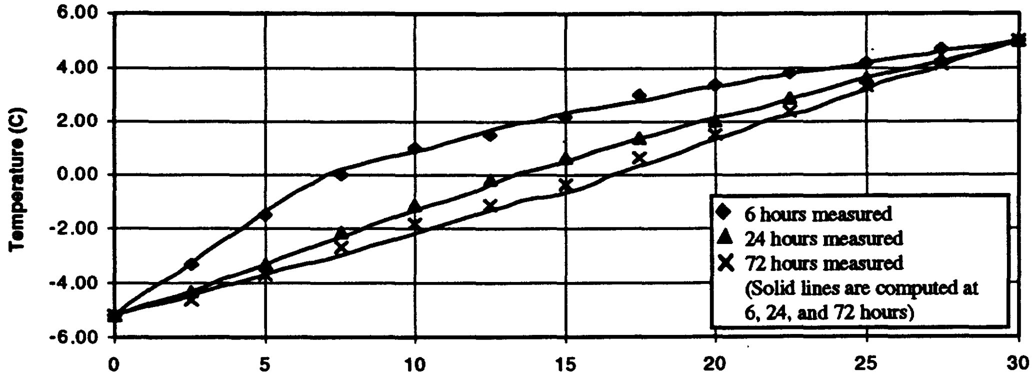
Figure 5.10 Comparison of Moisture Profiles for Test 3 Using Three Different Saturated Coefficients of Permeability, SWC 3, and No Impedance Factor



Position (cm)
Figure 5.11a E = 0



Position (cm)
Figure 5.11b E = 6



Position (cm)
Figure 5.11c E = 12

Figure 5.11 Comparison of Temperature Profiles for Test 3 Using Three Different Ice Impedance Factor Coefficients, SWC 3, and Ksat = 7.0 x 10^{-5} cm / s

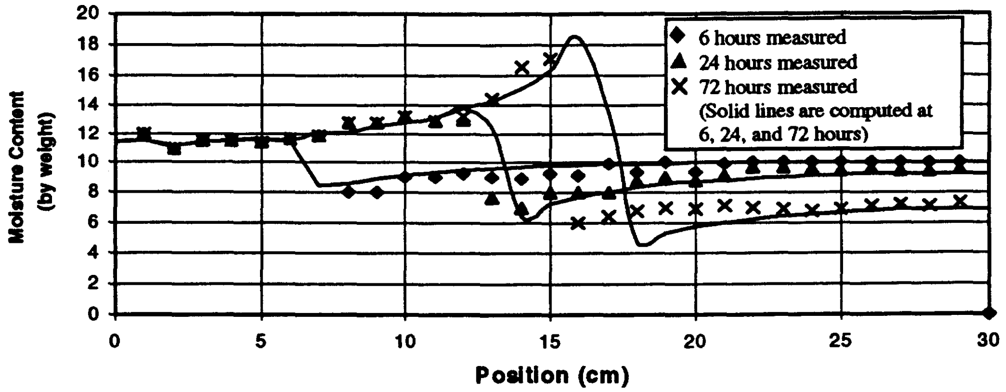


Figure 5.12a E = 0

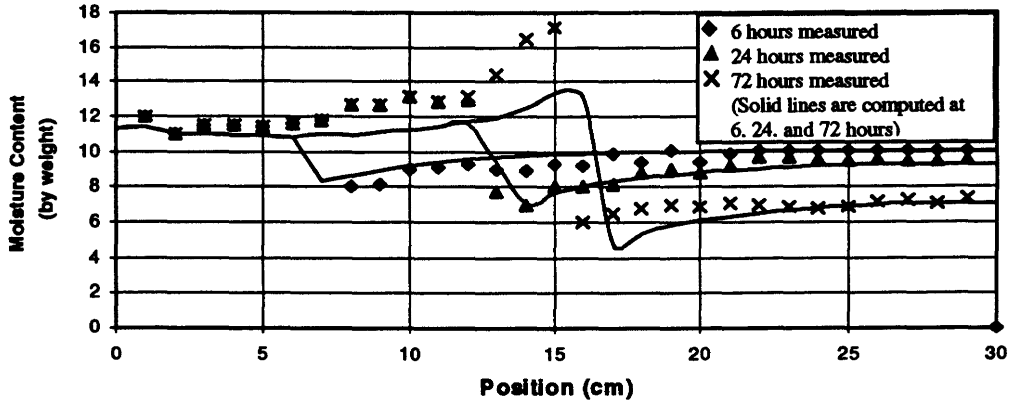


Figure 5.12b E = 6

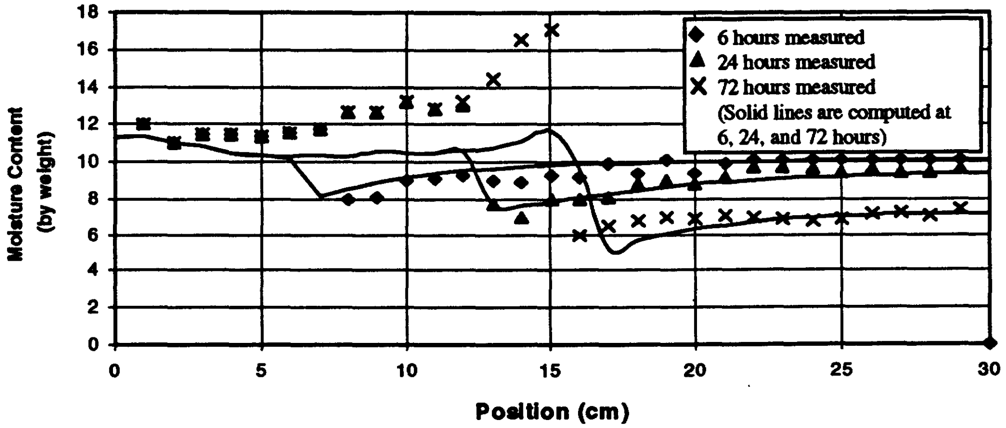


Figure 5.12c E = 12

Figure 5.12 Comparison of Moisture Profiles for Test 3 Using Three Different Ice Impedance Factor Coefficients, SWC 3, and Ksat = 7.0 x 10^{-5} cm / s

CHAPTER 6
DISCUSSION OF MODELLING RESULTS

6.1 Introduction

In Chapter 3 the theoretical framework for heat and mass transfer in freezing soils was presented. The computer program SoilCover (MEND, 1993) was modified to provide a numerical solution for the proposed theory. In Chapter 5 the results of the computer simulations for heat and mass flow in freezing soils under controlled were presented. The discussions in this chapter address several issues including the results of the laboratory data simulation program, and the advantages and limitations of the numerical model.

6.2 Advantages and Limitations of the Revised Numerical Solution

This section discusses some of the advantages and limitations of the revised heat and mass transfer model. The advantages deal mainly with how the phase change theory was incorporated into an existing non-freezing computer model. The limitations deal mainly with the assumptions used in developing the revised model. In addition, specific reasons for minor discrepancies between measured and computed results are discussed at the end of section 6.3 for the laboratory data modelling program.

In a non-freezing, unsaturated soil, the heat and mass transfer equations can be coupled by the water vapour pressure term which appears in each equation (Wilson, 1990). The dominant coupling term between the heat and mass transfer continuity relationships for freezing soils is the volumetric ice content. In the freezing case mass transfer continuity relationship, water that changes phase to ice is no longer considered free to transfer through the system. Furthermore, water that changes phase instantly introduces latent heat into the system at the point of phase change. The original SoilCover (MEND, 1993) program for non-freezing conditions was written so that the dependent variables used in the solution scheme are matric suction and temperature which are coupled together by the vapour pressure of the free water. The addition of the volumetric ice content variable causes problems to the numerical solution because the system of equations becomes indeterminate (i.e., there are two equations and three unknowns).

A common approach to the solution for this problem is to estimate an ice content for each new time step so that the corresponding latent heat of phase change and moisture sink quantities can be used to balance the heat and mass continuity equations over the next time step. This was the approach taken by various researchers who developed soil freezing models. For example, Jame (1977) estimated a new ice content for each time step by computing the heat transfer over the next time step assuming no moisture flux. A change in ice content was then back calculated based on the difference in unfrozen water contents between the current temperature and the estimated new temperature. The change in ice content was then applied to the main coupled heat and mass transfer equations and the procedure continued until convergence was achieved. This proved to be a cumbersome approach which required long computing times with convergence difficulties as the mass transfer component was neglected in the initial estimate of change in ice content.

In the revised numerical model, latent heat of phase change is applied to the heat transfer equation intrinsically because the mass transfer equation is incorporated in the heat transfer equation. There is no need to guess a change in ice content outside of the main solution algorithm and, as a result, the computing time is greatly reduced and convergence becomes a minor problem that is easily rectifiable by adjusting time step or convergence criteria. The actual change in ice content over the previous time step is calculated at the end of each iteration so that the soil thermal and hydraulic properties can be modified between iterations. Once the system has converged, the current change in ice content over the previous time step is computed and added to an total nodal ice content array.

The limitations of the revised model relate primarily to the assumptions made for the theoretical development. Convective heat transfer was omitted from the heat transfer equation. This is a reasonable assumption when modelling freezing and thawing in compacted fine materials. However, it is a questionable assumption when modelling freezing and thawing in less dense, coarser materials especially if high liquid fluxes are anticipated (i.e., from snow melt infiltration, or from large water sources at lower depths in an open system). The current application of the revised SoilCover model is to predict moisture redistribution throughout the winter in compacted clay covers over mine waste materials. For this application, it should be reasonable to neglect convective heat transfer.

Sublimation of ice (i.e., direct solid to vapour phase mass transfer) is also neglected in the frozen zone behind the freezing front. Sublimation depends on the partial vapour pressure difference between ice and the surrounding air, but the vapour pressure of ice is not included in the model formulation. If water changes phase to vapour due to a vapour pressure difference between the liquid water and air, then some of the ice must melt to increase the liquid water volume to that predicted by the soil freezing curve. This scenario is included in the model formulation.

The numerical model does not account for heat and mass transfer across a snow layer. However, a proven approach to this problem does not appear in the literature. Snow cover effects are a fundamental problem when modelling winter conditions in the field. Heat and mass transfer through snow layers is difficult to model because the physical and thermal properties of snow crystals are continually changing and this in turn changes the hydraulic and thermal properties of the snow layers. For example, the thermal conductivity of snow has been shown to range from 0.046 W/cmK for fresh, light snow, to

0.326 W/cmK for old, dense snow (Stepphuhn, 1981). If adequate meteorological data including snow depth and density are available, then the temperature of the soil surface beneath the snow pack can be approximated using a simple Fourier heat conduction formulation. Perhaps a simpler estimation of soil surface temperature can be obtained using data reported by Stepphuhn (1981). He reports that the difference in air to soil temperature across a snow layer in Eastern Europe ranged from 1.1 °C per centimeter of snow when the snow was 0 - 10 cm thick, to 0.1°C per centimeter of snow when the snow was 70 - 80 cm thick.

6.3 The Laboratory Data Modelling Program

The Laboratory data modelling program achieved three objectives listed below.

1) The simulation program verified that temperature and moisture content profiles measured by Jame (1980) could be simulated using the proposed theoretical approach.
2) The sensitivity analysis permitted some conclusions to be made about the sensitivity of the computed results to <u>small</u> changes in certain soil property input parameters.
3) The process examined the use of arbitrarily chosen ice impedance factors in soil freezing models.

6.3.1 General Comments Regarding the Simulated Temperature and Moisture Profiles

Figures 6.1 to 6.3 compare the computed and simulated temperature and water (liquid and ice) content profiles for the three freezing tests reported by Jame (1980).

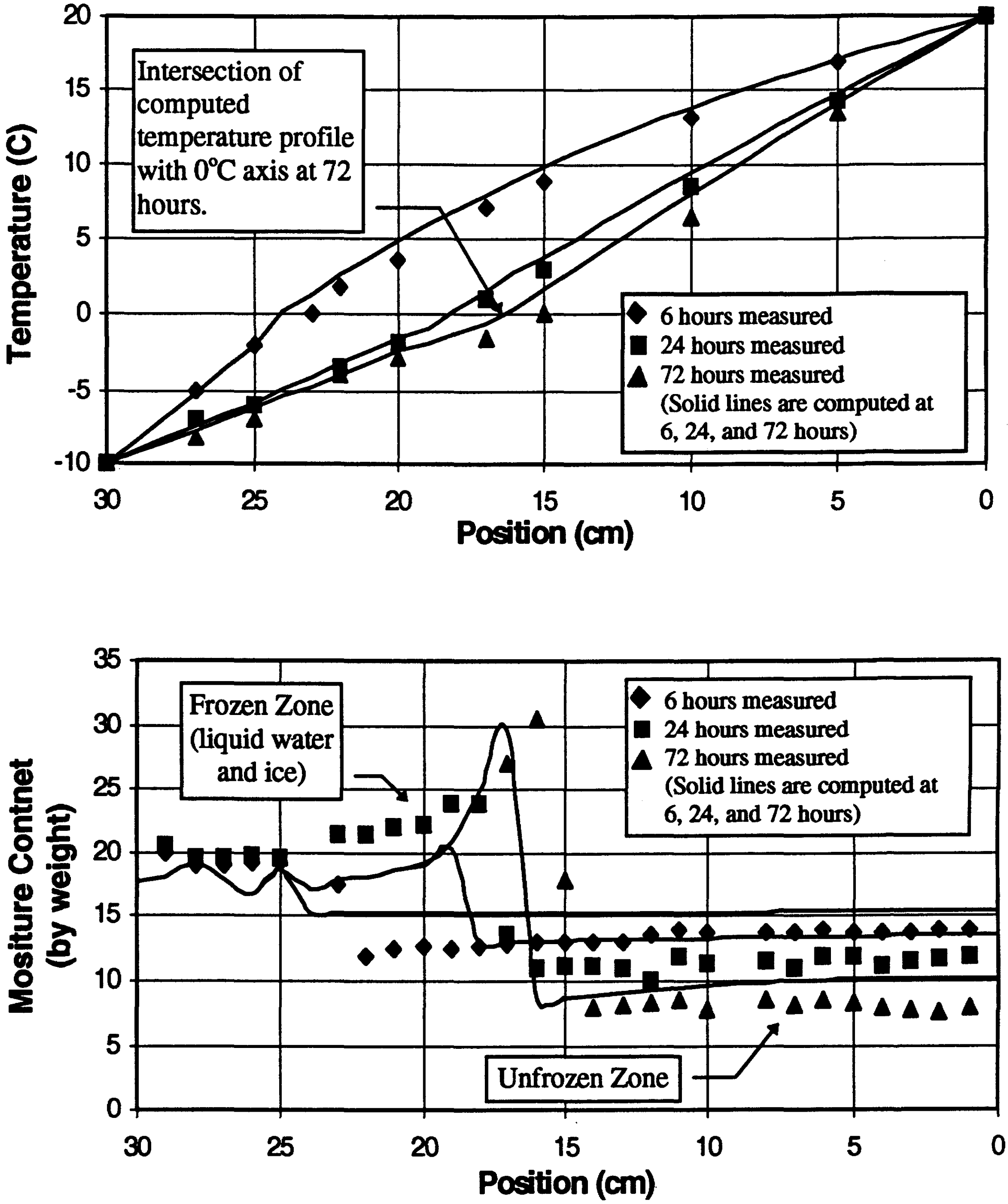


Figure 6.1 Modelling Results of Test 1

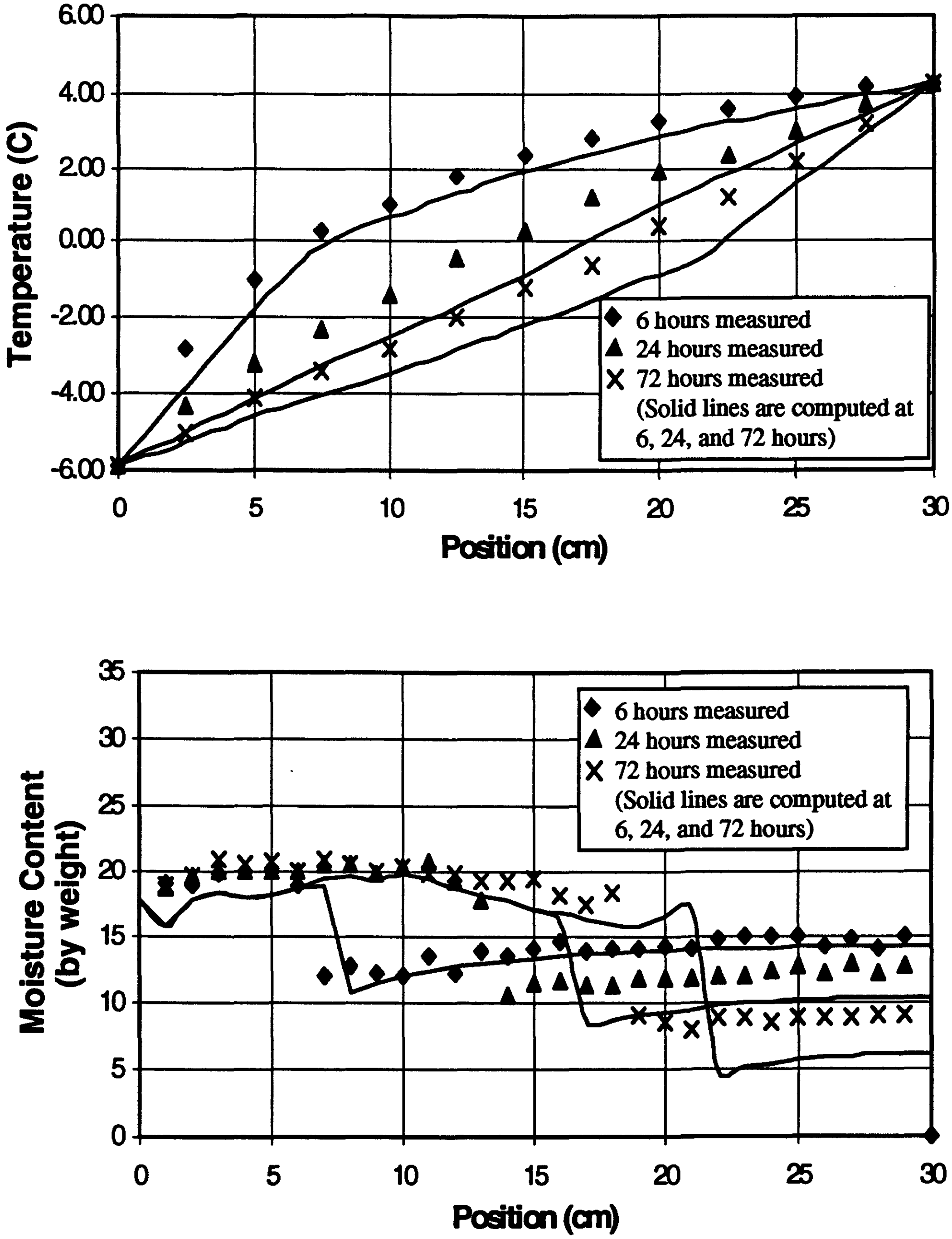
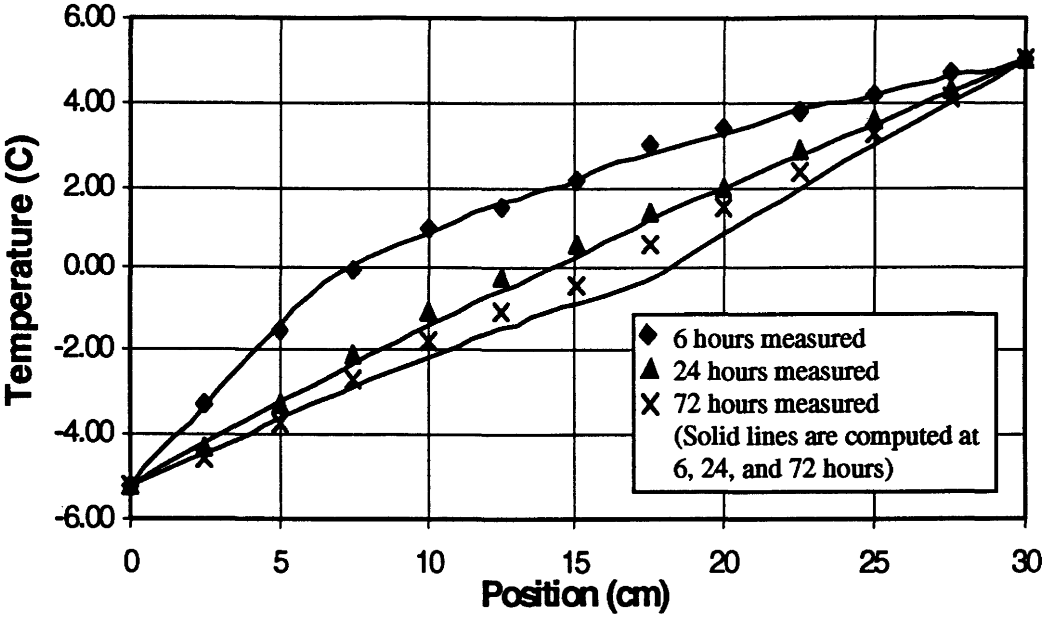


Figure 6.2 Modelling Results of Test 2



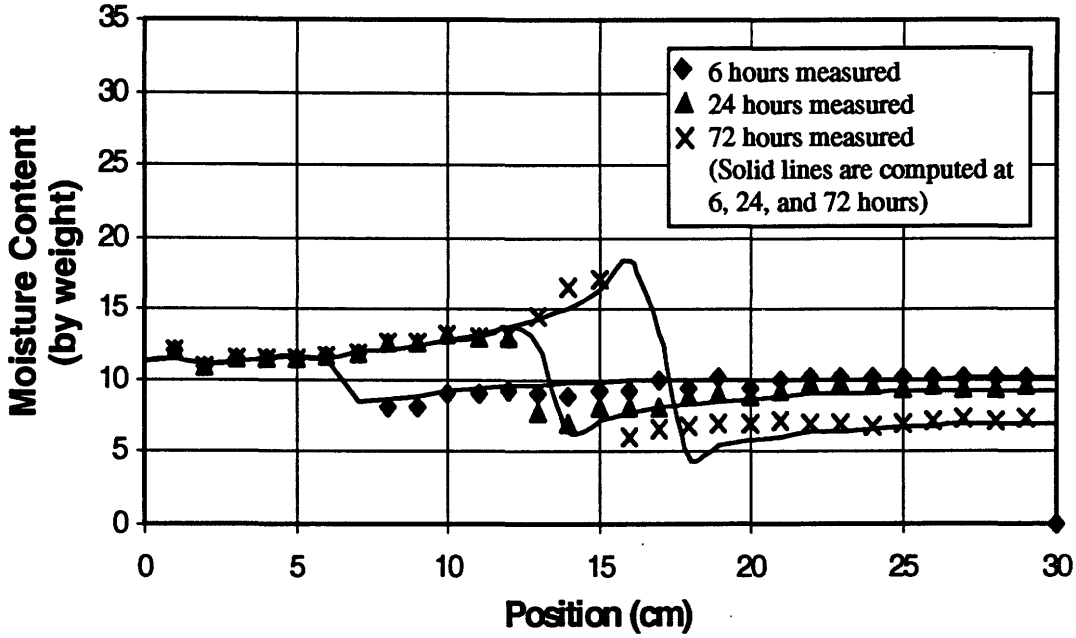


Figure 6.3 Modelling Results of Test 3

The simulated temperature and total moisture content profiles fit the experimental data with varied accuracy. The agreement between the computed and measured liquid water contents and temperatures seem to vary depending on the initial water content and temperature boundary conditions of the freezing test being simulated. Jame (1977) selected the initial and boundary conditions such that different temperature gradients were imposed on the soil at different initial water contents. He also ensured that the initial water contents were low enough to prevent frost heave from occurring in the silty material during closed system testing.

During freezing test 1 (Figure 6.1) the initial uniform temperature was 20 °C and the initial water content was 15.6 %. The cold end temperature was set at -10 °C which induced a thermal gradient of about 1 °C / cm throughout the horizontal column. In this simulation the computed temperatures are within 3.3% of the measured values at both 6 hour and 24 hour times. At 72 hours, the computed temperatures lag behind the measured values to a maximum of 5% at the frost front. The frost front is assumed to be the intersection of the computed temperature profile with the 0 °C axis in Figure 6.1.

The computed moisture content values are compared with measured ice and unfrozen water contents in the lower chart of Figure 6.1. In this figure, the agreement between measured and computed values is less accurate. The computed ice contents (i.e., left side of Figure 6.1) are a maximum of 18% lower than the measured values in the interval between 6 and 24 hours. Between 48 and 72 hours the advancing frost front appears to become stationary and an ice build up occurs. The quantity of computed ice at the frost front is within 3.3% of measured ice values except the frost front is positioned about 1 cm short of the measured frost front. The computed unfrozen water contents are 12% to 25% higher than the measured water contents at all times in the tests with the maximum difference occurring during the earlier stages of the simulation. Both the temperature and moisture profiles show excellent trend agreement between the measured and computed values.

The initial uniform temperature for freezing test 2 (i.e., Figure 6.2) was 5 °C and the initial water content was 15%. The cold end temperature was set to - 4.25 °C which resulted in a thermal gradient of 1/3 °C / cm throughout the column. The computed temperature profiles precede the measured profiles at all times during the simulation. At the 6 hour interval the maximum difference between measured and computed temperatures is 8.3%. At 72 hours the computed temperature profile precedes the measured profile up to a maximum of 2 cm (or 6.6%) at the intersection of the temperature line with the 0 °C axis.

The computed ice content (i.e., left side of lower chart in Figure 6.2) varies from the measured ice content to a maximum of 7.5% during the simulation. The difference between the computed and measured frost front positions increased during the simulation to a maximum of 2 cm (or 6.6%) at the end of the test. Figure 6.2 shows good agreement (i.e., less than 5% error) between computed and measured unfrozen water contents earlier in the simulation, but by 72 hours the percent difference between computed unfrozen water contents and measured water contents is about 30 % . Again, there is excellent agreement between measured and computed temperature and moisture content trends, even to the extent that both the measured and computed ice content profiles show a small build up of ice at 72 hours when the advancing frost front became somewhat stationary.

The measured results of Test 2 significantly differ from test 1 in one way. Both tests were carried out using a sample with an initial water content of about 15%. However, in test 1 there was a higher thermal gradient across the column (i.e., 1 ° C / cm for test 1 compared with 1/3 °C / cm for test 2). The higher thermal gradient in test 1 seemed to cause a large increase in ice content at the frost front, whereas this did not occur in test 2. At the higher thermal gradient the frost front advanced more slowly and allowed moisture to transfer from the unfrozen zone towards the frost front. Common sense would suggest that the frost front would advance faster at higher thermal gradients, however, this was not the case. This was due to the fact that in test 1, the warm end temperature (i.e., 20

°C) was twice the magnitude as the cold end temperature (i.e., - 10 ° C). Thus, even though the thermal gradient was higher in test 1, more heat had to be removed in test 1 and the frost front advanced more slowly.

The results of freezing test 3 are presented in Figure 6.3. In this test the initial uniform temperature was 5 ° C and the initial water content was 10%. The cold end temperature was set to - 5 ° C which resulted in a thermal gradient equal to 1/3 ° C / cm. During this simulation the maximum difference between computed and measured temperature values was 8% which occurred at the duration of the simulation. At the 6 and 24 hour intervals the maximum difference between computed and measured temperatures is 3%. The computed temperature profile preceded the measured temperature profile by about 1 cm at the 72 hour mark, and as a result, the frost front in the computed moisture profile is also about 1 cm (or 3.3%) ahead of the measured frost front at the 72 hour mark. The maximum difference between computed and measured moisture contents is 7.5% at the 72 hour mark, while the maximum difference at all other times in the simulation is 3%.

During test 3 there appeared to be a small build up of ice at the frost front in the later stages of the test. The thermal gradient imposed on the sample in test 3 was the same as that of test 2 which showed no ice build up. However, in test 3 the initial water content was only 10% as compared with 15% for test 2. The lower permeability associated with the lower water content in test 2 did not permit as much water flux to the frost front in the early stages of the test. As a result, the frost front advanced rapidly until it approached a thermal steady state condition, at which time water slowly made its way to the frost front and accumulated as ice.

6.3.2 Reasons for the Discrepancy Between Computed and Measured Results

The differences between computed and measured temperature and moisture content profiles can be attributed to several factors. These factors fall into two categories: numerical solution technique approximations, and soil property function accuracy.

Numerical factors are discussed next, and the sensitivity of the computed results to various soil property factors is discussed in the following section.

In the SoilCover finite element computer algorithm, the element stiffness and mass storage matrices are developed at every Gauss point in every element, starting at the ground surface (i.e., element # 1) and proceeding deeper into the soil. The freezing point depression temperature is determined at each Gauss point based on the local liquid water content at the end of the last time step. If the new temperature at that Gauss point is below the freezing point temperature, the modified heat equation is turned on and the mass transfer equation is turned off. If the new temperature is above the freezing point then the element stiffness and mass storage matrices are formulated using the non-freezing coupled heat and mass transfer equations.

By observing computed Gauss point temperatures and suctions during a simulation, it was noticed that the temperature profile would advance rapidly through the soil until the water at a Gauss point location would start to freeze. At that point, the latent heat of phase change released into the system slowed the advancing cold front and ice would build up. The cold front would then start to advance rapidly again until the next Gauss point temperature was low enough for freezing. In this way, the advancing cold front seemed to speed up, then slow down, then speed up etc. The simulated freezing process was not continuous because Gauss points are located a finite distance from each other. The discontinuous nature of the finite element formulation geometry introduces some error in the computed results.

Another problem related to the finite element formulation geometry is that the program is not able to accurately predict suction values just ahead of the advancing frost front. The numerical model uses suction values to estimate the soil properties at the Gauss points between nodes, but it does so without knowledge of the exact location of the frost front in this region. In the finite element formulation, the Gauss point suctions are estimated based on the suctions at the previous and adjacent nodes. This estimation process can result in suctions which are too high in the zone immediately ahead of the frost front (i.e., there are high suctions even though the temperature has not lowered to the point when ice forms). Figure 6.4 illustrates the potential problems introduced by erroneous Gauss point property estimations. In this figure, ice is assumed to form at 0°C. The estimated Gauss point ice content profile shows similar problems as the suction profile for the case where one node has ice build up and the adjacent node does not. In general, the numerical model can not determine the location of the frost front between adjacent nodes.

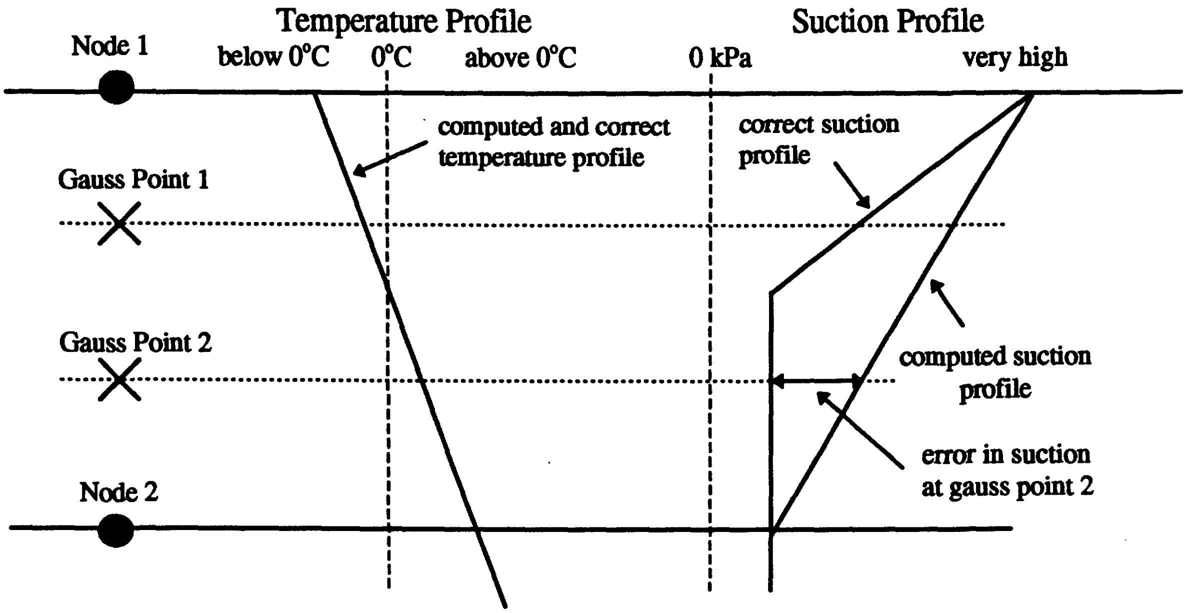


Figure 6.4 Problems With Numerical Gauss Point Suction Estimations

By observing computed Gauss point temperatures and suctions during a simulation it was noticed that the temperature profile through the Gauss points was estimated with good accuracy, but the suction profiles were often erroneous. An overestimate of suctions just ahead of the frost front resulted in a lower than actual estimate of coefficient of permeability. In return, less water flowed to the frost front to change phase and release latent heat. This in turn resulted in a frost front which advanced too rapidly. In other words, less heat was put into the system to slow the frost fronts advance. Clearly, the computed results of test 2 and test 3 (i.e., Figure 6.2 and 6.3) show a frost front slightly ahead of the actual frost front. When the frost front advanced too rapidly, too many nodes would change phase and as a result the suctions in the unfrozen zone would tend to get too high. This, in turn, resulted in lower than actual computed water content values in the unfrozen zone.

The finite element method (or any numerical procedure) can only be used to approximate a physical system. The results presented above clearly show that some errors are inherent in the finite element formulation when modelling a rapidly advancing cold front with high moisture redistribution.

6.4 The Sensitivity of Computed Results to Soil Property Functions

Other factors accounting for differences between computed and measured results are related to the soil property functions used in the simulations. Figure 6.5 shows three slightly different soil water characteristic curves used in the sensitivity analysis. The curves have the same air entry values and approximately the same values of residual matric suction. However, the three curves have slightly different radii of curvature near their residual water contents. Figures 6.6 to 6.8 show the changes in computed results obtained by making small changes to different soil property functions. Figure 6.6 shows the computed results for the same freezing test simulated with the three soil water characteristic curves.

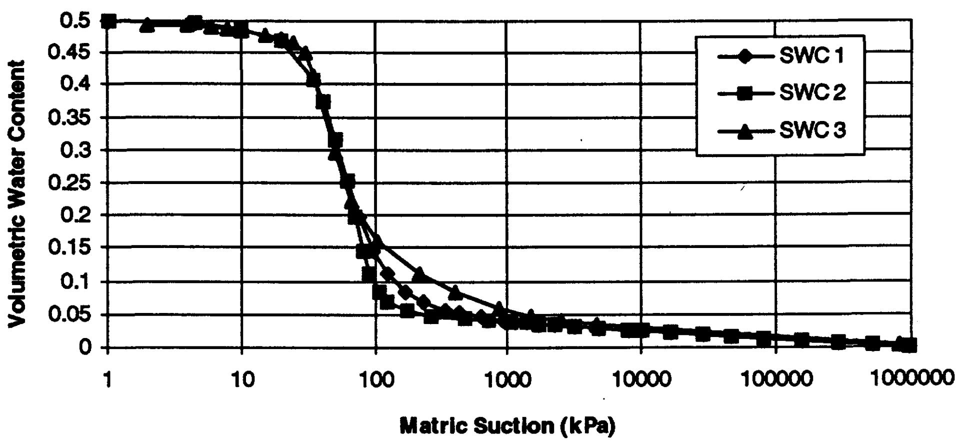


Figure 6.5 Soil Water Characteristic Curves Used in Sensitivity Study

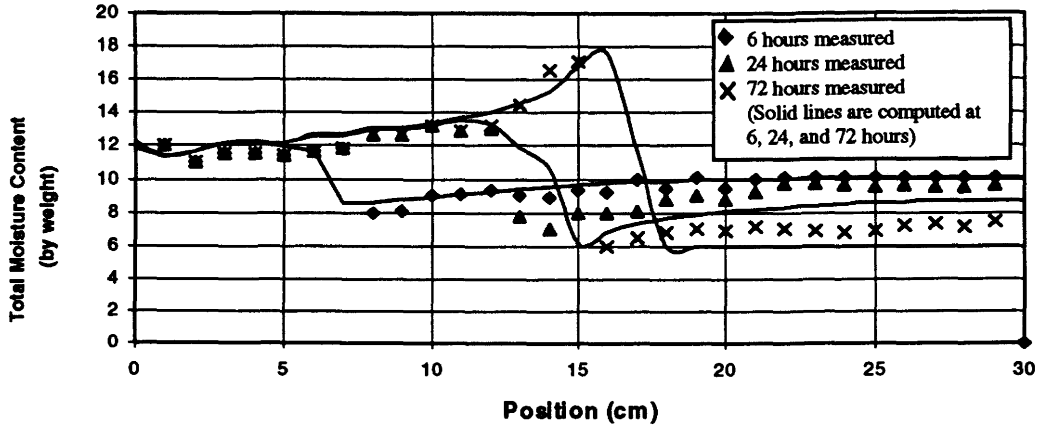


Figure 6.6a SWC 1

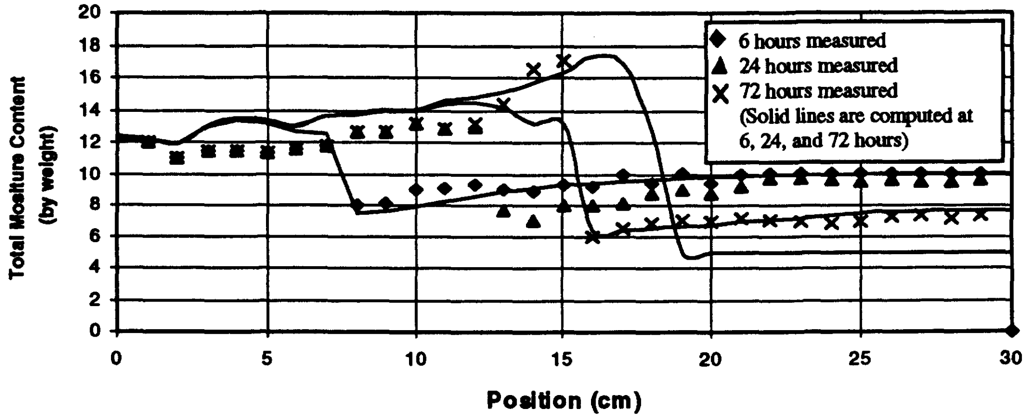


Figure 6.6b SWC 2

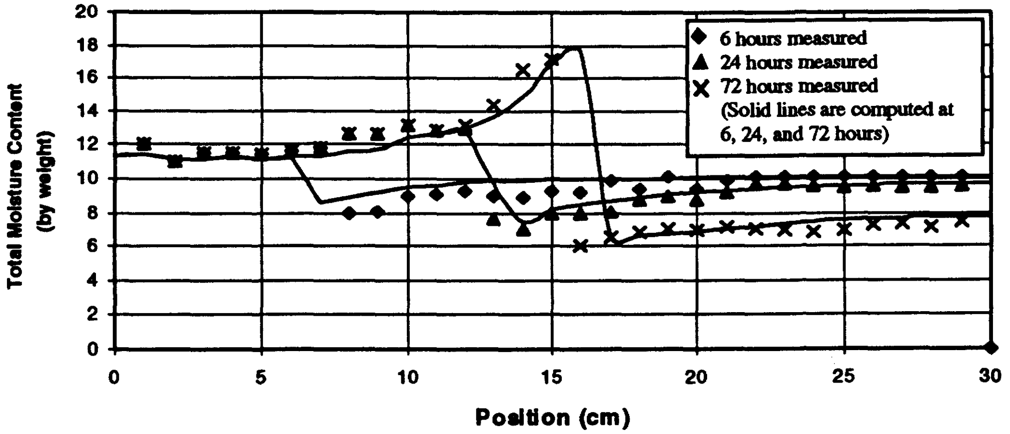


Figure 6.6c SWC 3

Figure 6.6 Moisture Content Profiles Computed Using Three Different Soil Water Characteristic Curves

The results for the simulations using the soil water characteristic curve with the intermediate radius of curvature (i.e., SWC 1 from Figure 6.5) show a computed frost front which precedes the measured frost front by about 2 cm. In addition, the computed ice contents are slightly high in the initial stages of the simulation, while the computed unfrozen water contents are slightly low in the later stages of the simulation.

The results for the simulations made using the soil water characteristic curve with the largest radius of curvature (i.e., SWC 2) show ice contents even higher than those computed using the soil water characteristic curve marked SWC 1 and unfrozen water contents much lower than those computed with SWC 1. In addition, the computed frost front precedes the measured frost front by about 3 cm.

The results of the simulations made using the soil water characteristic curve marked as SWC 3 in Figure 6.5 show a computed frost front position which agrees well with the measured frost front position. In addition, ice and unfrozen water contents are in agreement with measured results. In summary, the results presented in Figure 6.6 shows that for this material and test conditions, small errors in the estimation or measurement of the radius of curvature of the soil water characteristic curve near the residual water content had a significant effect on the accuracy of the computed results.

Figure 6.7 compares the simulation results for the same test obtained with slightly different saturated coefficients of permeability. The measured saturated coefficient of permeability for the silica flour used in this study varied about one order of magnitude over a porosity range of 0.48 to 0.51. Because it was not possible to re-construct the material used by Jame (1980) (i.e., soil type, density), some flexibility was used in the selection of the saturated coefficient of permeability used in this study.

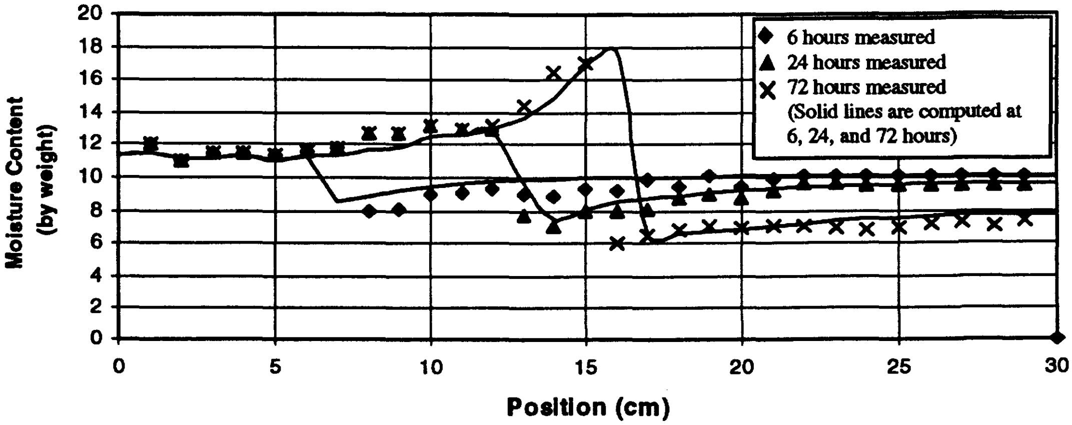


Figure 6.7a Ks = 4.5E-5

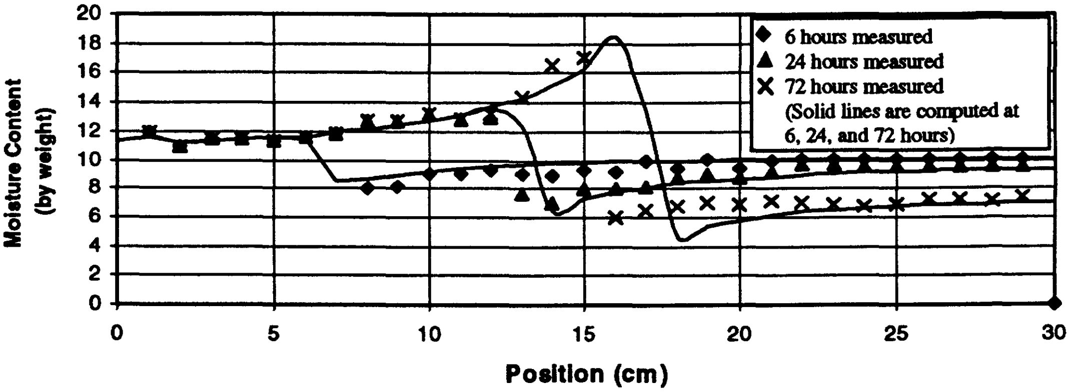


Figure 6.7b Ks = 7.0E-5

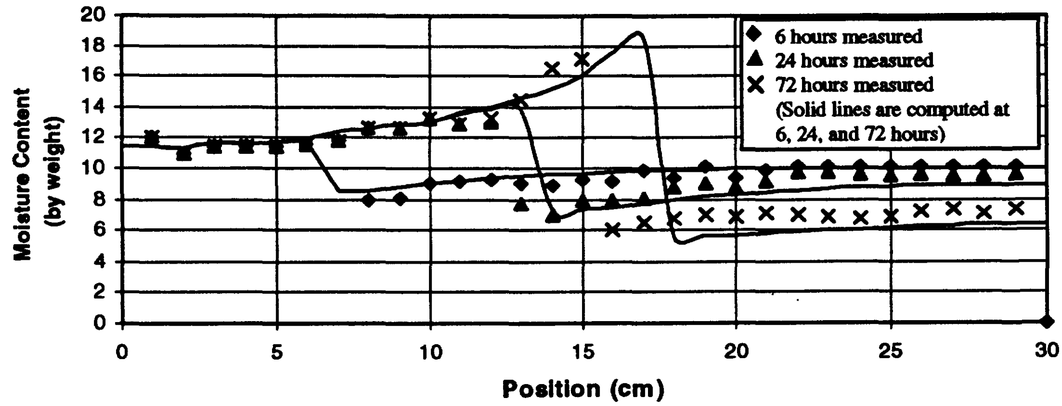


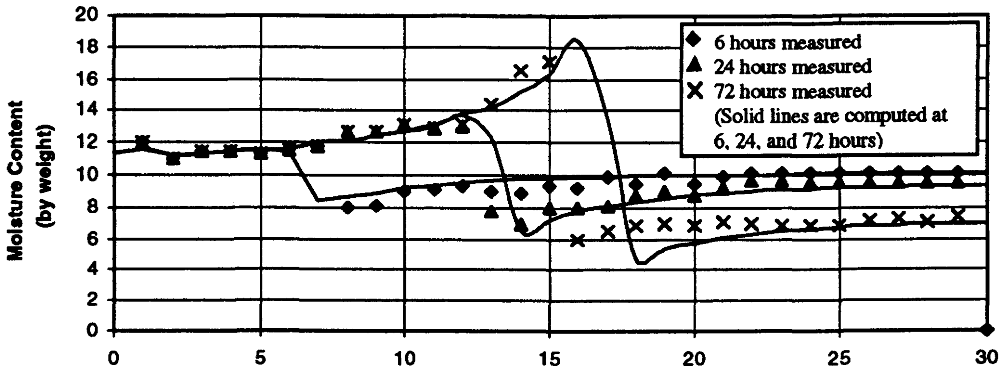
Figure 6.7c Ks = 9.5E-5

Figure 6.7 Moisture Content Profiles Computed Using Three Different Saturated Coefficients of Permeability

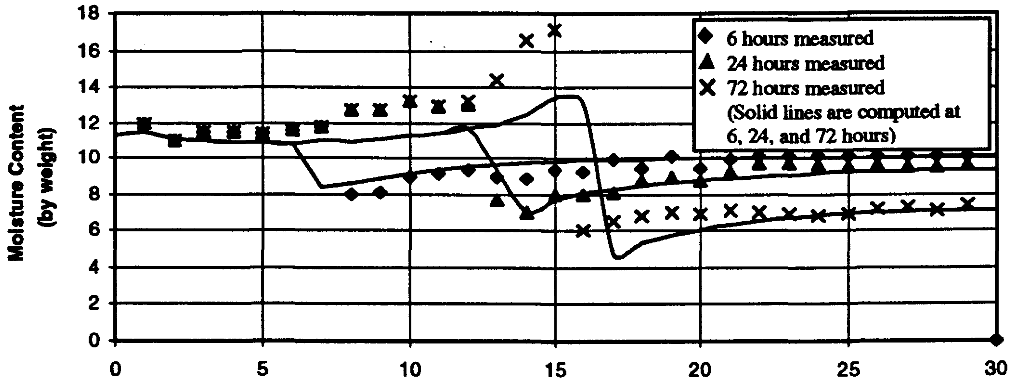
Figure 6.7a shows the simulated temperature and moisture contents using a Ksat of 4.5 x 10 $^{-5}$ cm/s. In this figure, there is excellent agreement between computed and measured temperatures and water contents. In Figure 6.7b the Ksat is 7.0 x 10 $^{-5}$ cm/s. It can be noted that the computed frost front slightly precedes the measured frost front, and the computed unfrozen water contents are slightly lower than the measured unfrozen water contents. Finally, with a Ksat of 9.5 x 10 $^{-5}$ cm/s (i.e., Figure 6.7c) the computed frost front also precedes the measured frost front, and the computed unfrozen water contents are even lower than those computed with the slightly lower Ksat.

Based on the results presented in Figure 6.7 it can be concluded that for this material and test conditions, a higher permeability resulted in a faster moving frost front, which in turn resulted in lower than actual predictions of water content in the unfrozen zones at the end of the test. It can be noted that for all three Ksat values, the agreement between measured and computed unfrozen water contents is better in the earlier stages of the test. This suggests that the Ksat has a greater effect on the rate of frost front advance than on the unfrozen water content. The reason the higher Ksat test shows a lower unfrozen water content at later stages of testing is that more nodes have frozen (due to the faster rate of frost front advance) and higher suctions have advanced deeper into the soil, thus drawing more moisture out of the unfrozen zone.

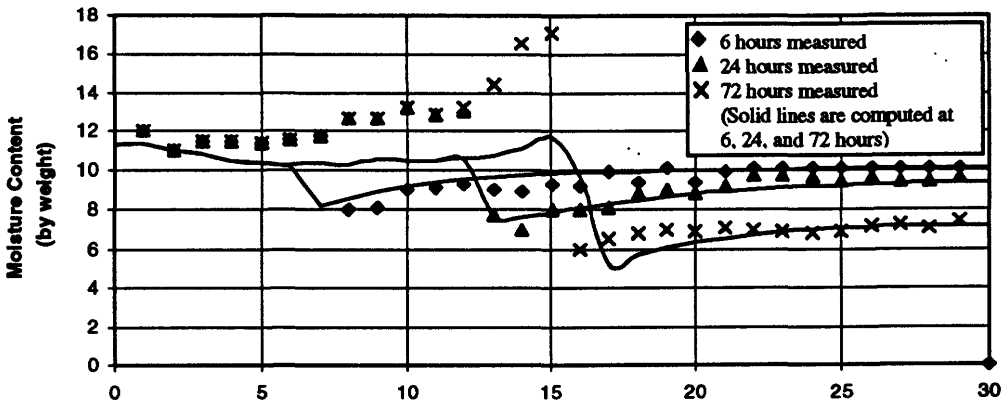
Figure 6.8 compares the simulated results of the same freezing test using three different ice impedance factor coefficients (i.e., E). In the top chart of the figure, no impedance factor was applied during any stage of the simulation. Contrary to results obtained by Jame (1977), Taylor and Luthin (1978) and others, these results show that reasonable predictions of ice content can be obtained without an arbitrarily chosen ice impedance factor.



Position (cm)
Figure 6.8a E = 0



Position (cm)
Figure 6.8b E = 6



Position (cm)
Figure 6.8c E = 12

Figure 6.8 Moisture Content Profiles Computed Using Three Different Ice Impedance Factors

The middle chart of Figure 6.8 shows the computed results obtained with an ice impedance factor coefficient (i.e., E) of 6. This implies a 0 to 1.5 order of magnitude drop in permeability as the volumetric ice content increases from 0 to 0.25. These results clearly show a large under estimation of ice content behind the frost front. The bottom chart shows the computed results obtained using an ice impedance factor coefficient of 12, or a 0 to 3 order magnitude drop in permeability as the volumetric ice content increases from 0 to 0.25. This figure shows there is an actual decrease in ice content during the middle stages of testing and then a very slight ice build up as the test approaches the 72 hour mark.

Based on the results presented in Figure 6.8, it can be concluded that application of an ice impedance factor reduces the computed build up of ice behind the frost front. These findings also raise serious questions about the necessity of the ice impedance factor. Harlan (1973) originally hypothesized that the water permeability of a frozen soil could be obtained from the suction - permeability relationship measured at room temperature for any given unfrozen water content. Harlan (1973), Jame (1977) and others were not able to verify this theory. In any case the impedance factor was introduced to make computed results fit measured data. Perhaps the earlier researchers were not able to accurately measure the soil water characteristic curve, or they were not able to extrapolate a reasonable suction - permeability function from the soil water characteristic curve. In this study, the relative permeability function was obtained using the Fredlund et al. (1994) method and it appears to have given reasonable results even in the frozen zone of the soil.

The hypothesis proposed by Harlan (1973) appears logical. If thermodynamic equilibrium requires that the larger pores freeze first (i.e., where the suction is lowest there is less freezing point depression) then any unfrozen water must remain in the smaller pores. In a draining, unfrozen soil, the larger pores also drain first because there is less surface tension across a larger radius of curvature pore. In this case, the remaining water is also stored in the smaller pores. The question can be raised, are these not the same smaller pores which remain unfrozen when the soil freezes? If the suction - permeability relationship can predict the permeability of capillary water in the unsaturated pores spaces in a drying soil, then the same relationship should apply to freezing soils. In any case, more work needs to be done to explore these possibilities.

Four tornado plots are presented in Figures 6.9 to 6.12. These plots illustrate the sensitivity of the computed results to the changes in soil property functions discussed above. The first plot compares the position of the computed and simulated temperature profile where it crosses the 0°C axis at 6, 12, and 72 hours. The second plot compares the computed and measured ice contents at the freezing front at 6, 12 and 72 hours. The third plot compares the unfrozen water contents at the freezing front; and the last plot compares the unfrozen water contents well ahead of the freezing front.

These figures clearly show that slight changes to the soil water characteristic curve or saturated coefficient of permeability cause significant differences between computed and experimental data. The ice impedance factor has a significant effect in the computation of ice contents behind the freezing front and also the computation of unfrozen water content at the freezing front. In general, these figures help illustrate the complex relationships which exist between suctions, temperatures, and liquid flux near the frost front in a freezing soil.

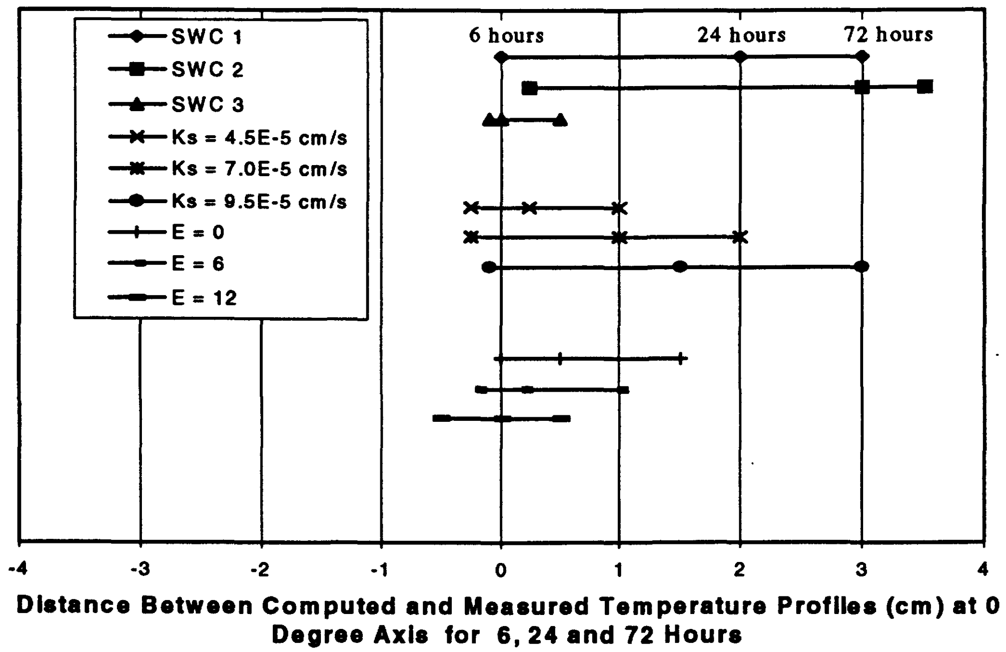


Figure 6.9 Tornado Plot Comparing Position of Temperature Profiles Computed Using Slightly Different Soil Property Functions

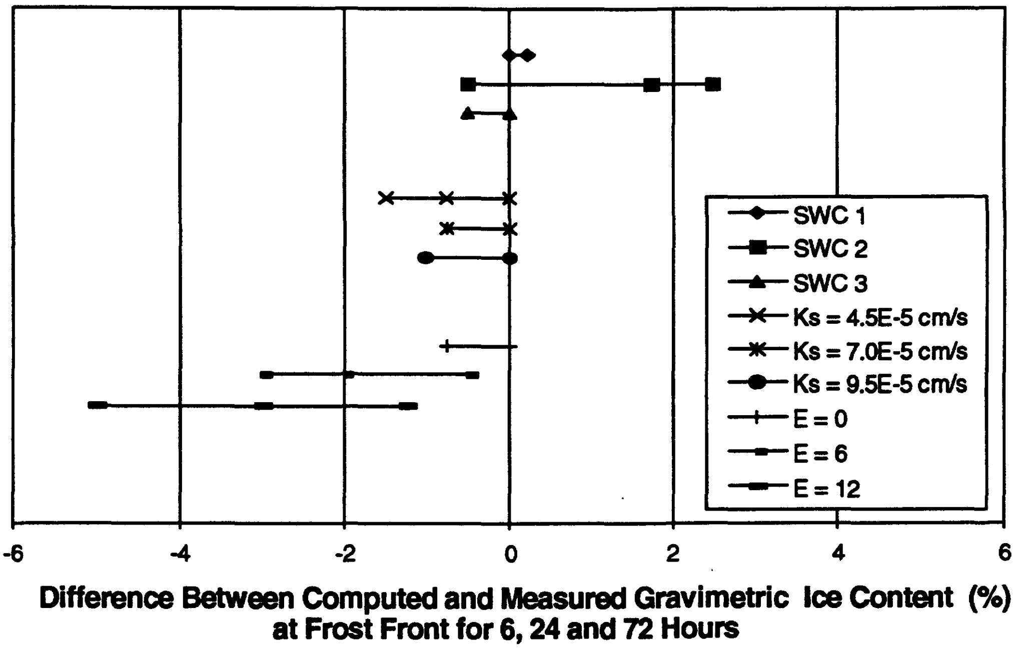


Figure 6.10 Tornado Plot Comparing Ice Contents Computed Using Slightly Different Soil Property Functions

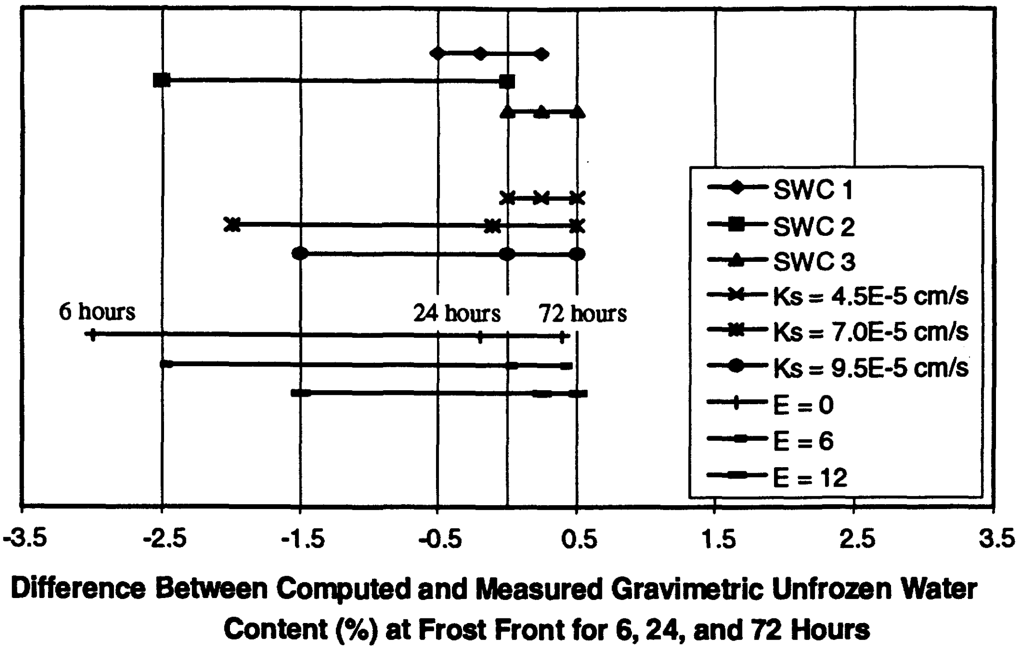


Figure 6.11 Tornado Plot Comparing Unfrozen Water Contents Computed Using Slightly Different Soil Property Functions

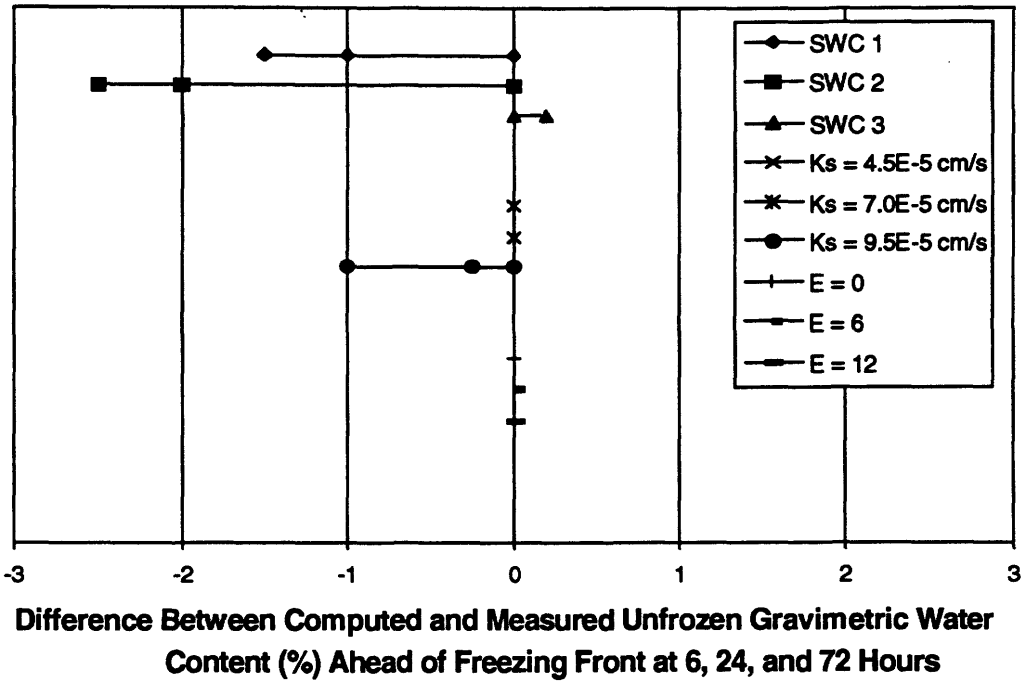


Figure 6.12 Tornado Plot Comparing Unfrozen Water Contents Computed Using Slightly Different Soil Property Functions

CHAPTER 7
CONCLUSIONS

Numerical analysis of freezing and thawing processes in unsaturated soils is complicated by many factors. The desire to obtain a clear understanding of the physics involved in soil freezing has led researchers to develop numerous theoretical models for analysing the problem. The literature shows that many different models have been developed, and that none of them are capable of being a general tool for geotechnical engineers. In fact, most of them were developed as research tools designed to fulfill a unique objective. All of the models, including the modified SoilCover (MEND, 1993) program, are restricted by the assumptions that were made as the models were developed.

Thermodynamic equilibrium theory has been applied to soil freezing with limited success. If the water in a soil is assumed to be held totally by either capillary or adsorptive forces, then it is possible to relate matric suction to sub-zero temperature using the appropriate form of the Clapeyron equation. However, the majority of soils contain both capillary and adsorptive water; thus, the Clapeyron equation falls short. Thermodynamic equilibrium theory, among others, has been used to develop freezing point depression relationships. Jame (1972) showed that the freezing point depression function was the same as the unfrozen water content versus sub-zero temperature function. This is advantageous in numerical modelling because the same curve can be used to determine when the pore-water will begin to freeze (based on unfrozen water contents in the pores) and also what the unfrozen water content will be after the majority of pore-water has frozen.

To date, the soil water characteristic curve has been used sparingly in soil freezing analysis. This curve plays a significant role in the modified SoilCover (MEND, 1993) model because it couples the thermal and stress states of the soil in the frozen or partially frozen zones. The unfrozen water content is common to both the soil freezing curve and the soil water characteristic curve which enables the matric suction to be computed for any sub-zero temperature in the soil. When the slope of the soil freezing curve is divided by the slope of the soil water characteristic curve the resulting value is a ratio of proportionality between changing suctions and changing temperatures. It provides the constant values used in the Clapeyron equation, and it is not limited by soil type. This relationship enabled a single equation to be derived which accounted for both the heat flow and mass transfer continuity.

The soil water characteristic curve can be used to compute the relationship between water permeability and suction in unfrozen soils. However, researchers are divided on how to determine the coefficient of permeability in a frozen or partially frozen soil. This division gave rise to the term 'ice impedance factor', an empirical relationship used to calibrate soil freezing models. It was initially intended that an ice impedance factor be used in this program. However, results of initial testing showed that the permeability versus suction function that was derived from the soil water characteristic curve using equations recently presented by Fredlund et al. (1994) gave excellent results for liquid flux in the frozen and unfrozen zones. This is a very important finding and needs to be investigated in more detail.

The objective of this research program was to present theory for heat and mass transfer in freezing unsaturated soils. The theory was then verified using laboratory data. The laboratory modelling program was carried out by simulating soil freezing of fine silica flour with large water fluxes in the unfrozen zone. The results of these simulations verified that the freezing analysis capabilities of the revised numerical model were working. A sensitivity analysis was also done to compare computed results using slightly different soil property functions. The results of the sensitivity study clearly showed that small changes to the soil water characteristic curve or saturated coefficient of permeability value caused significant differences between computed and experimental results. Inclusion of an ice impedance factor had a significant effect in the computation of ice contents behind the freezing front, and also in the computation of unfrozen water content at the freezing front. The results of this study suggest that an ice impedance factor is not necessary if an accurate permeability versus suction relationship can be predicted.

The field modelling exercise (presented in Appendix A) was carried out to demonstrate that the revised model was capable of both freezing and thawing analysis, and that it could do so within the framework of the existing computer code. In the field data simulations, good agreement between computed and measured temperature profiles was obtained for times when ice was not present in the soil. An incorrect calculation of pore ice affected the calculated thermal conductivity values which in turn resulted in a calculated frost front that advanced too deep into the soil during one period of the simulation. The accuracy of the computed results could be improved by using a more accurate soil freezing curve relationship, as it directly affects the computed ice content values. In addition, more information is needed about the bottom and top boundary conditions (i.e., in the waste rock and beneath the snow pack).

This revised numerical model should be considered as a first step in developing a truly year round soil heat and mass transfer model. The current formulation uses suction and temperature as dependent variables and relies heavily on the soil freezing and soil water characteristic curves. More information about the complex relationship between these curves is needed. In particular, it would be desirable to be able to accurately predict the soil freezing curve from soil water characteristic curve data. More research should also be carried out to explore the relationships between soil water characteristic curves and suction - permeability functions in freezing soils.

Future modifications to the model should include: adding convective heat transfer and sublimation effects; adding an algorithm to couple the soil surface with the snow and the snow surface with the atmosphere; and incorporating unsaturated soil mechanics theory to account for total stress, effective stress, ice pressures and frost heave.

The revised numerical model is useful to engineers in its current form. If a user has a clear understanding of the limitations and advantages inherent in the model, then he or she should be able to carry out field response and predictive modelling on a year round basis.

REFERENCES

Anderson, Duwayne M. and N.R. Morgenstern, 1973. Physics, Chemistry, and Mechanics of Frozen Ground: A Review. Proceedings: Second International Conference on Permafrost. Yakutsk, USSR., pp. 257-288.

Beskow, G., 1935. Soil Freezing and Frost Heaving with Special Application to Roads and Railroads. The Swedish Geological Society, C, no. 375. Year Book no. 3 (Translated by J.O. Osterberg), Technological Institute, Northwestern University.

Black, Patrick B. and Allen R. Tice, 1989. Comparison of Soil Freezing Curve and Soil Water Curve Data for Windsor Sandy Loam. Water Resources Res., vol. 25(10), pp. 2205-2210.

Black, Partick B. and Mark J. Hardenberg, Editors, 1991. Historical Perspective in Frost Heave Research, The Early Works of S. Taber and G. Beskow. CRREL Special Report 91-23. U.S. Army Corps of Engineers, Hanover, New Hampshire.

Boyoucous, G.J., 1920. Degree of Temperature to Which Soils Can Be Cooled Without Freezing. Journal of Agricultural Research., vol. 20, pp. 267-269.

Dakshanamurthy, V. and D.G. Fredlund, 1981. A Mathematical Model for Predicting Moisture Flow in an Unsaturated Soil under Hydraulic and Temperature Gradients. Water Resources Research Journal., vol. 17(3), pp. 714-722.

De Vries', D.A., 1952. The Thermal Conductivity of Soils. Mededelingen Van Der Landbouwhogeschool te Wageningen, 52, pp. 1-73.

Dirksen, C. and R.D. Miller, 1966. Closed-System Freezing of Unsaturated Soil. Soil Science Society of America Proceedings., vol. 30, pp. 168-173.

Everett, D.H., 1961. The Thermodynamics of Frost Damage to Porous Solids. Faraday Soc. Trans. 57 (465, Part 9), pp. 1542-1551.

Farouki, Omar, 1981. Thermal Properties of Soils. CRREL Monograph 81-1. U.S. Army Corps of Engineers, Hanover, New Hampshire, USA.

Fisher, R.A., 1924. The Freezing of Water in Capillary Systems: A Critical Discussion. Journal of Physical Chemistry., vol. 28, pp. 36-67.

Flerchinger, Gerald Norma, 1987. Simultaneous Heat and Water Model of a Snow-Residue-Soil System. Ph.D. Thesis, Washington State University, Pullman, Washington.

Fredlund, D.G. and H. Rahardjo, 1993. Soil Mechanics for Unsaturated Soils. John Wiley & Sons, Inc. New York.

Fredlund, D.G., Xing, Anqing, and Shangyan Huang, 1994. Predicting the Permeability Function for Unsaturated Soils Using the Soil-Water Characteristic Curve. Canadian Geotechnical Journal., vol. 31, pp. 533-546.

Fredlund, D.G., and Anqing Xing, 1994. Equations for the Soil-Water Characteristic Curve. Canadian Geotechnical Journal, vol. 31, pp. 521-532.

Gilpin, R.R., 1980. A Model for the Prediction of Ice Lensing and Frost Heave in Soils. Water Resources Research, Vol. 16 (5). pp. 918-930.

Gosnik, J.P., Kawasaki, K., Osterkamp, T.E., and J. Holty, 1988. Heat and Moisture Transport During Annual Freezing and Thawing. Proceedings: Fourth International Conference on Permafrost. July 17-22. Fairbanks, Alaska., pp. 355-360.

Gray, D.M., R.J. Granger, and G.E. Dyck, 1985. Overwinter Soil Moisture Changes. Transactions, ASCE., vol. 28(2), pp. 442-447.

Guymon, Gary L, and James N. Luthin, 1974. A Coupled Heat and Moisture Transport Model for Arctic Soils. Water Resources Res., vol. 10(5), pp. 995-1001.

Harlan, R.L., 1973. Analysis of Coupled Heat-Fluid Transport in Partially Frozen Soil. Water Resource Res., vol. 9, pp. 1314 - 1323.

Hoekstra, P., 1966. Moisture Movement in Soil Under Temperature Gradients with the Cold Side Temperature Below Freezing. Water Resource Res., vol. 2, pp. 241-250.

Hromadka, Ted, 1987. A Nodal Domain Integration Model of Two-Dimensional Heat and Soil-Water Flow Coupled by Soil-Water Phase Change. CRREL Special Report 87-9. U.S. Army Corps of Engineers, Hanover, New Hampshire.

Jame, Y.W., 1972. Temperature Effects on Phase Composition of a Partially Frozen Soil. M.Sc. Thesis. University of Saskatchewan, Saskatoon, Canada.

Jame, Y.W., 1977. Heat and Mass Transfer in Freezing Soil. Ph.D. Thesis. University of Saskatchewan, Saskatoon, Canada.

Jame, Y.W. and D.I. Norum, 1980. Heat and Mass Transfer in Freezing Unsaturated Porous Media. Water Resource Res., vol. 16, pp. 811-819.

Johansen, O., 1975. Thermal Conductivity of Soils, Ph.D. Theses. Trondheim, Norway. (CRREL Draft Translation 637, 1977).

Joshi, Bhaskar, 1993. A Finite Element Model for the Coupled Flow of Moisture and Heat is Soils under Atmospheric Forcing. M.Sc. Dissertation, University of Saskatchewan, Saskatoon, Canada.

Jumikis, A.R., 1966. Thermal Soil Mechanics. Rutgers University Press., New Brunswick, N.J.

Kemper, W.D., 1960. Water and Ion Movement in Thin Films as Influenced by the Electrostatic Charge and Diffuse Layer of Cations Associated with Clay Mineral Surfaces. Soil Science Society of America Proceedings., vol. 24, pp. 10-16.

Kersten, M.S., 1949. Laboratory Research for the Determination of the Thermal Properties of Soils. ACFEL Technical Report 23. AD 712516.

Konrad, Jean-Marie, and Norbert R. Morgenstern, 1980. A Mechanistic Theory of Ice Lens Formation in Fine-Grained Soils. Canadian Geotechnical Journal., vol. 17, pp. 473-486.

Lange, G.R. and H.L. McKim, 1966. Saturation, Phase Composition, and Freezing-Point Depression in a Rigid Soil Model. Permafrost: Proceedings of an International Conference. National Academy of Sciences, Washington, D.C., pp. 187-192.

Low, P.F., D.M. Anderson and P. Hoekstra, 1968. Some Thermodynamic Relationships for Soil At or Below the Freezing Point: 1. Freezing Point Depression and Heat Capacity. Water Resources Res., vol. 4, pp. 379-394.

Lunardini, V.J., 1991. Heat Transfer with Freezing and Thawing. Elsevier Science Publishers B.V., New York, N.Y.

Mageau, D.W. and N.R. Morgenstern, 1980. Observations on Moisture Migration in Frozen Soils. Canadian Geotechnical Journal., vol. 17, pp. 54-60.

Martynov, G.A., 1959. Heat and Moisture Transfer in Freezing and Thawing Soils. In Principles of Geocryology. Nathional Research Council of Canada, Technical Translation 1065, Chapter VI.

Maulem, Y. 1976. A New Model for Predicting the Hydrualic Conductivity of Unsaturated Porous Media. Water Resource Research., vol. 12, pp. 513-522.

MEND, 1993. SoilCover, User's Manual for an Evaporative Flux Model. University of Saskatchewan, Saskatoon, Saskatchewan, Canada.

McGaw, R.W., R.L. Berg, and J.W. Ingersoll, 1982. An Investigation of Transient Processes in an Advancing Zone of Freezing. Proceedings: Fourth International Conference on Permafrost. July 17-22. Fairbanks, Alaska., pp. 821-825.

Miller, Robert D., 1966. Phase Equilibria and Soil Freezing. Proceedings of an International Conference. National Academy of Sciences, Washington, D.C., pp. 193-197.

Miller, Robert D., 1973. Soil Freezing in Relation to Pore Water Pressure and Temperature. Proceedings: Second International Conference on Permafrost. Yakutsk, USSR., pp. 244 - 352.

Nakano, Y., A.R. Tice, J.L. Oliphant, and T.F. Jenkins, 1982. Soil-Water Diffusivity of Unsaturated Soils at Sub-Zero Temperatures. Proceedings: Fourth International Conference on Permafrost. July 17-22. Fairbanks, Alaska., pp. 889-893.

Nersesova, Z.A., and N.A. Tsytovich, 1966. Unfrozen Water in Frozen Soils. Permafrost: Proceedings of an International Conference. National Academy of Sciences, Washington, D.C., pp. 230-233

Nixon, J.F., 1975. The Role of Convective Heat Transport in the Thawing of Frozen Soils. Canadian Geotechnical Journal., vol. 12, pp. 425-429.

Nixon, J.F., 1991. Discrete Ice Lens Theory for Frost Heave in Soils. Canadian Geotechnical Journal., vol. 28, pp 843-859.

Nixon, J.F., 1992. Discrete Ice Lens Theory for Frost Heave Beneath Pipelines. Canadian Geotechnical Journal., vol. 29, pp. 487-497.

O'kane, M, 1995. Instrumentation and Monitoring of Engineered Soil Covers for Acid Generating Mine Waste. M.Sc. Thesis, Department of Civil Engineering, University of Saskatchewan, Saskatoon, Saskatchewan, Canada.

Oliphant, J.L., A.R. Tice and Y. Nakano, 1982. Water Migration Due to a Temperature Gradient in Frozen Soil. Proceedings: Fourth International Conference on Permafrost. July 17-22. Fairbanks, Alaska., pp. 951-956.

Penner, E., 1959. The Mechanism of Frost Heaving in Soils. Highway Research Board. Bulletin 225. Washington. pp. 1-22.

Penner, E., 1967. Heaving Pressure in Soils During Unidirectional Freezing. Canadian Geotechnical Journal., vol. 4, pp. 398-408.

Philip, J.R. and D.A. de Vries, 1957. Moisture Movement in Porours Materials under Temperature Gradients. Transaction, American Geophysical Union., vol 38, no. 2, pp. 222-232.

Schofield, R.K., 1935. The PF of the Water in Soil. Transactions, 3rd. International Congress of Soil Science., vol. 2, pp. 37-48.

Smith, W.O., 1939. Thermal Conductivitites in Moist Soils. Soil Science Society of America Proceedings., vol. 4, pp. 521-524.

Smith, M.W., 1985. "Models of Soil Freezing", in Field and Theory, Lectures in Geocryology., M. Church and O. Slaymaker editors. University of British Columbia Press.

Swanson, D.A., 1995. Predictive Modelling of Moisture Movement in Engineered Soil Covers for Acid Generating Mine Waste. M.Sc. Thesis, Department of Civil Engineering, University of Saskatchewan, Saskatoon, Saskatchewan, Canada.

Taber, S., 1929. The Mecanichs of Frost Heaving. Journal of Geology., vol. 38, pp. 308-317.

Takagi, Shunsuke, 1966. Theory of Freezing-Point Depression with Special Reference to Soil Water. Proceedings of an International Conference. National Academy of Sciences, Washington, D.C., pp. 216-224.

Tao, Y.-X. and D.M. Gray, 1994. Prediction of Snowmelt Infiltration into Frozen Soils. Numerical Heat Transfer, Part A., vol. 26, pp. 643-665.

Taylor, George S. and James N. Luthin, 1978. A Model for Coupled Heat and Moisture Transfer During Soil Freezing. Canadian Geotechnical Journal., vol. 15, pp. 548-555.

Tice, A.R., D.M. Anderson, and A.Banin, 1976. The Prediction of Unfrozen Water Contents in Frozen Soils from Liquid Limit Determinations. CRREL Report 76-8, Hanover, New Jersey.

Williams, P.J., 1964. Unfrozen Water Content of Frozen Soils and Moisture Suction. Geotechnique., vol. 14, pp. 133-142.

Williams, P.J., 1966. Suction and Its Effects in Unfrozen Water of Frozen Soils. Proceedings of an International Conference. National Academy of Sciences, Washington, D.C., pp. 225 - 229.

Williams, P.J., 1967. Properties and Behaviour of Freezing Soils. Norwegian Geotechnical Institute., Oslo. Publication 72.

Wilson, G. Ward, 1990. Soil Evaporative Fluxes for Geotechnical Engineering Problems. Ph.D. Thesis. University of Saskatchewan, Saskatoon, Saskatchewan, Canada.

Yong, Raymond, 1965. Soil Suction Effects on Partial Soil Freezing. Highway Research Rec., vol. 68, pp. 31-42.

APPENDIX A
FIELD DATA MODELLING APPLICATION

A1 Introduction to Field Data Modelling

To illustrate a practical use for the revised model, a simulation was carried out to compare computed and measured over winter soil temperature data that was collected from an instrumented soil cover over mine waste at Equity Mine near Houston, B.C. The field data modelling program also confirmed that all of the revisions to the SoilCover (MEND, 1993) program were compatible with the existing program. In particular, it was necessary to ensure that a smooth transition between non-freezing and freezing conditions took place as the surface boundary conditions changed and the upper layers of soil began to freeze. Also, it was necessary to ensure that the transition between a freezing soil and a thawing soil did not disrupt the solution process. Recall that the laboratory data modelling program did not deal with thawing soils.

The soil cover system at Equity Mine is comprised of 50 cm of compacted glacial till overlain by 30 cm of loose glacial till. For modelling purposes, 1 m of waste rock was assumed to exist below the compacted till. The loose till cover at the field site was vegetated to decrease erosion and to reduce precipitation and snow melt runoff. Reduced runoff may lead to higher infiltration which in turn keeps the cover system near saturation. However, vegetation increases evapotranspiration which tends to reduce saturation. It is desirable to keep the cover saturated because this reduces the infiltration of oxygen which reacts with water in the waste rock. This results in the oxidation of sulfide bearing minerals in the waste rock and leads to acid mine drainage problems.

A detailed description of the instrumentation installed at the Equity site is given by O'kane (1995). The instrumentation consists of a fully automated weather station which measures air temperature, relative humidity, wind speed, precipitation (excluding snow fall), and global and net radiation. Vertical culverts were installed at three locations on the cover to give access to thermal conductivity sensors which were inserted horizontally into the cover and waste rock at different depths. The thermal conductivity sensors were connected to automatic data loggers to record suctions and temperatures at different depths. Neutron probe access tubes, lysimeters, and oxygen probes were also installed at various locations on the cover. The data used in this study was obtained from a culvert stationed on the south west face of the main dump.

A2 Soil Properties Used in Field Data Modelling Program

The types of soil properties required as input for SoilCover are the same as those discussed in section 4.3. Details of the laboratory testing and field calibration of these soil properties are given by Swanson (1995). All the soil property relationships presented below are the same as those used by Swanson (1995) for the non-freezing modelling of the soil cover at Equity Silver Mine with the exception of the soil freezing curves for the three materials used in this study. Modifications to the soil properties for frozen soils are included in the program computer code.

Figure A1 shows soil water characteristic curves used by Swanson (1995) in modelling the Equity cover system. The waste rock soil water characteristic curve used in the study is based on the curve for Beaver Creek sand. It fits the trend expected for waste rock and it indicates that a capillary break should exist between the compacted clay and the coarse waste rock.

Figure A2 shows the relative coefficients of permeability for the three soils used in this study. The saturated coefficient of permeability of the loose till, compacted till, and waste rock were 5.7x 10^{-7} cm/s, 2.0x10^{-8} cm/s, and 1.3.0x10^{-3} cm/s respectively. The specific gravity of the till was calculated to be 2.77, with a field dry density of 1.74 Mg/m^3 and a calculated porosity of 0.37. The average dry density of the compacted till was estimated to be 1.85 Mg/m^3 with a porosity of 0.33 (Swanson, 1995). The porosity of the Beaver Creek sand was calculated to be 0.4 with an air entry value of 3.8 kPa (Wilson, 1990).

The soil freezing curves for the three materials are shown in Figure A3. These curves were computed using the soil water characteristic data curves shown in Figure A1 and the form of the Clapeyron equation presented by Black and Tice (1989) (i.e., discussed in Chapter 2). In applying the Clapeyron equation, it is necessary to know whether the pore-water is held by capillary or adsorptive forces. In this case, the water in the waste rock was assumed to be held by capillary forces and water in the tills was assumed to be held totally by adsorptive forces.

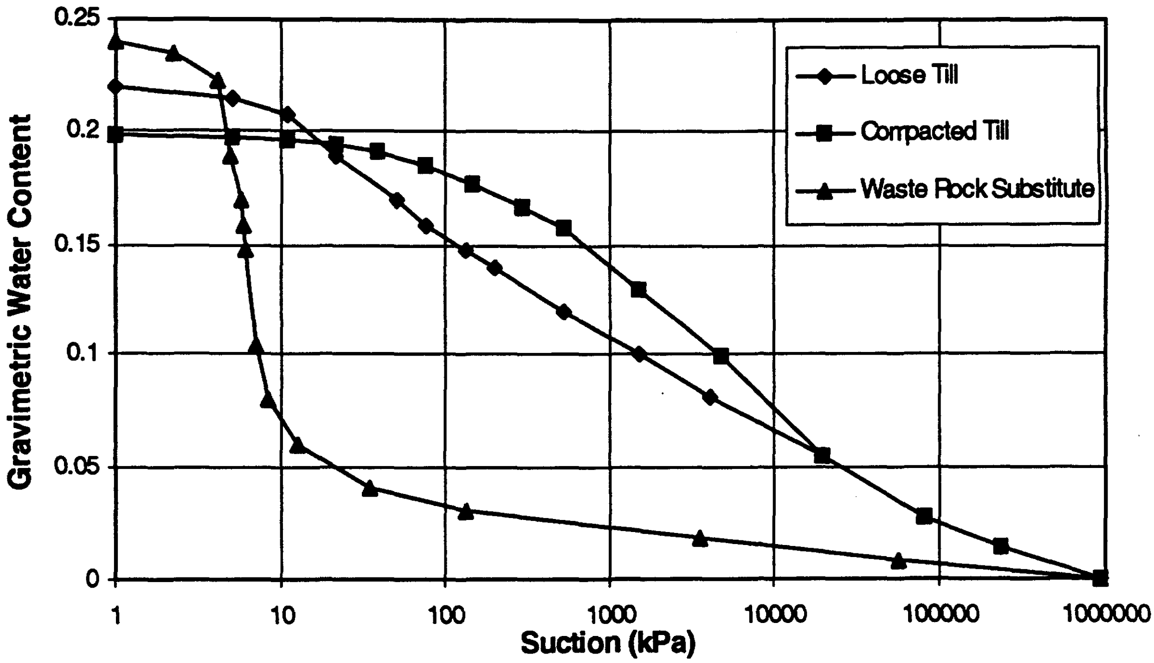


Figure A1 Soil Water Characteristic Curves for Modelling Field Data (after Swanson, 1995)

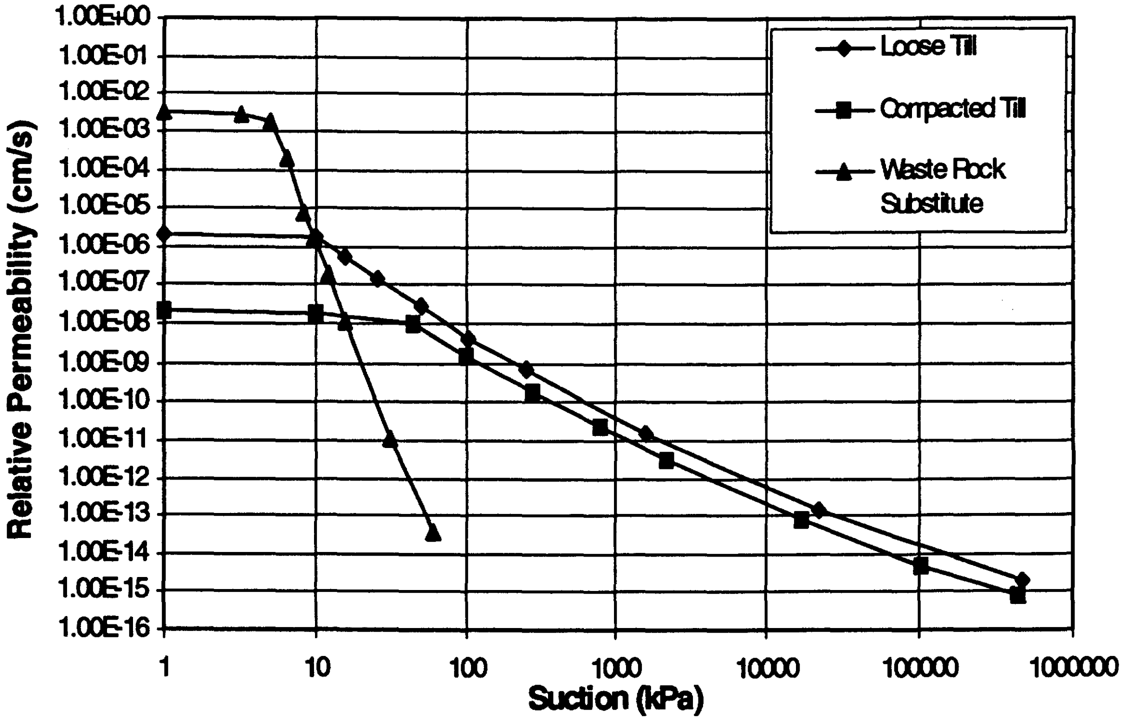


Figure A2 Relative Coefficients of Permeability for Modelling Field Data (modified after Swanson, 1995)

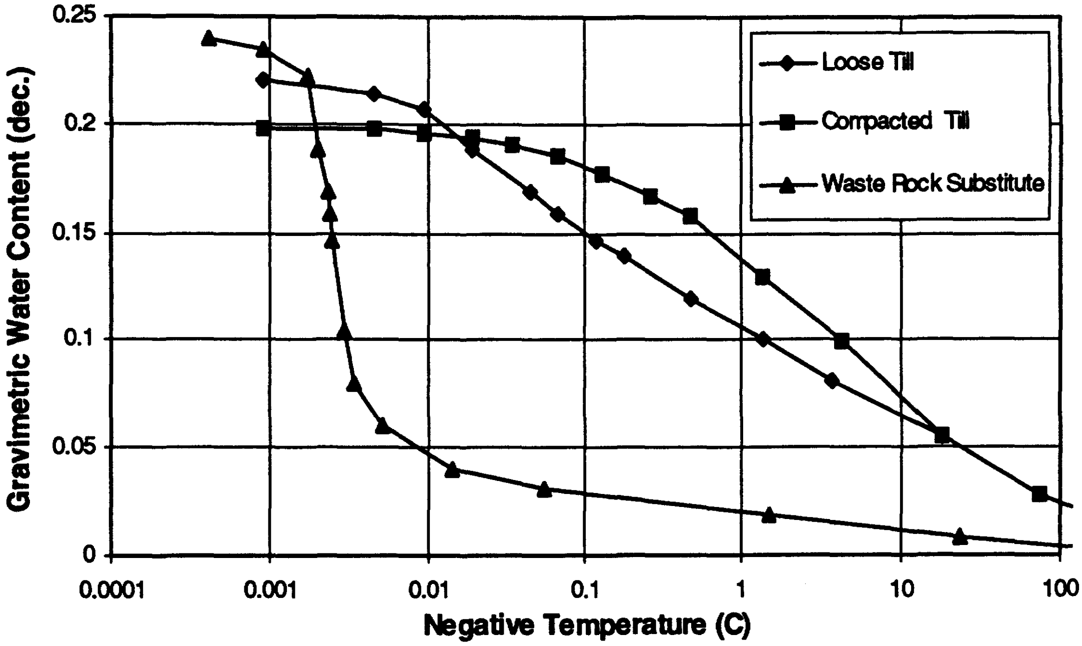


Figure A3 Soil Freezing Curve Data Used in Field Modelling Program

Figure A4 shows the linearized 'G' function which relates changes in temperature with changes in suction. Recall that this term is required to render the modified heat transfer equation determinate. It should also be noted in Figure A4 that the 'G' function is constant for all water contents. This is a result of deriving the soil freezing curve from the soil water characteristic curve (i.e., the values of 'G' in Figure A4 are the constants of proportionality inherent in the Clapeyron equations given by Black and Tice (1989).

Figure A5 shows the thermal conductivity function of each soil for the non-freezing case. The appropriate form of the Johansen (1975) method for computing unfrozen thermal conductivity was matched to the curves in Figure A5 so that the necessary constants could be obtained for use in the method proposed by Johansen (1975) for frozen soils (i.e., Chapter 2). The dry thermal conductivity, λ_{dry} , was calculated to be 0.4 W/m°C for the both the till and waste rock. The effective solids thermal conductivity, λ_s , was calculated to be 5.05 W/m°C and 4.05 W/m°C for the till and waste rock respectively.

Figure A6 shows the computed volumetric specific heat of the till and waste rock materials for non-freezing conditions. The volumetric specific heat of a freezing soil is a function of both water and ice contents and is not a single valued function. For the freezing case, the volumetric specific heat is computed by adding the appropriate specific heat for the volume of ice present as described in Chapter 2. This is done within the SoilCover program.

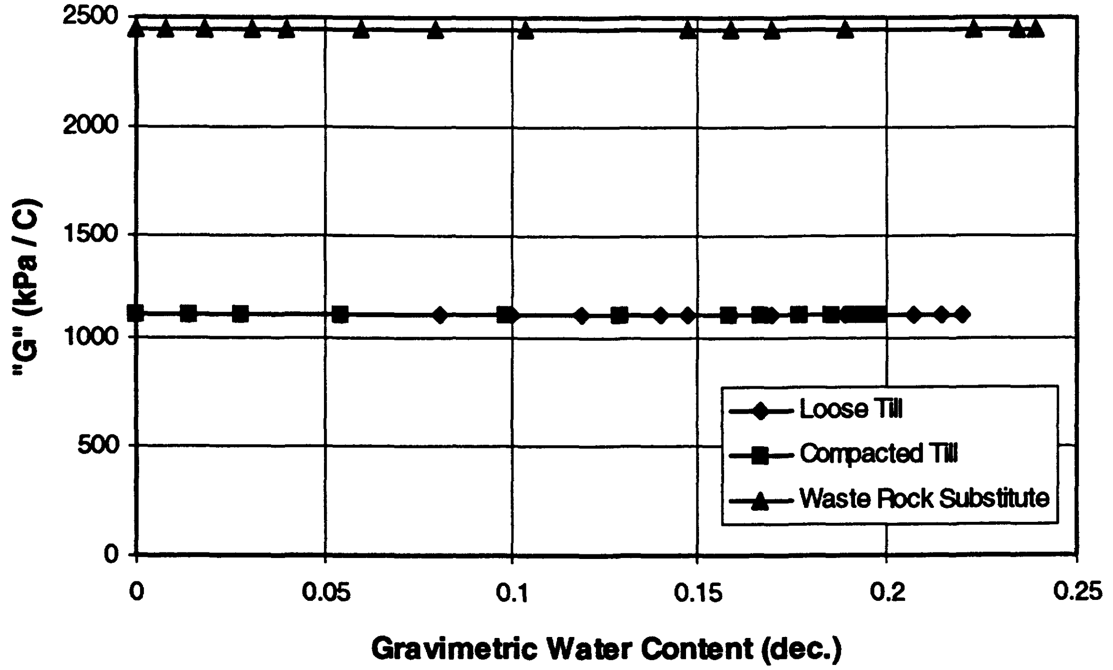


Figure A4 Ratio of Proportionality, 'G' Between Change in Suction and Change in Temperature

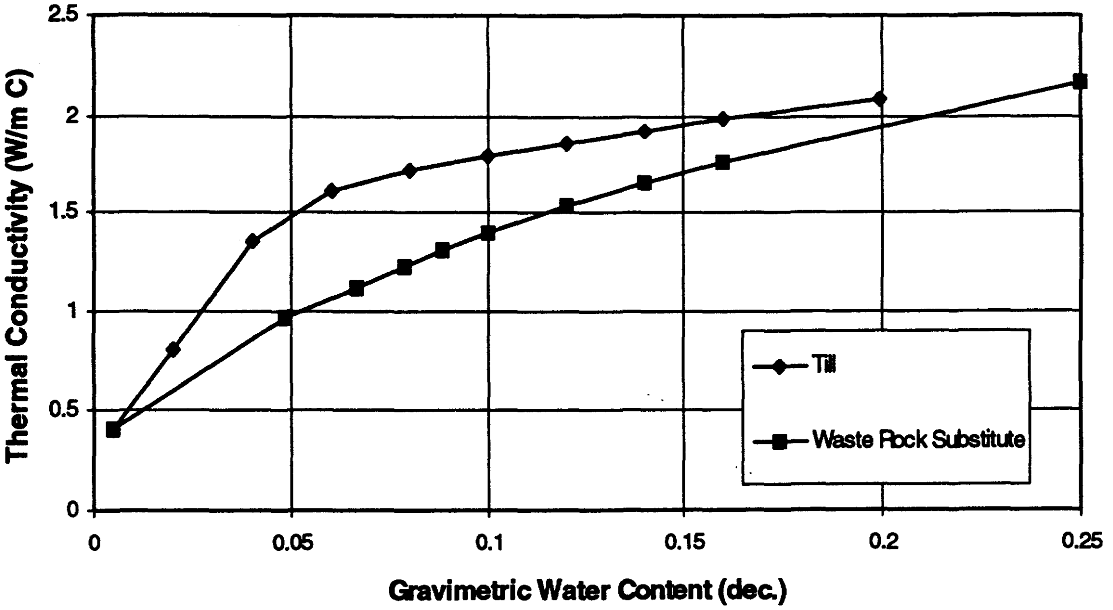


Figure A5 Thermal Conductivity Functions of Cover Materials for Non-Freezing Case (after Swanson, 1995)

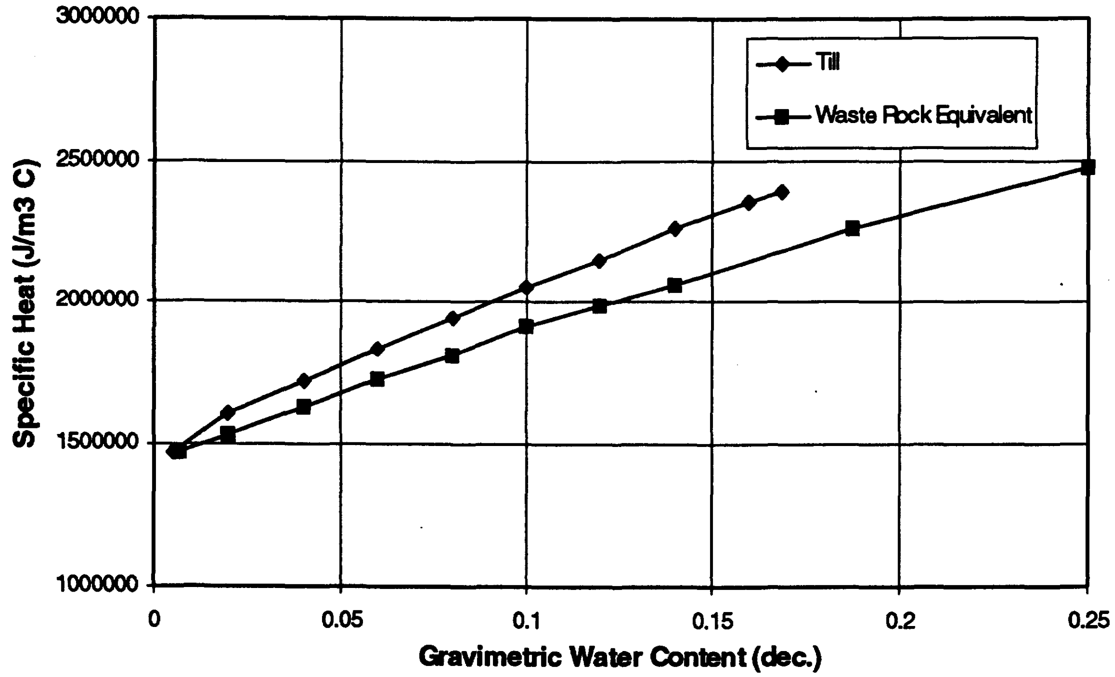


Figure A6 Specific Heat Functions of Cover Material for Non-Freezing Case (after Swanson, 1995)

A3 Modelling Equity Mine Field Data

The geometry of the soil cover system used for modelling consisted of three layers: 30 cm of loose till over 50 cm of compacted till, over 100 cm of waste rock (i.e., simulated using Beaver Creek Sand). The finite element mesh had 33 nodes, with nodal spacings ranging from 2 cm at the soil surface, to 10 cm at the base of the waste rock. Time steps varied from 30 seconds to 21600 seconds (i.e., 6 hours), and the system was considered to have converged if the suctions and temperatures did not change by more than 1% between successive iterations.

Swanson (1995) presents field response modelling results over a time period between the start of May and the end of October, 1993, when freezing temperatures were not encountered. Field weather data collected by O'kane (1995) shows that air and ground temperatures begin to fall below 0°C in early November. The air temperature remains below 0°C until some time in late April or early May. Air temperature data collected by O'kane (1995) for the winter of 1993 / 1994 are presented in Figure A7. The modelling of the field data was carried out over the time period shown in Figure A7.

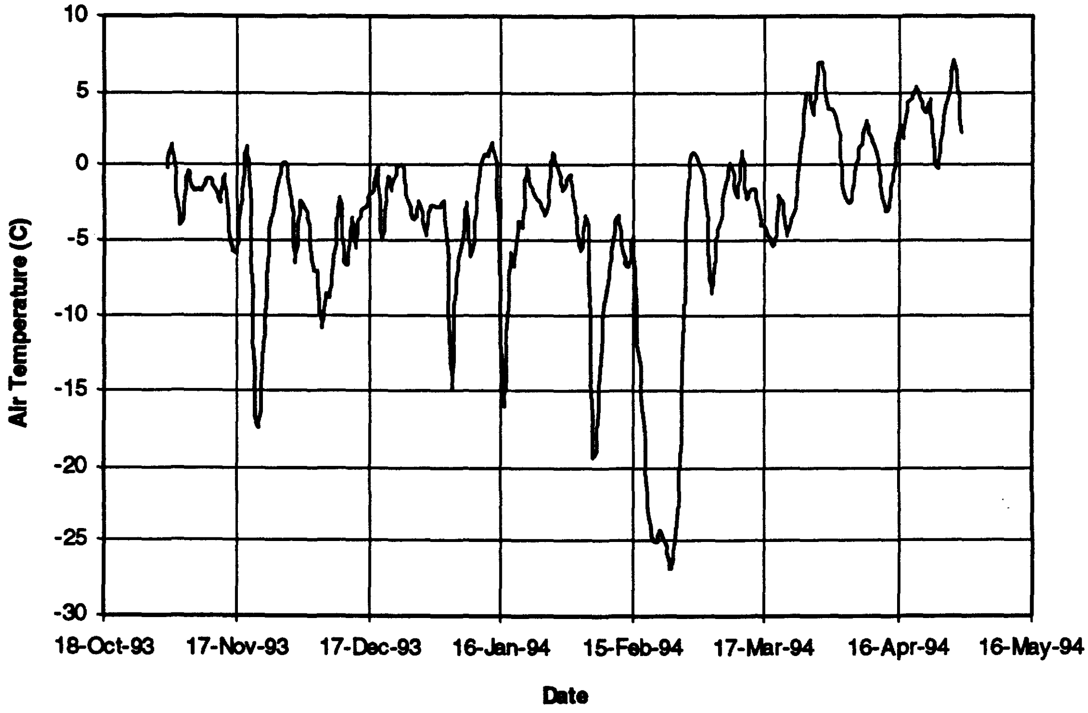


Figure A7 Mean Daily Air Temperatures Over Winter of 1993 / 1994 At Weather Station On Top of Main Waste Rock Dump

The non-freezing version of SoilCover (MEND, 1993) uses air temperature, wind speed, relative humidity, and net radiation to compute the soil temperature at the surface. These computations can only be made in the current version of SoilCover if the soil surface is not covered by snow. A hydrology report prepared for Equity Silver Mines indicates the equivalent of 370 mm of water falls as snow during the winter months at the mine site, and approximately 40% of this value is lost due to melting or sublimation over the winter (Ker, Priestman, 1983). The remaining 60 % melts during the spring thaw. The modified freeze / thaw version of SoilCover does not include heat and mass transfer across a snow layer, therefore it was necessary to make some assumptions about the soil surface boundary conditions. These are subsequently discussed.

Figure A8 shows measured soil temperatures at depths of 5 cm, 10 cm, and 31 cm for the winter of 1993 / 1994. Comparison of the soil temperature profile at 5 cm with the air temperatures from Figure A7 reveals that the soil temperature dropped below 0°C during a cold period in the fall, and again during another period in the late spring. These events most likely occurred because the snow pack in the early winter and late spring was not thick enough to act as an insulating material preventing heat loss from the soil. Figure A8 reveals that the soil remained above freezing during the majority of the winter, and also that there was little temperature difference between 5 cm and 10 cm depths.

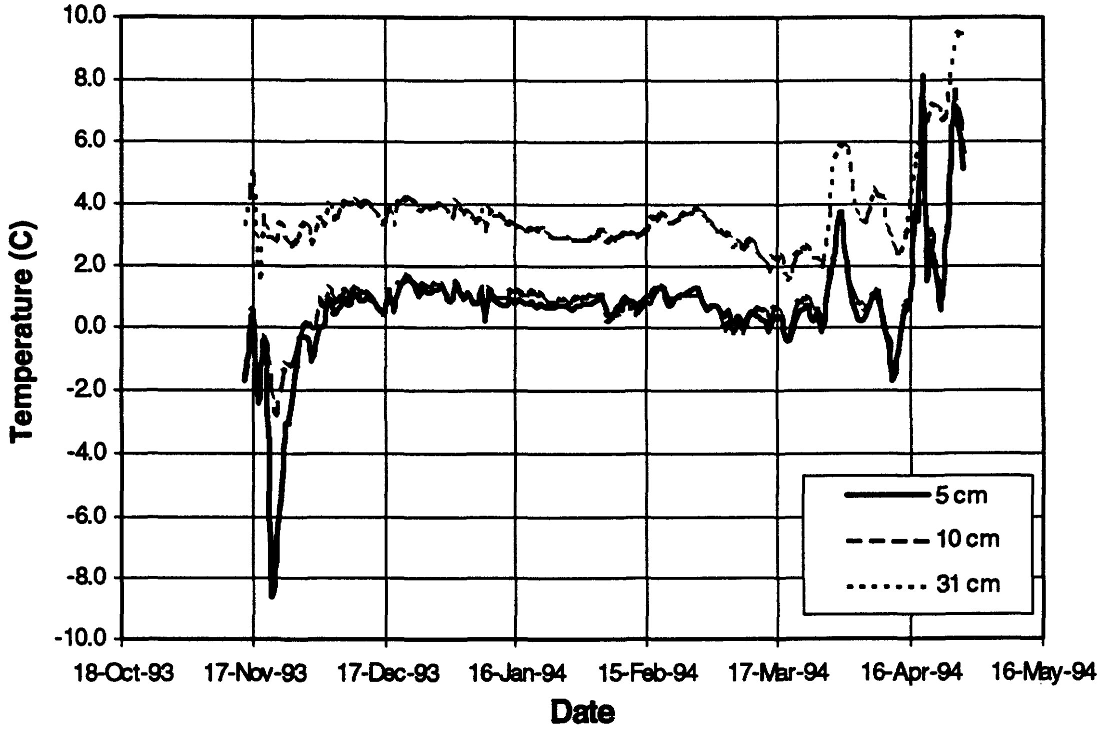


Figure A8 Measured Over Winter Soil Temperatures at Three Different Depths

In order to verify that the modified SoilCover (MEND, 1993) program can model freeze and thaw behaviour in the soil it was necessary to impose temperature and flux boundary conditions on the system. No attempt was made to have the computer model predict energy and evaporative fluxes at the surface. The surface temperature was set to be 0.5°C colder than the 5 cm measured temperature. This seemed reasonable given that there was almost no difference in temperature between 5 cm and 10 cm depths. The warm end temperature at the surface of the waste rock (i.e., at a depth of 1.5m) was assumed constant at 25°C as suggested by Swanson (1995). Based on the hydrology report, the snow equivalent of 222 mm of water was assumed to infiltrate into the cover at an even rate during the last two weeks of the thaw period (i.e., April 23 to May 7).

Simulated and measured temperature profiles at two different depths over the winter of 1993 and 1994 are illustrated in Figure A9. Comparison of computed results with measured results reveals three points. First, there is excellent trend agreement between measured and computed results at the 5 cm depth. This should be expected given the fact that the model boundary conditions were based on the measured values near the surface. The second point to note is that there is excellent agreement between computed and measured results for all times when the soil temperature is above freezing. The third point is that there is poor agreement between measured and computed results at the 30 cm depth when the temperatures are below freezing and ice is present in the soil.

Figure A10 shows the ice content at a depth of 5 cm during the test period, and Figure A11 shows the corresponding thermal conductivity values of the soil at the same depth over the same period.

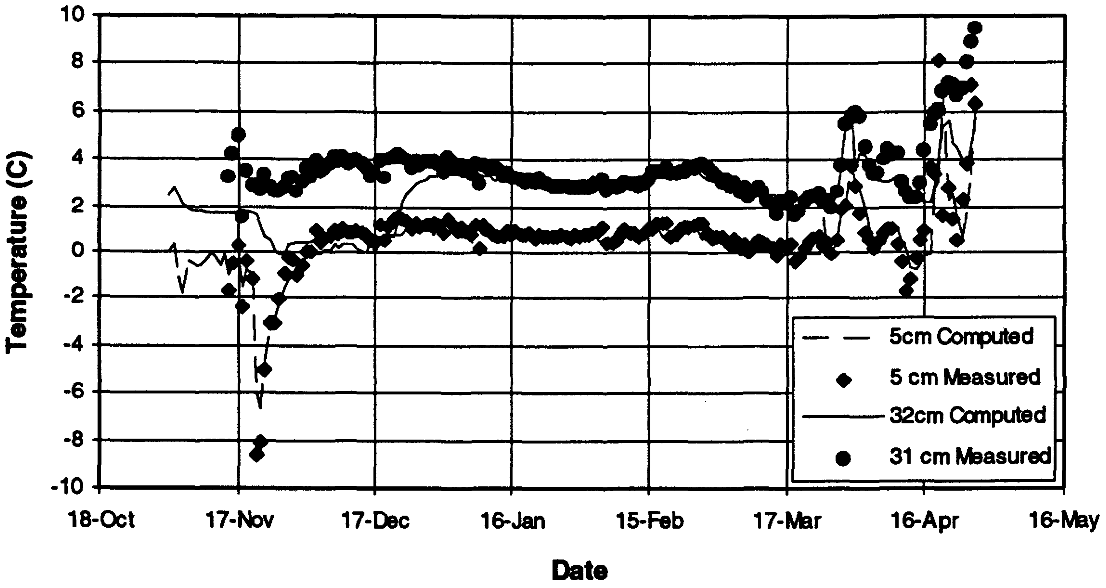


Figure A9 Computed and Measured Soil Temperatures at Two Depths During Field Data Simulation

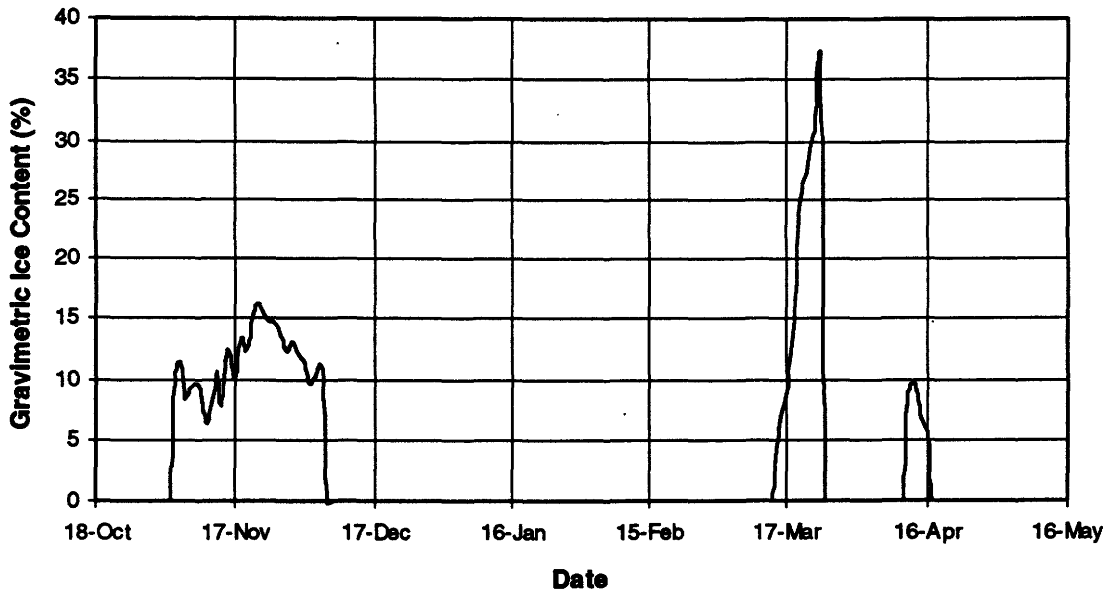


Figure A10 Computed Gravimetric Ice Contents at 5cm Depth During Field Data Simulation

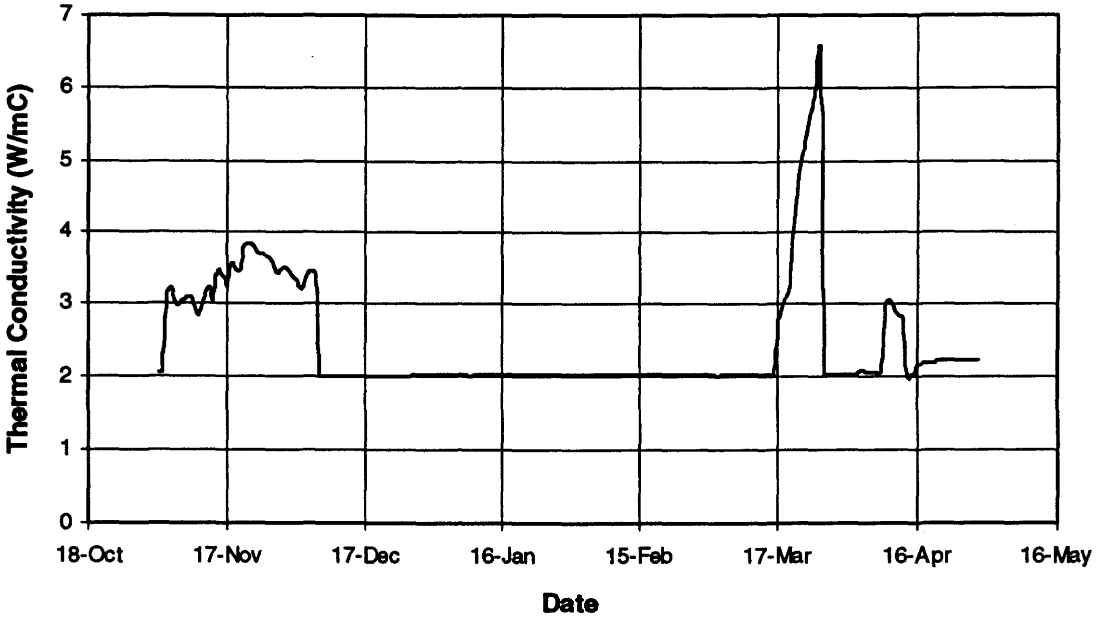


Figure A11 Computed Soil Thermal Conductivity Values at a Depth of 5cm During Field Data Simulation

Figure A11 illustrates the increase in thermal conductivity values that result when ice forms in the soil. This increase is likely the reason for the poor agreement between computed and measured temperature profiles in Figure A9. When too much heat is removed from the soil system, the result is a cold front that advances faster and deeper than it should.. As a frost front advances, ice forms in some of the pores. Because ice has a higher thermal conductivity than water more heat is removed from the warmer soil. In turn, the cold front moves deeper and more water freezes.

Initial observation of the computed thermal conductivity values suggests that they may be computed incorrectly. However, this is not likely the case. There may be some error in thermal conductivity estimations based on the calculation method, but as the thermal conductivity is significantly affected by the ice content (as illustrated in Figure A11) the likely cause of disagreement between computed and measured soil temperature values is the incorrect calculation of ice content. Ice contents are computed incorrectly if the soil freezing curve relationship is not accurate. For example, if the soil freezing curve indicates that ice will form at -0.05^{0}C instead of -0.5°C then higher ice content values will be computed as the temperature drops. In this simulation the soil freezing curve relationship was estimated using a measured soil water characteristic curve and the form of the Clapeyron equation given by Black and Tice (1989). As discussed in Chapter 2, there are problems associated with this approach. An experimentally determined soil freezing curve should yield more accurate results.

Figure A12 shows the computed liquid water contents over the duration of the test. This figure depicts the expected trends. Measured water content data near the surface during the winter months are not available because the thermal conductivity sensors used to measure matric suction do not operate effectively if pore-ice is present. It can be noted that the reductions in liquid water content values occur at the same time there is an increase in ice content (i.e., Figure A10). Figure A12 also shows that there was little redistribution of pore-water from warmer regions to colder regions as illustrated by the constant water content at a 32 cm depth. This is directly attributable to the lower permeability of the clay till, compared with the high permeability and high moisture flux observed in the laboratory testing of silica flour.

During the last two weeks of the simulation period, a snow water equivalent of 15 mm per day was applied as a flux at the top boundary. This value agrees with the snow melt predicted by Ker, Priestman (1983) in their hydrology study of the Equity Silver Mine region. A majority of the 15 mm of water applied each day was computed by SoilCover to be runoff. The amount which did infiltrate contributed to an increase in computed water contents to near saturation levels at the surface. This increase is reflected in Figure A12

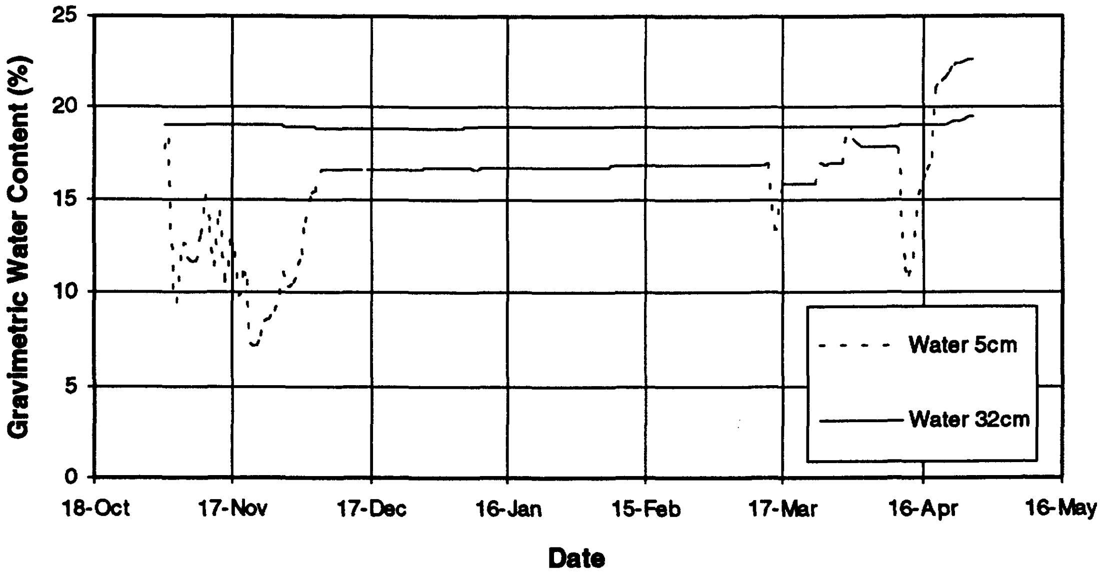
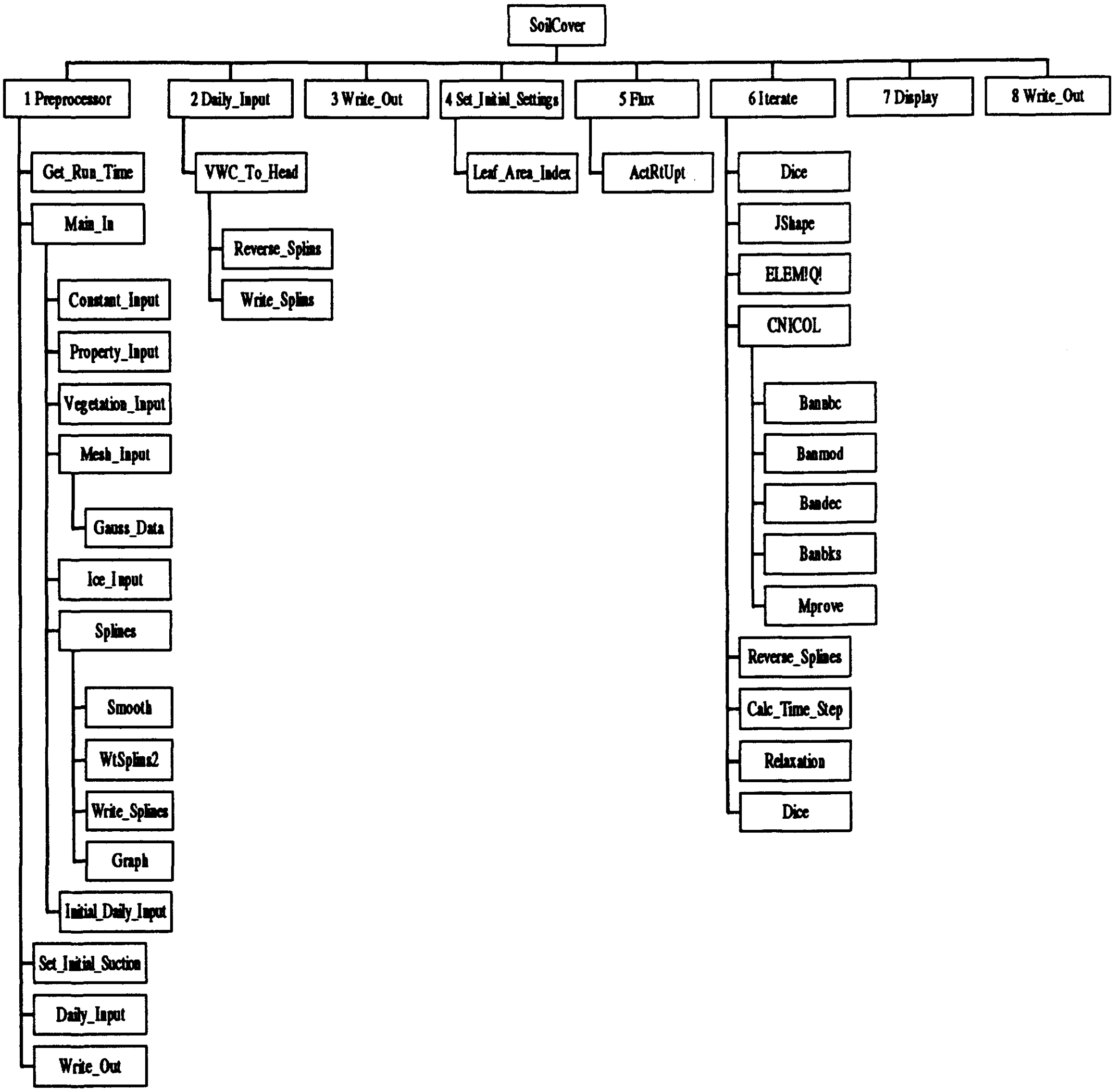


Figure A12 Computed Liquid Water Contents at Two Depths During Field Data Simulations

Finally, some comments can be made about the general trends observed in these results. The objective of a soil cover is to minimize infiltration of oxygen into the waste rock. This is done by keeping the water content in the compacted layer of the soil cover near saturation. Results of the freezing tests on the silica flour (presented earlier) clearly show that there is a large re-distribution of water within the soil. This pattern was not evident during the freezing simulations of the clay till cover and it was not expected. The permeability of the compacted clay till is sufficiently low that it limits the quantity of liquid flux over time. The soil freezing curve for the clay till shows that large quantities of liquid water are present in the soil even at temperatures as low as -5 or -6 °C. If the majority of water did not freeze in the cover, then the matric suctions did not increase sufficiently high enough to draw water out of the warmer soils. As a result, the compacted layer of the soil cover tended to remain saturated which is the desired effect.

APPENDIX B
SUBROUTINE CALL OUT DIAGRAM

Subroutine Call Out Chart



APPENDIX C
MAIN COMPUTER PROGRAM CODE

Note: All revisions for freeze / thaw analysis are in **bold** type.

Program SoilCover

```
C ==============================================================================
C          Version 1.2 June 1994 *** MODIFIED FOR FREEZE / THAW SEPT. 1995 ***
C
C THIS IS THE MAIN PROGRAM FOR THE FINITE ELEMENT MODELLING OF
C          SOIL EVAPORATIVE FLUXES.
C
C  Produced by the Department of Civil Engineering at the University
C  of Saskatchewan, Saskatoon, Saskatchewan, Canada. Postal Code
C  S7N 0W0.
C
C ==============================================================================
      IMPLICIT NONE
      INCLUDE 'FUNCTION.FT'
      INCLUDE 'DECLARE.FT'          ! All include files in this file
      INCLUDE 'SPPCOMMO.FT'          ! Used by SPP ploting routines
C  ============================================================================
      integer  dayclose          ! flag used to close at end of day prematurely
      REAL   Lapsed_Time          ! Total run time in seconds
      character*2 hotkey         ! hot key to close SoilCover prematurely
      INTEGER  ivid,i
      LOGICAL  printed,out
      REAL    water
      REAL    WaterBalance        ! (mm/day) Function to Calculate the WaterBalance
C  ============================================================================

C  ============================================================================
C  Write out program information header
C  ============================================================================
      WRITE(*,*) '                                  '
      WRITE(*,*) '                                  '
      WRITE(*,*) '         SoilCover  Version 1.2   '
      WRITE(*,*) '              June 1994           '
      WRITE(*,*) '       Department of Civil Engineering'
      WRITE(*,*) '       University of Saskatchewan  '
      WRITE(*,*) '        Saskatoon, Saskatchewan    '
      WRITE(*,*) '          Canada  S7N 0W0          '
      WRITE(*,*) '                                  '
      WRITE(*,*) '                                  '
C ==============================================================================

C  ============================================================================
C  Start the preprocessor to do all the preliminary work which includes
C    getting the runtime settings, reading in the data files,
C    setting the initial suctions and temperatures, and writing the
C    initial conditions to the output file.
C  ============================================================================
      TTIME   = 0.0D0
      DAYCLOSE = 1
      AEsum   = 0.0D0
      PEsum   = 0.0D0
      ATsum   = 0.0D0
      PTsum   = 0.0D0
      RunOff  = 0.0D0
      DO i=1,NNODES
        SFLUX  (i) = 0.0D0
        SFLUXL  (i) = 0.0D0
        SFLUXV  (i) = 0.0D0
        SFLUXARU(i) = 0.0D0
        SFLUXPRU(i) = 0.0D0
      ENDDO
      MAXD_OUT_TODAY = 0.0  ! Clear daily maxd_out counter
      CALL PREPROCESSOR
      out=true
C  ============================================================================
C             ENTERING THE DAY LOOP
C  ============================================================================
      DO NDAY=1,DAYS
```

```
      CALL DAILY_INPUT
C  --------------------------------------------
C     Initialize total time and flux counter
C  --------------------------------------------
      TTIME   = 0.0D0
      AEsum   = 0.0D0
      PEsum   = 0.0D0
      ATsum   = 0.0D0
      PTsum   = 0.0D0
      RunOff  = 0.0D0
      SHUTDOWN = FALSE      ! We will accept Qspec>Qsat
      printed  = FALSE
      MAXD_OUT_TODAY = 0.0      ! Clear daily maxd_out counter
      DO i=1,NNODES
        SFLUX  (i) = 0.0D0
        SFLUXL  (i) = 0.0D0
        SFLUXV  (i) = 0.0D0
        SFLUXARU(i) = 0.0D0
        SFLUXPRU(i) = 0.0D0
      ENDDO
C  ================================================================
C              ENTERING THE TIME LOOP
C  ================================================================
      DO WHILE( TTIME.LT.86400.0D0 )         ! 1 day = 86400 seconds
        hotkey = INKEY()              ! looks for hot key
        IF (ICHAR(hotkey(1:1)).EQ.17)THEN
         CLOSE(IUNITV)               ! Close the graphics screen
         CALL GMODE(IVID)               ! Restore the display to original settings
         GOTO 888
        ELSEIF(ICHAR(hotkey(1:1)).EQ.5)THEN
         DAYCLOSE=2                  ! sets flag to close at end of day (CTRL E)
        ELSEIF(ICHAR(hotkey(1:1)).EQ.2)THEN   ! Prints out the current values before ending

         IF(DETAILED)THEN
          CALL WRITE_OUT(water)
         ELSE
          CALL WRITE_NOD(water)
         ENDIF
         CLOSE(IUNITV)               ! Close the graphics screen
         CALL GMODE(IVID)               ! Restore the display to original settings
         GOTO 888
        ENDIF
        CALL SET_INITIAL_SETTINGS         ! Performs all non-iterative calculations
c       if(ttime.lt.1.and.nday.eq.1) ttime=21600
        CALL FLUX                    ! Calc AE, PE, and surface temp.
        CALL ITERATE                   ! Iterative solution calculations
        TTIME   = TTIME   + DELTAT        ! (secs)  Update the total run time for current day

        DO i=1,NNODES
         SFLUX(i)  = SFLUX (i) + VFLUX(i) *DELTAT ! (mm/day) Update the flux sum
         SFLUXL(i) = SFLUXL(i) + FLUX_L(i)*DELTAT
         SFLUXV(i) = SFLUXV(i) + FLUX_V(i)*DELTAT
        ENDDO
        IF( VEGETATION.AND.(LAI.GE.0.1) )THEN
          VFLUXAT = 0.0
          DO i=RootTop(2),RootDepth(2)
           VFLUXAT    = VFLUXAT    + ARU(i)   ! (mm/day) actual root uptake for each time step
           SFLUXARU(i) = SFLUXARU(i) + ARU(i)
     1            /NodeContrib(i)*DELTAT  ! (mm/day/cm) aru daily total for each node
           SFLUXPRU(i) = SFLUXPRU(i) + PRU(i)
     1            /NodeContrib(i)*DELTAT  ! (mm/day/cm) PRU daily total for each node
          ENDDO
          ATsum   = ATsum + VFLUXAT*DELTAT  ! (mm/day) update the act trans flux
          PTsum   = PTsum + VFLUXPT*DELTAT  ! (mm/day) update the PT flux
        ELSE
          VFLUXAT = 0.0
          VFLUXPT = 0.0
        ENDIF
        AEsum  = AEsum + PENMAN   *DELTAT   ! (mm/day) Update the AE flux
        PEsum  = PEsum + VFLUXPE  *DELTAT   ! (mm/day) Update the PE flux
        RunOff = RunOff+ INCRUNOFF*DELTAT    ! (mm/day) Update the total daily runoff
```

```
      water = WaterBalance()          ! (mm/day) Calculate the current water balance
      IF(GRAPHICS) THEN
        CALL DISPLAY(ivid,VFLUXPE,PENMAN,VFLUXAT,RAIN,water)
      ENDIF
      IF(.NOT.printed.AND.PrintTime.EQ.1)THEN   ! Print output at noon
       If(TTIME.GE.43200.D0 )THEN
        printed = TRUE
        IF(DETAILED)THEN
          CALL WRITE_OUT(water)
        ELSE
          CALL WRITE_NOD(water)
        ENDIF
       ENDIF
      ENDIF
c      write(*,*) 'day,ksat,ttime',nday,satk(3),ttime
      if(nday.eq.1 .and. ttime.ge.21600 .and.
   1     out .and. ttime.le.22600) then  ! 6 hour output
      call write_nod(water)
      out=false
      elseif(nday.eq.1 .and. ttime.ge.43200 .and. .not.out)then  ! 12 hour output
      call write_nod(water)
      out=true
      endif

     ENDDO  ! DO WHILE( TTIME.LT.86400.0D0 )
C    ===========================================================
C            FINISHED THE TIME LOOP
C    ===========================================================


     IF(PrintTime.EQ.2)THEN                   ! Print output at midnight
       IF(DETAILED)THEN
        CALL WRITE_OUT(water)
       ELSE
        CALL WRITE_NOD(water)
       ENDIF
     ENDIF

C    ===========================================================
C     Resetting the time step to the initial time step specified
C    to reduce the shock on the system induced by the new bondary
C    conditions applied on a new day.
C    ===========================================================
     DELTAT = FIRST_DELTAT

     IF(GRAPHICS)THEN
       CLOSE(IUNITV)       ! Close the graphics screen
       CALL GMODE(IVID)      ! Restore the display to original settings
     ENDIF
     IF(DAYCLOSE.EQ.2)THEN      ! Closes SoilCover due to hot key
      CLOSE(IUNITV)         ! Close the graphics screen
      CALL GMODE(IVID)        ! Restore the display to original settings
      GOTO 888
     ENDIF
   ENDDO  ! DO NDAY = 1,DAYS

C   ============================================================
C            FINISHED THE DAY LOOP
C   ============================================================
C   ==========================================================
C            Determine total run time
C   ==========================================================
   Lapsed_Time = SECNDS(TIME0)/60.0
   WRITE( *,777) Lapsed_Time
   WRITE(48,777) Lapsed_Time
777  FORMAT(' ','Total run time = ',F8.2,' minutes')
C  ------------------------------------
888  CLOSE(UNIT=48)    ! Close the outfile
   CLOSE(UNIT=12)    ! Close the daily data file
   IF( VEGETATION )THEN
     CLOSE(UNIT=10)    ! Close the vegetation file
```

```
      ENDIF
C     ================================================================
      END  ! Of program SoilCover
```

APPENDIX D
SUBROUTINE CODE

```
C   This subroutine calculates the actual root uptake for each node
C   ==========================================================
      SUBROUTINE ActRtUpt(node)          !(mm/sec)
C   ==========================================================
      IMPLICIT NONE               ! Ensure that all variables have been correctly defined
      INCLUDE 'FUNCTION.FT'          ! Contains all FUNCTION declarations
      INCLUDE 'DECLARE.FT'           ! Contains all common block declarations
C   ==========================================================
      INTEGER  node       ! (HRS) Number of hours in a day
      REAL    limitFactor  ! Plant Limiting Factor
C   ==========================================================
      limitFactor = Calc_PlantLimitFactor(SUCNOD(node))  ! calcs PLF
      ARU(node)   = PRU(node) * limitFactor
C   ==========================================================
      RETURN
      END
```

SUBROUTINE banbks(a,n,m1,m2,np,mp,al,mpl,indx,b)

```
      INTEGER m1,m2,mp,mpl,n,np,indx(n)
      double precision a(np,mp),al(np,mpl),b(n)
      INTEGER i,k,l,mm
      double precision dum
      mm=m1+m2+1
      if(mm.gt.mp.or.m1.gt.mpl.or.n.gt.np) pause 'bad args in banbks'
      l=m1
      do 12 k=1,n
       i=indx(k)
       if(i.ne.k)then
        dum=b(k)
        b(k)=b(i)
        b(i)=dum
       endif
       if(l.lt.n)l=l+1
       do 11 i=k+1,l
        b(i)=b(i)-al(k,i-k)*b(k)
11      continue
12     continue
      l=1
      do 14 i=n,1,-1
       dum=b(i)
       do 13 k=2,l
        dum=dum-a(i,k)*b(k+i-1)
13      continue
       b(i)=dum/a(i,1)
       if(l.lt.mm) l=l+1
14     continue
      return
      END
C  (C) Copr. 1986-92 Numerical Recipes Software '7-,31..
```

SUBROUTINE bandec(a,n,m1,m2,np,mp,al,mpl,indx)

```
      INTEGER m1,m2,mp,mpl,n,np,indx(n)
      double precision a(np,mp),al(np,mpl),TINY
      PARAMETER (TINY=1.D-20)
      INTEGER i,j,k,l,mm
      double precision dum
      mm=m1+m2+1
      if(mm.gt.mp.or.m1.gt.mpl.or.n.gt.np) pause 'bad args in bandec'
      l=m1
      do 13 i=1,m1
       do 11 j=m1+2-i,mm
```

```
        a(i,j-l)=a(i,j)
11      continue
       l=l-1
       do 12 j=mm-l,mm
        a(i,j)=0.0D0
12      continue
13     continue
      l=m1
      do 18 k=1,n
       dum=a(k,1)
       i=k
       if(l.lt.n)l=l+1
       do 14 j=k+1,l
        if(dabs(a(j,1)).gt.dabs(dum))then
          dum=a(j,1)
          i=j
        endif
14      continue
       indx(k)=i
       if(dum.eq.0.0D0) a(k,1)=TINY
       if(i.ne.k)then
        do 15 j=1,mm
          dum=a(k,j)
          a(k,j)=a(i,j)
          a(i,j)=dum
15        continue
       endif
       do 17 i=k+1,l
        dum=a(i,1)/a(k,1)
        al(k,i-k)=dum
        do 16 j=2,mm
          a(i,j-1)=a(i,j)-dum*a(k,j)
16        continue
        a(i,mm)=0.0D0
17      continue
18     continue
      return
      END
C  (C) Copr. 1986-92 Numerical Recipes Software '7~,31..
```

SUBROUTINE BANMOD(a,n,m1,np,mp,b)

```
C  ==========================================================
C     This subroutine modifies the system stiffness matrices so
C   that the suction and temperature boundary conditions specified
C   will be removed from the simultaneous equations.
C  ==========================================================
      IMPLICIT NONE                 ! Ensure that all variables have been correctly defined
      INCLUDE 'DECLARE.FT'             ! Contains all common block declarations
C  ==========================================================
      INTEGER n,np,mp,m1              ! array size used and physical size of array
      double precision   a(np,mp)     ! The [a] matrix
      double precision   b(np)        ! The {b} vector
      double precision   ebcw         ! Current Suction boundary condition
      double precision   ebch         ! Current Temperature boundary condition
      INTEGER i,j,k,mm                 ! Loop Counters
      INTEGER nx1                     ! Node which to apply current suction boundary condition
      INTEGER nx2                     ! Node which to apply current temperature boundary condition
C  ==========================================================
      mm = 2*m1 + 1
      DO k = 1,NEBC
            ebcw  = EBW  (k,2)          ! Suction boundary condition
            ebch  = EBH  (k,2)         ! Temperature boundary condition
            nx1   = NODEB(k,2)*2-1        ! Node to apply suction
            nx2   = nx1 + 1             ! Node to apply temperature
C    ----------------------------------------------
            IF( EBW(k,2).EQ.1E10 )THEN
              ebcw = 0.0D0
            ENDIF
```

```
          IF( EBH(k,2).EQ.1E10 )THEN
            ebch = 0.0D0
          ENDIF
C     ---------------------------------------------
          i = 1
          DO i = 1,(mm-1)
           j = nx1 + m1 + 1 - i
           IF( j.GE.1 .AND. j.LE.n )THEN
             b(j) = b(j) - ebcw*a(j,i) - ebch*a(j,i+1)
           ENDIF
          ENDDO
C     ---------------------------------------------
          IF( EBW(k,2).NE.1E10 )THEN
           DO i = 1,mm
             a(nx1,i) = 0.0D0  ! Zero row of matrix at EBC node.
           ENDDO
           j = nx1 - m1
           i = mm
           DO WHILE( i.GT.0 .AND. j.LE.n ) ! Zero column of matrix at EBC node.
             IF( i.LE.mm .AND. j.GT.0 )THEN
               a(j,i) = 0.0D0
             ENDIF
             j = j + 1
             i = i - 1
           ENDDO
           a(nx1,m1+1) = 1.0D0
           b(nx1)     = ebcw
          ENDIF
C     ---------------------------------------------
          IF( EBH(k,2).NE.1E10 )THEN
           DO i = 1,mm
             a(nx2,i) = 0.0D0  ! Zero row of matrix at EBC node.
           ENDDO
           j = nx2 - m1
           i = mm
           DO WHILE( i.GT.0 .AND. j.LE.n ) ! Zero column of matrix at EBC node.
             IF( i.LE.mm .AND. j.GT.0 )THEN
               a(j,i) = 0.0D0
             ENDIF
             j = j + 1
             i = i - 1
           ENDDO
           a(nx2,m1+1) = 1.0D0
           b(nx2)     = ebch
          ENDIF
      ENDDO  ! DO k = 1,NEBC
C     ==========================================================
      RETURN
      END
```

SUBROUTINE banmul(a,n,m1,m2,np,mp,x,b)

```
      INTEGER m1,m2,mp,n,np
      double precision a(np,mp),b(n),x(n)
      INTEGER i,j,k
      do 12 i=1,n
        b(i)=0.0D0
        k=i-m1-1
        do 11 j=max(1,1-k),min(m1+m2+1,n-k)
          b(i)=b(i)+a(i,j)*x(j+k)
11      continue
12    continue
      return
      END
C  (C) Copr. 1986-92 Numerical Recipes Software '7~,31..
```

```
C  ==================================================================
        SUBROUTINE bannbc(Iteration,lsys,B)
C  ==================================================================
C     This subroutine determines the surface flux for addition to the
C    system {b} vector.
C  ==================================================================
        IMPLICIT NONE                    ! Ensure that all variables have been correctly defined
        INCLUDE 'FUNCTION.FT'               ! Contains all function declarations
        INCLUDE 'DECLARE.FT'                ! Contains all common block declarations
C  ==================================================================
        double precision actual          ! (mm/s)  Actual flux to apply
        double precision B (MAX_NODESx2)  !      The global load vector
        REAL   delay                     ! (hr) time at which const. precip. occurs
        INTEGER i                        ! Loop Counter
        real   area                      ! the area below the unit infiltration curve
        INTEGER Iteration                  ! The current iteration
        INTEGER lsys                     ! 2 * NNODES
        INTEGER node                      ! Node at which to apply boundary condition
        INTEGER row                       ! Node at which to apply boundary condition
        REAL   saturated                 ! (mm/s)  Saturated Hydraulic Conductivity
        REAL   spec_top_flux               ! (mm/s)  User specified flux
        REAL   spec_bot_flux               ! (mm/s)  User specified flux
C  ==================================================================
        type=soiltype(1)
        INCRUNOFF = 0.0D0              ! Initialize run off
        saturated = SATK(TYPE)*10.0 ! (mm/s) saturated mass flow
        IF(QW(1,2).EQ.1.0E+20)THEN
          spec_top_flux = 0.0
        ELSEIF(Duration(2).GT.0.0)THEN
          delay = Duration(2)            ! (hr) time when constant precip. occurs
          area = 10000*(EXP(delay*3600./10000.)-EXP(0/10000.))+
     1        EXP(delay*3600./10000.)*(86400.-delay*3600.)
          IF(TTIME.LT.delay*3600)THEN  ! If during ramp period
                spec_top_flux = QW(1,2)*86400000/area
     1          *EXP(TTIME/10000. 0)
          ELSE                         ! if during constant precip. period
                spec_top_flux = QW(1,2)*86400000/area
     1          *EXP(delay*3600/10000)
          ENDIF
        ELSE                           ! if not specified to be ramped
          spec_top_flux = QW(1,2)*1000.0
        ENDIF
        row = 1                        ! Re-Organized matrix row
        actual = spec_top_flux + PENMAN
        IF( SHUTDOWN )THEN
          IF( Iteration.LT.3 .AND. VFLUX(1).GT.spec_top_flux )THEN
                SHUTDOWN = FALSE
                EBW(1,2) = 1.0E+10
                VFLUX(1) = actual
          ELSE
                EBW(1,2) = 0.0
                actual  = 0.0D0
          ENDIF
          INCRUNOFF = spec_top_flux - VFLUX(1)        ! Calculating RunOff
        ELSEIF( Iteration.GT.1 .AND. SUCNOD(1).LE.0.0
     1   .AND. actual.GT.0.0 )THEN
          SHUTDOWN  = TRUE                    ! Shutdown Infiltration for rest of day
          EBW(1,2)  = 0.0
          actual   = 0.0D0
          INCRUNOFF = spec_top_flux - VFLUX(1)    ! Calculating RunOff
          IF( INCRUNOFF.LT.0.0D0 )THEN
                INCRUNOFF = 0.0D0
          ENDIF
        ELSE
          VFLUX(1) = actual
        ENDIF
        RAIN = spec_top_flux - INCRUNOFF
        B(1) = B(1) + actual*DELTAT/1000.0D0
C  ==================================================================
```

```
      IF(QW(2,2).EQ.1.0E+20)THEN
        spec_bot_flux = 0.0
      ELSE
        spec_bot_flux = QW(2,2)*1000.0
        VFLUX(NNODES) = spec_bot_flux
        row = 2*NNODES - 1                 ! Re-Organized matrix row
        B(row) = B(row) + spec_bot_flux*DELTAT/1000.0D0
      ENDIF
C   -------------------------------------------------
C   Apply root uptake flux over the nodes throughout the root depth
C   if a vegetation was specified
C   -------------------------------------------------
      IF( VEGETATION.AND.(LAI.GE.0.1) )THEN
        DO i=RootTop(2),RootDepth(2)             ! Add in act root uptake
              row   = 2 * i - 1
              B(row) = B(row) + ARU(i) * DELTAT / 1000.0D0
        ENDDO
      ENDIF
      RETURN
      END
```

```
C   ===============================================================
```

SUBROUTINE CALCULATE_TIME_STEP

```
C   ===============================================================
C     This subroutine calculates the maximum time step allowed which
C   will not result in the temperatures or suctions changing by more
C   than the amount specified in TOLS and TOLT.
C   Subroutines used by calculate_time_step:
C       - banmul: banded matrix multiplication.
C
C   ===============================================================
    IMPLICIT NONE                  ! Ensure that all variables have been correctly defined
    INCLUDE 'DECLARE.FT'             ! Contains all common block declarations
C   ===============================================================
    real        diff               ! Difference in actual change to maximum change
    INTEGER       i,k                  ! Loop Counters
    INTEGER       node                  ! Controlling node
    INTEGER       lsys                 ! 2 * NNODES
    real        min_mult              ! The minimum time step multiplier
    real        sc                 ! Maximum allowable change in suction
    real        tc                 ! Maximum allowable change in temperature
C   ===============================================================
C   Calculating ({x}-{y}) & ({x}+{y})
C   ===============================================================
    lsys = 2 * NNODES
    sc  =  TOLS               ! Maximum suction change
    tc  =  TOLT               ! Maximum temperature change
    min_mult = MAX_DELTAT/DELTAT
    DO k = 1,NNODES
     i = k + NNODES
     diff = ABS( SUCNOD(k)-PHIA(k) )
     IF( diff.GT.0.0 )THEN
       diff = ABS( sc*PHIA(k)/diff )
     ELSE
       diff = min_mult
     ENDIF
     IF( diff.LT.min_mult )THEN
      min_mult = diff
      node    = k
     ENDIF
     diff = ABS( TEM(k)-PHIA(i) )
     IF( diff.GT.0.0 )THEN
       diff = ABS( tc*PHIA(i)/diff )
     ELSE
       diff = min_mult
     ENDIF
     IF( diff.LT.min_mult )THEN
      min_mult = diff
```

```
        node   = k
      ENDIF
     ENDDO
     DELTAT = DELTAT * min_mult
     IF( DELTAT.LT.MIN_DELTAT )THEN
       DELTAT = MIN_DELTAT
     ENDIF
c    write(*,99)deltat,node,X1(node),min_mult,ttime
c99  format(' ',F9.3,I6,F14.4,F18.6,F14.3)
c    ---------------------------------------------------
c     Ensure that the simulation time during the current day doesn't
c    exceed the amount of time in a day.
c    ---------------------------------------------------
     IF( DELTAT+TTIME.GT.86400.0D0 )THEN
       DELTAT = 86400.0D0-TTIME
     ENDIF
C    ==================================================================
     RETURN
     END
```

```
C    ==================================================================
```

SUBROUTINE CNICOL(Iteration)

```
C    ==================================================================
C      This subroutine solves for the new nodal suctions and temperatures
C    using the Crank Nicholson time marching scheme.
C    Subroutines used by cnicol:
C         - banmul: banded matrix multiplication.
C         - banmod: finite element modification for banded matrices
C         - bandec: splits banded matrices into an upper & lower matrix
C         - banbks: performs back substitution on the upper & lower matrices
C                to solve for the unknowns.
C
C    ==================================================================
C
C       [a]     {x}    =            [b]
C    _________________  t+dt  ______________________________________
C    |           |    |                      |
C                       [p]
C                  _______________
C                  |          |
C
C    (     dt   )       (     dt   )      dt (          )
C    ( [C] + ---[K] ){x}   = ( [C] - ---[K] ){x}  + --- ({F} +{F}   )
C    (      2   )  t+dt (       2   )  t   2 (  t   t+dt)
C
C    ==================================================================
     IMPLICIT NONE                    ! Ensure that all variables have been correctly defined
     INCLUDE 'FUNCTION.FT'                ! Contains all function declarations
     INCLUDE 'DECLARE.FT'                ! Contains all common block declarations
C    ==================================================================
     INTEGER    MAX_BAND
c    integer soil
     PARAMETER( MAX_BAND = 11 )          ! For a quadratic element
     DOUBLE PRECISION THETA,THETA1
     PARAMETER( THETA  = 0.5D0    )
     PARAMETER( THETA1 = 1.D0-THETA)
C    ==================================================================
     double precision a   (MAX_NODESx2,MAX_BAND)    ! The banded 'a' matrix where [a]{x} = {b}
     double precision au  (MAX_NODESx2,MAX_BAND)    ! The upper half of the lu decomposition of 'a'
     double precision al  (MAX_NODESx2, 5     )     ! The lower half of the lu decomposition of 'a'
     double precision b   (MAX_NODESx2)             ! The 'b' vector where [a]{x} = {b}
     double precision f0  (MAX_NODESx2)             ! The load vector ( @t   )
     double precision f1  (MAX_NODESx2)             ! The load vector ( @t+dt )
     INTEGER indx(MAX_NODESx2)                      ! Used by Numerical Recipies bandec & banksb & mprove
     INTEGER Iteration                       ! The current iteration
     INTEGER i,j,k,l,m,type                       ! Loop Counters
     INTEGER m1,m2,mm                             ! Lower and Upper Half Band Widths, and total band width
     INTEGER lsys                            ! 2 * NNODES
```

```
    double precision stor(MAX_NODESx2,MAX_BAND)   ! The banded version of the storage matrix.
    double precision p  (MAX_NODESx2,MAX_BAND)   ! Matrix = [C]-(dt/2)*[K] NOTE: This matrix has been rearranged and
banded
    double precision stif(MAX_NODESx2,MAX_BAND)   ! The banded version of the stiffness matrix.
    double precision x  (MAX_NODESx2)            ! The 'x' vector where [a]{x} = {b}
   save  f0,f1                                 ! This must be saved between calls

C  ================================================================
C  Forming the banded versions of the load vector, the vector of
C  knowns, the stiffness matrix, and the storage matrix.
C  ================================================================
    IF( PNODES.EQ.2 )THEN    ! Linear Element
      m1 = 3
      m2 = 3
    ELSE                    ! Quadratic Element
      m1 = 5
      m2 = 5
    ENDIF
    mm  = m1 + m2 + 1
    lsys = 2 * NNODES
    IF( Iteration.EQ.0 )THEN
    DO i = 1,lsys
      f0(i) = f1(i)
    ENDDO
    ENDIF
    DO j = 1,lsys

    l = 2*j - 1                ! Re-Organized matrix column
    IF(l.GT.lsys)THEN              ! Temperature related
      l = l - lsys + 1
    ENDIF
    f1(l) = SYSF(j)              ! Load vector @ t+dt
    IF( j.GT.NNODES )THEN          ! Then this is temperature
      x (l) = PHIA(j)          ! {x}  vector @ t
    ELSE
      x (l) = -PHIA(j)         ! {x}  vector @ t
    ENDIF

    DO i = 1,lsys
     k = 2*i - 1               ! Re-Organized matrix row
     IF(k.GT.lsys)THEN              ! Temperature related
       k = k - lsys + 1
     ENDIF
     m = (l+m1+1-k)             ! Banded matrix row
     IF( m.GE.1 .AND. m.LE.mm )THEN  ! If row falls within the band
       stif(k,m) = SYSTIF(i,j)
       stor(k,m) = Lump(i,j,m1,m2,lsys)
     ENDIF
    ENDDO

    ENDDO
C  ================================================================
    IF( Iteration.EQ.0 .AND. TTIME.EQ.0.0D0 .AND. NDAY.EQ.1 )THEN
      DO i = 1,lsys
        f0(i) = f1(i)
      ENDDO
    ENDIF
C  ================================================================
C  FORMING THE a AND p MATRICES(REF:SEGERLIND)
C  ================================================================
    IF( TRANSIENT )THEN
     do m = 1,mm
       do k = 1,lsys
         a(k,m) = stor(k,m) + THETA *DELTAT*stif(k,m)
         p(k,m) = stor(k,m) - THETA1*DELTAT*stif(k,m)
       enddo
     enddo
    ELSE
     do m = 1,mm
       do k = 1,lsys
         a(k,m) = stif(k,m)
```

```
        p(k,m) = 0.0D0
      enddo
    enddo
   ENDIF
C  ================================================================
C  Applying Boundary Conditions
C  ================================================================
c  CALL bannbc(iteration,lsys,b)          ! Modification for Natural Bndry Cond.
c  CALL banmod(a,lsys,m1,MAX_NODESx2,MAX_BAND,b) ! Modification for Essent. Bndry Cond.
C  ================================================================
C  Calculating the {b} vector
C  ================================================================
   IF( TRANSIENT )THEN
     CALL banmul(p,lsys,m1,m2,MAX_NODESx2,mm,x,b)
     DO i = 1,lsys
       b(i) = b(i) + DELTAT*( THETA1*f0(i) +THETA*f1(i) )
     ENDDO
   ELSE                 ! This is a steady state analysis
     DO i = 1,lsys
       b(i) = f1(i)
     ENDDO
   ENDIF
C  ================================================================
C  Applying Boundary Conditions
C  ================================================================
   CALL bannbc(iteration,lsys,b)          ! Modification for Natural Bndry Cond.
   CALL banmod(a,lsys,m1,MAX_NODESx2,MAX_BAND,b) ! Modification for Essent. Bndry Cond.
C  ================================================================
C  Making A Copy of the 'A' Matrix and 'B' Vector
C  ================================================================
   DO i = 1,lsys
     x(i) = b(i)
     DO j = 1,mm
       au(i,j) = a(i,j)
     ENDDO
   ENDDO
C  ================================================================
C  Solving the Equations
C  ================================================================
   CALL bandec(au,lsys,m1,m2,MAX_NODESx2,MAX_BAND,al,5,indx)  ! Splitting A into upper & lower matrices
   CALL banbks(au,lsys,m1,m2,MAX_NODESx2,MAX_BAND,al,5,indx,x) ! Solving for the unknowns
   CALL mprove(au,lsys,m1,m2,MAX_NODESx2,MAX_BAND,al,5,indx,x,a,b) ! Improving Accuracy
C  ================================================================
C  Updating the nodal suctions and temperatures with the new
C     suctions and temperatures which have been placed in [b] by
C     banbks.
C  ================================================================
   DO i = 1,NNODES
     j = 2*i - 1
     k = 2*i
     SUCNOD(i) = -x(j)     ! {  Suction  } @t+dt
     TEM  (i) =  x(k)     ! {Temperature} @t+dt
   ENDDO
C  ================================================================
   RETURN
   END
```

```
C     This function is used by the subroutine 'ITERATE' to determine
C  if convergence has been achieved.
C  ================================================================
```

LOGICAL FUNCTION Convergence()

```
C  ================================================================
   IMPLICIT NONE
   INCLUDE 'DECLARE.FT'              ! Contains all common block declarations
C  ================================================================
   INTEGER i,soil                   ! Loop Counter
   LOGICAL converged                ! Logical flag to indicate when system has converged
```

```
      REAL   diff              ! Relative change in Suction or Temperature
      REAL   volwc,voluwc
      REAL   fn_point
C   ==================================================================
      converged = TRUE
      i = 0
      DO WHILE( i.LT.NNODES .AND. converged )
        i = i + 1

C     ----------------------------------------
C      Convergence for suction is based upon a relative convergence
C        which is checked at every node.
C     ----------------------------------------
        IF(PRESNOD(i).NE.0.0E0)THEN       ! Prevent division by zero
          diff =ABS( SUCNOD(i)-PRESNOD(i) )/PRESNOD(i)
        ELSEIF(SUCNOD(i).NE.0.0E0)THEN
          diff = 1.0E0
        ENDIF

        IF(diff.GT.PUSNORM)THEN
         converged = FALSE
         IF( STEADYSTATE)THEN
           WRITE(*,1)i,diff*100.0,SUCNOD(i)
 1          FORMAT('   Node',I3,'   Change ',F6.2,
   1             '%    Suction',G17.4)
         ENDIF
        ENDIF

C     ----------------------------------------
C      Convergence for temperature is based upon a relative conv.
C        which is checked at every node.
C     ----------------------------------------
        IF(PRETNOD(i).NE.0.0E0)THEN       ! Prevent division by zero
          diff  = ABS( TEM(i)-PRETNOD(i) )/PRETNOD(i)
        ELSEIF(TEM(i).NE.0.0E0)THEN
          diff  = 1.0E0
        ENDIF

        IF( diff.GT.PUTNORM )THEN
          IF( STEADYSTATE.AND.converged )THEN
            WRITE(*,2)i,diff*100.0,TEM(i)
 2           FORMAT('   Node',I3,'   Change ',F6.2,
   1             '%    Temperature',F9.2)
          ENDIF
          converged = FALSE
        ENDIF

      ENDDO
      Convergence = converged
C   ==================================================================
      RETURN
      END
```

subroutine display(ivid,PE,AE,AT,INFIL,Water)

```
c   ==================================================================
c     This subroutine plots the time step, potential evaporation,
c    actual evaporation, temperature, and relative humidity versus
c    the total time to the screen.
c     subroutines called:
c       - initgr: initializes the display and draws the axis.
c       - color: sets the plot color
c       - lines: plots a straight line
c   ==================================================================
      implicit none
      include 'sppcommo.fi'        ! Declares some SPP parameters
      include 'function.fi'
      include 'declare.fi'
c   ----------------------------------------
```

```
      character fname*10
      character gname*10
      integer  ivid
      real     tta(4),da(4),pa(4),ea(4),ra(4),ia(4),ta(4),wa(4),sa(4)
      save     tta,da,pa,ea,ra,ia,ta,wa,sa
c     -------------------------------------------------
      DOUBLE PRECISION AE              ! Actual Evaporative Flux
      DOUBLE PRECISION AT              ! Actual Transpiratory Flux
      REAL        INFIL        ! Rainfall minus runoff
      DOUBLE PRECISION PE              ! Potential Evaporation
      real        Water          ! The current water balance
c     ===========================================================
      if(ttime.gt.DELTAT)then
        tta(1)  = tta(2)    ! This is the total time array
        da (1)  = da (2)    ! This is the time step array
        sa (1)  = sa (2)    ! This is the surface suction array
        pa (1)  = pa (2)    ! This is the potential evap. array
        ea (1)  = ea (2)    ! This is the actual evap. array
        ra (1)  = ra (2)    ! This is the actual transp. array
        ia (1)  = ia (2)    ! This is the rainfall minus runoff array
        ta (1)  = ta (2)    ! This is the surface temperature array
        wa (1)  = wa (2)    ! This is the water balance
      else
       call init_graph(ivid,tta,da,pa,ea,ra,ia,ta,wa,sa)
       tta(1) = 0.0
       da (1) = DELTAT        ! seconds
       if( SUCNOD(1).GT.0.0 )then
         sa(1) = SUCNOD(1)     ! kPa
       else                 ! Can't graph negative values on a log scale
         sa(1) = 0.1          ! kPa
       endif
       pa(1)  = PE    * 86400.0      ! Convert to mm/day
       ea(1)  = AE    * 86400.0      ! Convert to mm/day
       ra(1)  = AT    * 86400.0      ! Convert to mm/day
       ia(1)  = INFIL * 86400.0      ! Convert to mm/day
       ta(1)  = TEM(1)
       wa(1)  = 0.0
      endif
      tta(2) = TTIME/3600.0          ! (hrs)    Total time
      da (2) = DELTAT                ! (s)      Time step
      if( SUCNOD(1).GT.0.1 )then
        sa(2) = SUCNOD(1)            ! (kPa)    Surface Suction
      else  ! Can't graph negative values on a log scale
        sa(2) = 0.1                  ! (kPa)    Surface Suction
      endif
      pa(2)  = PE    * 86400.0       ! (mm/day)  Potential Evaporation
      ea(2)  = AE    * 86400.0       ! (mm/day)  Actual Evaporation
      ra(2)  = AT    * 86400.0       ! (mm/day)  Actual Transpiration
      ia(2)  = INFIL * 86400.0       ! (mm/day)  Rainfall minus Runoff
      ta(2)  = TEM(1)                ! (Celcius) Surface Temperature
      wa(2)  = Water                 ! (mm/day)  Water Balance
c     -------------------------------------------------
      call color(15)
      call origin(0,+6.,0)
      call lgline(tta,da,2,1,1,32,+1,.1)  ! Deltat vs TTIME
      call origin(0,-6.,0)
      call color(14)
      call lines(tta,ta,2,1,1,32,.1)      ! Surface Temperature vs TTIME
      call color(12)
      call lgline(tta,sa,2,1,1,32,+1,.1)  ! Surface Suction vs Temperature
      call color(4)
      call lines(tta,wa,2,1,1,32,.1)      ! WaterBalance vs TTIME
      IF(VEGETATION)THEN
        call color(9)
        call lines(tta,ra,2,1,1,32,.1)    ! AT vs TTIME
      ENDIF
      call color(13)
      call lines(tta,ia,2,1,1,32,.1)      ! Rainfall minus Runoff vs TTIME
      call color(10)
      call lines(tta,pa,2,1,1,32,.1)      ! PE vs TTIME
      call color(11)
```

```
      call lines(tta,ea,2,1,1,32,.1)     ! AE vs TTIME
c   ====================================================================
      return
      end
```

SUBROUTINE ELEM1Q1(Element)

```
C   ====================================================================
C     This subroutine computes the element stiffness and storage
C    matrices.
C   ====================================================================
      IMPLICIT NONE                  ! Ensure that all variables have been correctly defined
      INCLUDE 'FUNCTION.FT'            ! All function defined here
      INCLUDE 'DECLARE.FT'             ! All include files in here
C   ====================================================================
      REAL   bt   (MAX_GAUSS ,MAX_GAUSS)   ! The transpose of btemp
      REAL   btbwk (MAX_PNODES,MAX_PNODES)  ! Element mass storage matrix @ t + dt
      REAL   btemp (MAX_GAUSS ,MAX_GAUSS)  ! The gradient matrix
      REAL   dsl1  (MAX_GAUSS)            ! Used to form gradient matrix
      REAL   dsl2  (MAX_GAUSS)            ! Used to form gradient matrix
      REAL   dsl3  (MAX_GAUSS)            ! Used to form gradient matrix
      REAL   dsf   (MAX_GAUSS ,MAX_GAUSS)  ! Used to form gradient matrix
      INTEGER Element                   ! Current Element Number
      INTEGER i,j,k,l,m               ! Loop Counters
      REAL   sf   (MAX_GAUSS ,MAX_GAUSS)   ! Shape Function Matrix
      REAL   sl1  (MAX_GAUSS)            ! Shape at First Node of Element
      REAL   sl2  (MAX_GAUSS)            ! Shape at Second Node of Element
      REAL   sl3  (MAX_GAUSS)            ! Shape at Third Node of Element
      REAL   sft  (MAX_GAUSS ,MAX_GAUSS)   ! The transpose of the Shape Function Matrix
      REAL   t                       ! Temporary Variable
      REAL   u1                       ! Coordinate of First node of Element
      REAL   u2                       ! Coordinate of Second node of Element
      REAL   u3                       ! Coordinate of Third node of Element
C   ====================================================================
c    Note:  Max_Guass must be greater than or equal to Max_Pnodes
C   ====================================================================
      IF(PNODES.EQ.2)THEN
       DO i=1,AGAUSS
        sl1 (i) =  0.5*(1.0-AX(i))
        sl2 (i) =  0.5*(1.0+AX(i))
        dsl1(i) = -0.5
        dsl2(i) =  0.5
       ENDDO
      ELSE  ! PNODES.EQ.3
       DO i=1,AGAUSS
        sl1 (i) =  0.5*(AX(i)*(AX(i)-1.0))
        sl2 (i) = -1.0*(AX(i)+1.0)*(AX(I)-1.0)
        sl3 (i) =  0.5*(AX(i)*(AX(i)+1.0))
        dsl1(i) =  0.5*(2.0*AX(i)-1.0)
        dsl2(i) = -2.0*AX(i)
        dsl3(i) =  0.5*(2.0*AX(i)+1.0)
       ENDDO
      ENDIF
C   ====================================================================
C     INITIALIZING THE ELEMENT MATRICES
C   ====================================================================
      DO j=1,PNODES
       DO i=1,PNODES
         STSW  (i,j) = 0.0  ! Mass moisture storage matrix
          STSH  (i,j) = 0.0  ! Mass heat storage matrix
         BTBW  (i,j) = 0.0  ! Suction stiffness matrix
         btbwk (i,j) = 0.0  ! Load related to gravity @ t + dt
         BTBH  (i,j) = 0.0  ! Temperature stiffness matrix
         BTBWH (i,j) = 0.0  ! Suction stiffness matrix associated with Temperature coupling
         BTBHW (i,j) = 0.0  ! Temperature stiffness matrix associated with Suction coupling
       ENDDO
      ENDDO
C   ====================================================================
C     STARTING CALCUATIONS IN THE GAUSS LOOP
```

```
C  ==========================================================================
    k = 1
    DO m = 1,AGAUSS
      IF(PNODES.EQ.2) THEN
        u1 = YCORD( NELCON(1,Element) )/100.0   ! Getting coordinate of First node of Element
        u2 = YCORD( NELCON(2,Element) )/100.0   ! Getting coordinate of Second node of Element
        DETJ(m) = ABS( dsl1(m)*u1 + dsl2(m)*u2 )
      ELSEIF(PNODES.EQ.3) THEN
        u1 = YCORD( NELCON(1,Element) )/100.0   ! Getting coordinate of First node of Element
        u2 = YCORD( NELCON(2,Element) )/100.0   ! Getting coordinate of Second node of Element
        u3 = YCORD( NELCON(3,Element) )/100.0   ! Getting coordinate of Third node of Element
        DETJ(m) = ABS( dsl1(m)*u1 + dsl2(M)*u2 + dsl3(M)*u3 )
      ENDIF
      DINVJ(m) = 1.0/DETJ(m)
C     ======================================================================
C     FORMING THE sf AND dsf MATRICES
C     ======================================================================
      IF(PNODES.EQ.2) THEN
        sf (m,1) = sl1 (m)
        sf (m,2) = sl2 (m)
        dsf(m,1) = dsl1(m)
        dsf(m,2) = dsl2(m)
      ELSEIF(PNODES.EQ.3) THEN
        sf (m,1) = sl1 (m)
        sf (m,2) = sl2 (m)
        sf (m,3) = sl3 (m)
        dsf(m,1) = dsl1(m)
        dsf(m,2) = dsl2(m)
        dsf(m,3) = dsl3(m)
      ENDIF
C     ======================================================================
C     FORMING THE GRADIENT MATRIX
C     ======================================================================
      DO i = 1,PNODES
        btemp(m,i) = dsf(m,i)*DINVJ(m)
      ENDDO
C     ======================================================================
C     FORMING THE TRANSPOSE OF sf,dsf, AND B MATRICES
C     ======================================================================
      DO i=1,PNODES
        sft (i,m) = sf  (m,i)*DETJ (m)
        bt  (i,m) = btemp(m,i)
      ENDDO
C     ======================================================================
C     FORMING THE PRODUCT MATRICES MULTIPLYING BY GAUSS WTS AND
C     SUMMING TO OBTAIN THE INTEGRATED BTB AND STS MATRICES
C     ======================================================================
      DO i=1,PNODES
        DO j=1,PNODES
          t = sft(i,k)*sf(k,j)*AW(m)
          STSW (i,j) = t*CWMASS  (m) + STSW (i,j)
          STSH (i,j) = t*CHMASS  (m) + STSH (i,j)
          t = bt(i,k)*btemp(k,j)*AW(m)*DETJ(m)
          BTBW  (i,j) = t*CWSTIFF (m) + BTBW  (i,j)
          btbwk (i,j) = t*CWK     (m) + btbwk (i,j)
          BTBWH (i,j) = t*CWHSTIFF(m) + BTBWH (i,j)
          BTBH  (i,j) = t*CHSTIFF (m) + BTBH  (i,j)
          BTBHW (i,j) = t*CHWSTIFF(m) + BTBHW (i,j)
        ENDDO
      ENDDO
      k = k + 1
C  ==========================================================================
 300 ENDDO ! End of do m = 1,AGUASS
C  ==========================================================================
C   Forming the element load vector related to gravity
C  ==========================================================================
    DO l = 1,PNODES
      DSTK(l) = 0.0
      DO m = 1,PNODES
        DSTK (l) = DSTK(l)+btbwk (l,m)*YCORD(NELCON(m,Element))/100.
      ENDDO
```

```
      ENDDO
C   ================================================================
      RETURN
      END
```

SUBROUTINE FLUX

```
C   ================================================================
C     This subroutine calculates the surface flux, the potential
C    evaporation, the root evapotranspiration, and the soil surface
C    temperature.  Also, the surface temperature boundary condition
C    is updated.
C   ================================================================
      IMPLICIT NONE               ! Ensure that all variables have been correctly defined
      INCLUDE 'FUNCTION.FT'        ! Contains all function declarations
      INCLUDE 'DECLARE.FT'         ! Contains all common block declarations
C   ================================================================
      INTEGER i                  ! Loop counter
      REAL limitFactor           ! (dec.)  Plant Limiting Factor
      REAL pv1                   ! (kPa)   Vapour Pressure of the Soil Surface
      REAL rh1                   ! (n/a)   Relative Humidity at Top Node
      REAL satvp0                ! (kPa)   Saturated Vapour Pressure of the Evaporating Pan Surface
      REAL satvp1                ! (kPa)   Saturated Vapour Pressure of the Soil Surface
      REAL water_temp            ! (C)     Temperature of the Water for Pan Evaporation
      Equivalence (water_temp,DURATION(2)) ! Shared Storage
C   ================================================================
c   ----------------------------------------------
C     CALCULATE THE ACTUAL EVAPORATION (AE) and update temperature
c   ----------------------------------------------
      rh1 = Calc_RH(SUCNOD(1),TEM(1)+273.0)      ! Calculate Relative Humidity at the Top Node
      IF(DFLUX)THEN
        satvp0  = Calc_SatVp(water_temp+273.0)*0.10  ! SatVapPress at Pan Water temperature
        satvp1  = Calc_SatVp(TEM(1)+273.0)*0.10   ! Saturated Vapour Pressure of the soil at the Top Node
        pv1     = satvp1*rh1               ! Vapour Pressure of the soil at the Top Node
        VFLUXPE = Calc_DfluxPE(satvp0,satvp1)     ! PE calculation
        PENMAN  = Calc_Dflux (pv1)          ! Vertical Vapour flux at top node
      ELSEIF(TRANSIENT)THEN
        VFLUXPE = Calc_VfluxPE()            ! PE calculation
        PENMAN  = Calc_Vflux(rh1)           ! Vertical Vapour flux at top node
      ELSE
        VFLUXPE = 0.0D0
        PENMAN  = 0.0D0
      ENDIF
      IF(CALCULATE_TEMPS)THEN
        TEM(1)    = calc_SoilTemp(WIND(2))
        EBH (1,2) = TEM(1)
      else
        if(nday.lt.days)then
        tem(1)=ebh(1,3) + (nexttoptemp-ebh(1,3))*ttime/86400  ! linearly ramps user specified surface temps from one day to next
        ebh(1,2) = tem(1)
        endif
      ENDIF

c ----------------------------------------------
C       CALCULATE THE Plant Root UPTAKE PROFILE
c   ----------------------------------------------
      IF(VEGETATION.AND.(LAI.GE.0.1))THEN
        VFLUXPT = Calc_VFLUXPT()        ! (mm/sec) CALCULATES POT. TRANSP
        DO i=RootTop(2),RootDepth(2)
          PRU(i) = Calc_PRU(i)        ! PRU calculation for node i
          CALL ActRtUpt(i)           ! ARU calculation for node i
        ENDDO
      ENDIF
c     if(nday.eq.1)then
c     if(ttime.le.7200) then
c     tem(1) = 0-(-0.+7)/7200*ttime
c     ebh(1,2) = tem(1)
c     elseif(ttime.gt.7200 .and. ttime.le.14400)then
c     tem(1) = -7-(-7+10)/(14400-7200.)*(ttime-7200)      ! hourly surface temp. algorithm for Jame (1980) modelling.
```

```
c   ebh(1,2) = tem(1)
c   endif

c   if(ttime.le.1800) then
c   tem(1) = 0-(-0.+3.8)/1800*ttime
c   ebh(1,2) = tem(1)
c   elseif(ttime.gt.1800 .and. ttime.le.14400)then
c   tem(1) = -3.8-(-3.8+5.2)/(14400-1800.)*(ttime-1800)
c   ebh(1,2) = tem(1)
c   endif
c   endif
C =====================================================================
    RETURN
    END
```

```
C =====================================================================
C   This subroutine reads gauss point weights and locations.
C =====================================================================
     SUBROUTINE GAUSS_DATA
C =====================================================================
    IMPLICIT NONE                    ! Ensure that all variables have been correctly defined
    INCLUDE 'FUNCTION.FT'              ! All function defined here
    INCLUDE 'DECLARE.FT'              ! All include files in here
C =====================================================================
    INTEGER    i                ! Loop Counter
    INTEGER    j                ! Loop Counter
C =====================================================================

    OPEN (UNIT=23,FILE=GaussLcFile,STATUS='OLD') ! Gauss Pt. Locations
    OPEN (UNIT=24,FILE=GaussWtFile,STATUS='OLD') ! Gauss Pt. Weights
    DO i=1,AGAUSS                    ! Read Locations & Weights
     DO j=1,i
      read(23,*) AX(j)
      read(24,*) AW(j)
     ENDDO
     DO J=i+1,MAX_GAUSS                  ! Initialize Rest of Array to zero
      AX(J) = 0.0
      AW(J) = 0.0
     ENDDO
    ENDDO
    CLOSE (UNIT=23)
    CLOSE (UNIT=24)
C ---------------------------------------
    RETURN
    END
```

```
SUBROUTINE Get_Run_Time
C =====================================================================
C     This subroutine gets the solve parameters from the user.
C   ie: input date file, output data file,spline graph options, GRAPHICS
C   option, and the number of days to run.
C     subroutines called:
C        main_in: reads all the input data from the supplied data file
C =====================================================================
    IMPLICIT NONE
    INCLUDE 'FUNCTION.FT'              ! Contains all function declarations
    INCLUDE 'DECLARE.FT'              ! Contains all common block declarations
C ---------------------------------------
    LOGICAL     Debug_Splines       ! Flag to indicate splines are to be graphed to screen
    CHARACTER   Char              ! Gets debug info
    LOGICAL     FILE_FOUND          ! Flag to indicate if file has been found
    INTEGER*4   IARGC              ! Returns number of args on command line
    CHARACTER*30 InFile              ! Name of the main input file
    CHARACTER*4 Numb_Days            ! Number of Days Argument
```

```
C ==========================================================
C         Get Name of Main Input File
C ==========================================================
      IF(IARGC().GT.1)THEN
        CALL GETARG(2,InFile)
      ELSE
1       WRITE (*,2)
2       FORMAT(' ','Input the name of the Input Data File = ',$)
        READ(*,3,ERR=1,END=1) InFile
3       FORMAT (A30)
      ENDIF
      INQUIRE(FILE=InFile,EXIST=FILE_FOUND)
      DOWHILE(.NOT.FILE_FOUND)
        WRITE(*,*) 'File not found. ',InFile
33      WRITE(*,2)
        READ(*,3,ERR=33,END=33)InFile
        INQUIRE(FILE=InFile,EXIST=FILE_FOUND)
      ENDDO
      GRAPHICS = FALSE
      IF (IARGC().GT.2) THEN
        CALL GETARG(3,Char)
        IF(Char.eq.'T'.or.Char.eq.'t')THEN
          GRAPHICS = TRUE
        ENDIF
      ELSE
332     WRITE(*,333)
333     FORMAT(' ',
     1  'Show Screen Graphics? [T/F] ',$)
        READ(*,334,ERR=332)GRAPHICS
334     FORMAT(L1)
      ENDIF
      Debug_Splines = FALSE
      IF (IARGC().GT.3) THEN
        CALL GETARG(4,Char)
        IF(Char.eq.'T'.or.Char.eq.'t')THEN
          Debug_Splines = TRUE
        ENDIF
      ELSE
335     WRITE(*,336)
336     FORMAT(' ',
     1  'Graph splines to screen?(Suct vs VolWC, etc.) [T/F] ',$)
        READ(*,337,ERR=335)Debug_Splines
337     FORMAT(L1)
      ENDIF
      DETAILED = FALSE
      IF (IARGC().GT.4) THEN
        CALL GETARG(5,Char)
        IF(Char.eq.'T'.or.Char.eq.'t')THEN
          DETAILED = TRUE
        ENDIF
      ELSE
338     WRITE(*,339)
339     FORMAT(' ',
     1  'Write Detailed output data (eg: DV,HydCond,etc) [T/F] ',$)
        READ(*,340,ERR=338)DETAILED
340     FORMAT(L1)
      ENDIF
C ----------------------------------------

C ==========================================================
C       Call subroutine to read all data from the input file
C ==========================================================
      CALL MAIN_IN(InFile,Debug_Splines)
C ==========================================================

C ==========================================================
C       Get number of simulation days to run program
C ==========================================================
      NDAY = DAYS                    ! Save total specified number days
      DAYS = DAYS + 1
```

```
      IF( TRANSIENT )THEN  ! This data is only required for transient solutions
        IF(IARGC().GT.5)THEN
          CALL GETARG(6,Numb_Days)
          READ(Numb_Days,'(I4)')DAYS
          if(days.eq.1) then
            WRITE(*,*) ' 1 day specified to run.'
          else
            WRITE(*,*) DAYS,' days specified to run.'
          endif
        ENDIF
        WRITE (*,*) NDAY,' days of data in data file.'
        IF(DAYS.LT.0 .OR. DAYS.GT.NDAY)THEN
44          WRITE (*,45)
45          FORMAT( ' ','INPUT ACTUAL DAYS OF SIMULATION = ',$)
          READ  (*,*,ERR=44,END=44) DAYS
        ENDIF
      ELSE
        NDAY = 0
        DAYS = 1
      ENDIF
C   ================================================================
c     End of Get_Run_Time
C   ================================================================
      RETURN
      END
```

subroutine graph(soil,np,xa,ya,za,yas,zas,X_Label,Y_Label,type)

```
c   ================================================================
c     This subroutine takes spline data and generates points from the
c     spline.  The splines is then printed to the screen.
c   ================================================================
      include 'sppcommo.fi'      ! Declares some SPP parameters
      include 'constant.fi'      ! declares array const.
c   ----------------------------------------
      integer  points
      parameter (points = 1000)
c   ----------------------------------------
      character X_Label*(*)         ! X Axis Label
      character Y_Label*(*)         ! Y Axis Label
      character fname*10
      integer i,j                 ! loop counters
      integer soil                ! Current layer
      integer np(max_types)          ! Number of points per layer
      integer type               ! Type of Plot Required
      real x(points+2)             ! X Coordinate for output points from spline data
      real xa(max_points,max_types)   ! X coordinates of spline data
      real y(points+2)             ! Y Coordinate for output points from spline data
      real ya(max_points,max_types)   ! Y coordinates of spline data
      real yas(max_points,max_types)  ! Smoothed Y Coordinate for output points from spline data
      real ys(points+2)            ! Y Coordinate for output points from spline data
      real z(points+2)             ! Slope of spline at different points
      real za(max_points,max_types)   ! Curvative of spline data
      real zas(max_points,max_types)  ! Smoothed Slope of spline at different points
      real zs(points+2)            ! Slope of spline at different points
      real    Fn_Point             ! Fortran spline function which returns a spline value
      real    Fn_Slope             ! Fortran spline function which returns the slope of a spline
c   ================================================================
      mon  =  16      ! MCA Graphics Display
      ifore =  15     ! White Foreground Color
      iback =  16     ! Blue  Background Color
      nprin =   8
      mode =   5
      isave =  -1
      fname='memory'
      call vsinit(mon,10.,8.,isave,fname,iunitv,ivid,ifore,iback,iunitm)
      call linwid(0,.01)
      call setasp(1.0)
```

```
c   ============================================================
c             Generate points to plot
c   ============================================================
      n = np(soil)
      do j=1,points
c       x(j) = exp( log( xa(1,soil) ) ! This one is for the new splines
c  +          + (j-1)*( log(xa(n,soil))-log(xa(1,soil)) )/points )
        x(j) = exp( xa(1,soil) + (j-1)*(xa(n,soil)-xa(1,soil))/points )
        y(j) = Fn_Point(soil,np,xa,ya,za,x(j))
        ys(j)= Fn_Point(soil,np,xa,yas,zas,x(j))
        z(j) = Fn_Slope(soil,np,xa,ya,za,x(j))
        zs(j)= Fn_Slope(soil,np,xa,yas,zas,x(j))
      enddo
c   ****************************************************************
c   ****************************************************************
      if(type.eq.linear)then
c     ==========================================================
c               LINEAR VS. LINEAR Graph
c     ==========================================================
        call color(15)
        call scale(x,6.,points,1)
        call scale(ys,8.,points,1)
        call axis (1.,1.,X_Label,0,-1,1,6.01,0.,x(points+1),
     *          x(points+2),.1,2)
        call axis (1.,1.,Y_Label,0,1,-1,8.01,90.,ys(points+1),
     *          ys(points+2),.1,1)
        y(points+1) = ys(points+1)
        y(points+2) = ys(points+2)
        call origin(1.,1.,0)
        call color(11)
        call lines(x,y,points,1,1,32,.1)
        call color(15)
        call lines(x,ys,points,1,1,32,.1)
        call origin(-1.,-1.,0)
c     ----------------------------------------
        call color(14)
        call scale(zs,8.,points,1)
        call axis (7.,1.,'Slope',0,-1,1,8.01,90.,zs(points+1),
     *          zs(points+2),.1,1)
        z(points+1) = zs(points+1)
        z(points+2) = zs(points+2)
        call origin(1.,1.,0)
        call color(11)
        call lines(x,z,points,1,1,32,.1)
        call color(14)
        call lines(x,zs,points,1,1,32,.1)
        call origin(-1.,-1.,0)
c     ==========================================================
c      Plot user supplied data points
c     ==========================================================
        call color(11)
        do i=1,n
          x(i) = exp(xa(i,soil))
          y(i) = exp(ya(i,soil))
        enddo
        x(n+1)=x(points+1)
        x(n+2)=x(points+2)
        y(n+1)=y(points+1)
        y(n+2)=y(points+2)
        call origin(1.,1.,0)
        call lines(x,y,n,1,-1,ichar('O'),.1)
        call origin(-1.,-1.,0)
c   ****************************************************************
c   ****************************************************************
      else if(type.eq.semi_log)then
c     ==========================================================
c               LOG VS. LINEAR GRAPH
c     ==========================================================
        call color(15)
        call lgscal(x,6.,points,1)
        call scale (ys,8.,points,1)
```

```
      call lgaxis(1.,1.,X_Label,0,-1,1,6.01,0.,x(points+1),
     *            x(points+2),.1)
      call axis  (1.,1.,Y_Label,0,1,-1,8.01,90.,ys(points+1),
     *            ys(points+2),.1,3)
      y(points+1) = ys(points+1)
      y(points+2) = ys(points+2)
      call origin(1.,1.,0)
      call color(11)
      call lgline(x,y,points,1,points/10,32,-1,.1)
      call color(15)
      call lgline(x,ys,points,1,points/10,32,-1,.1)
      call origin(-1.,-1.,0)
c     ------------------------------------------
      call color(14)
      call scale (zs,8.,points,1)
      call axis  (7.,1.,'Slope',0,-1,1,8.01,90.,zs(points+1),
     *            zs(points+2),.1,3)
      z(points+1) = zs(points+1)
      z(points+2) = zs(points+2)
      call origin(1.,1.,0)
      call color(11)
      call lgline(x,z,points,1,points/10,32,-1,.1)
      call color(14)
      call lgline(x,zs,points,1,points/10,32,-1,.1)
      call origin(-1.,-1.,0)
c     ==============================================================
c       Plot user supplied data points
c     ==============================================================
      call color(11)
      do i=1,n
       x(i) = exp(xa(i,soil))
       y(i) = exp(ya(i,soil))
      enddo
      x(n+1)=x(points+1)
      x(n+2)=x(points+2)
      y(n+1)=y(points+1)
      y(n+2)=y(points+2)
      call origin(1.,1.,0)
      call lgline(x,y,n,1,-1,ichar('O'),-1,.1)
      call origin(-1.,-1.,0)
c    ***************************************************************
c    ***************************************************************
     else if(type.eq.logarithmic)then
c     ==============================================================
c                  LOG VS. LOG GRAPH
c     ==============================================================
      call color(15)
      call lgscal(x,6.,points,1)
      call lgscal(ys,8.,points,1)
      call lgaxis(1.,1.,X_Label,0,-1,1,6.01,0.,x(points+1),
     *            x(points+2),.1)
      call lgaxis(1.,1.,Y_Label,0,1,-1,8.01,90.,ys(points+1),
     *            ys(points+2),.1)
      call origin(1.,1.,0)
      y(points+1) = ys(points+1)
      y(points+2) = ys(points+2)
      call color(11)
      call lgline(x,y,points,1,points/10,32,0,.1)
      call color(15)
      call lgline(x,ys,points,1,points/10,32,0,.1)
      call origin(-1.,-1.,0)
c     ------------------------------------------
      call color(14)
      call scale (zs,8.,points,1)
      call axis  (7.,1.,'Slope',0,-1,1,8.01,90.,zs(points+1),
     *            zs(points+2),.1,3)
      z(points+1) = zs(points+1)
      z(points+2) = zs(points+2)
      call origin(1.,1.,0)
      call color(11)
      call lgline(x,z,points,1,points/10,32,-1,.1)
```

```
        call color(14)
        call lgline(x,zs,points,1,points/10,32,-1,.1)
        call origin(-1.,-1.,0)
c      ==================================================================
c       Plot user supplied data points
c      ==================================================================
        call color(11)
        do i=1,n
         x(i) = exp(xa(i,soil))
         y(i) = exp(ya(i,soil))
        enddo
        x(n+1)=x(points+1)
        x(n+2)=x(points+2)
        y(n+1)=y(points+1)
        y(n+2)=y(points+2)
        call origin(1.,1.,0)
        call lgline(x,y,n,1,-1,ichar('O'),0,.1)
        call origin(-1.,-1.,0)
c     ****************************************************************
c     ****************************************************************
      endif

      close(iunitv) ! Close the screen file
c     ==================================================================
c           Get User to Press a Key when finished
c     ==================================================================
      call msg(0.,.1,.2,'Press a Key to Continue.',0.,0,1)
      ans = getc()
c     ==================================================================
c             Reset video to original state
c     ==================================================================
      call gmode(ivid)
c     ==================================================================
      return
      end
```

```
C     ==================================================================
      SUBROUTINE DICE(oldtemp,newtemp,oldsuc,newsuc)
C     ==================================================================
C
C     ==================================================================
C     ==================================================================
      IMPLICIT NONE                   ! Ensure that all variables have been correctly defined
      INCLUDE 'FUNCTION.FI'           ! Contains all function declarations
      INCLUDE 'DECLARE.FI'            ! Contains all common block declarations
C     ==================================================================
      integer i,type                  ! loop counter, soil type
      real volwc
      real crittemp                   ! freezing point depression
      real deltemp                    ! change in temp. over last time step
      real m2i,m2w                    ! slope of soil freezing / soil water characteristic curves
      real avetemp,avesuc             ! average temp. / suction over last time step
      real newtemp(max_nodes),newsuc(max_nodes) ! new temp. / suction nodal arrays
      real oldtemp(max_nodes),oldsuc(max_nodes)    ! old temp. / suction nodal arrays
      real delwat                     ! liquid flux at a given node (dec.)
      real asuc1,asuc2                ! average mid-nodal suctions
      real wflux1,wflux2              ! average mid-nodal liquid flux
      real ,head1,head2,head3,avk1,avk2 ! last node, current node, next node total head; aveage mid nodal hyd. cond.
      real vflux1,vflux2              ! average mid-nodal vapour flux
      real pv1,pv2,pv3,dv1,dv2        ! last node, current node, next node vapour pressure; average mid nodal
                                      ! dv terms
      real vwat1,vwat2                ! average mid nodal volumetric water contents
      real osuc1,osuc2,osuc3, otem1,otem2,otem3 ! last, current, next node old temps and suctions
      real  atem1,atem2,ice1,ice2     ! average nodal new temps and ice contents
      real,oice1,oice2,oice3,rh1,rh2,rh3 ! last node, current node, next node old ice contents and relative humidity

      do i=1,nnodes
```

```
C  ==========================================================================
C   Calculate the critical temperature for freezing
C  ==========================================================================
   type=soiltype(i)

   volwc=calc_volwc(type,oldsuc(i)) ! based on start of time step suction
   crittemp=-fn_point(type,points7,xvoluwc,xtem,splinsl7,volwc)
   if(nodvolice(i).gt.0.) crittemp=oldtemp(i)
C  ==========================================================================
C   Calculate average temp. and if freezing or thawing occured
C  ==========================================================================
   if(newtemp(i).le.crittemp.and.oldtemp(i).lt.-0.05) then ! freezing
       avetemp=(newtemp(i)+crittemp)/2.
       deltemp=newtemp(i)-crittemp
   elseif(newtemp(i).gt.crittemp.and.newtemp(i).lt.-0.05        ! thawing.  Note:  -0.05 °C chosen to
                                                                ! prevent errors using SFC near 0°C.
  1      .and.nodvolice(i).gt.0.)then
       avetemp=(newtemp(i)+crittemp)/2.
       deltemp=newtemp(i)-crittemp
   elseif(newtemp(i).ge.-0.05.and.nodvolice(i).gt.0.)then  ! thawing
       avetemp=(-0.05+crittemp)/2.
       if(avetemp.gt.0) avetemp=-avetemp
       deltemp=abs(-0.05-crittemp)
   else
     avetemp=99.  ! no freezing or thawing happening at this node
   endif


C  ==========================================================================
C   Calculate liquid flux over previous time step
C  ==========================================================================
   if(avetemp.ne.99) then

     if(i.eq.1) then
       osuc1=oldsuc(i)
       otem1=oldtemp(i)+273.16
       oice1=nodvolice(i)
     else
       osuc1=oldsuc(i-1)
       otem1=oldtemp(i-1)+273.16
       oice1=nodvolice(i-1)
     endif

     osuc2=oldsuc(i)
     otem2=oldtemp(i)+273.16
     oice2=nodvolice(i)

     if(i.eq.nnodes)then
       osuc3=oldsuc(i)
       otem3=oldtemp(i)+273.16
       oice3=nodvolice(i)
     else
       osuc3=oldsuc(i+1)
       otem3=oldtemp(i+1)+273.16
       oice3=nodvolice(i+1)
     endif

     asuc1=(osuc1+osuc2)/2.
     asuc2=(osuc2+osuc3)/2.
     atem1=(otem1+otem2)/2.
     atem2=(otem2+otem3)/2.
     ice1=(oice1+oice2)/2.
     ice2=(oice2+oice3)/2.

     avk1=calc_k(type,asuc1,ice1)
     avk2=calc_k(type,asuc2,ice2)
     dv1=calc_vapour_diff(atem1,vwat1,ice1,pors(type))
     dv2=calc_vapour_diff(atem2,vwat2,ice2,pors(type))


     if(i.eq.nnodes) then
```

```
      head1=ycord(i-1)/100.-osuc1/grav
      head2=ycord(i)/100.-osuc2/grav
      rh1=calc_rh(oldsuc(i-1),oldtemp(i-1)+273.16)
      rh2=calc_rh(oldsuc(i),oldtemp(i)+273.16)
      pv1=calc_satvp(oldtemp(i-1)+273.16)*rh1*0.1
      pv2=calc_satvp(oldtemp(i)+273.16)*rh2*0.1

      wflux2=avk2*(head2-head1)/(ycord(i)-ycord(i-1))*100.
      vflux2=dv2*(pv2-pv1)/(ycord(i)-ycord(i-1))*100
      delwat=(wflux2+vflux2)/(ycord(nnodes)-ycord(nnodes-1))*100.


     elseif(i.eq.1) then

      head1=ycord(i)/100.-osuc2/grav
      head2=ycord(i+1)/100.-osuc3/grav
      rh1=calc_rh(oldsuc(i),oldtemp(i)+273.16)
      rh2=calc_rh(oldsuc(i+1),oldtemp(i+1)+273.16)
      pv1=calc_satvp(oldtemp(i)+273.16)*rh1*0.1
      pv2=calc_satvp(oldtemp(i+1)+273.16)*rh2*0.1

      wflux1=-avk1*(head2-head1)/(ycord(i)-ycord(i+1))*100.
      vflux1=-dv1*(pv2-pv1)/(ycord(i)-ycord(i+1))*100
      delwat=(wflux1+vflux1)/(ycord(2)-ycord(1))*100.

     else

      head1=ycord(i-1)/100.-osuc1/grav
      head3=ycord(i+1)/100.-osuc3/grav
      head2=ycord(i)/100.-osuc2/grav
      rh1=calc_rh(oldsuc(i-1),oldtemp(i-1)+273.16)
      rh2=calc_rh(oldsuc(i),oldtemp(i)+273.16)
      rh3=calc_rh(oldsuc(i+1),oldtemp(i+1)+273.16)
      pv1=calc_satvp(oldtemp(i-1)+273.16)*rh1*0.1
      pv2=calc_satvp(oldtemp(i)+273.16)*rh2*0.1
      pv3=calc_satvp(oldtemp(i+1)+273.16)*rh3*0.1
      wflux1=avk1*(head2-head1)/(ycord(i)-ycord(i-1))*100.
      wflux2=avk2*(head3-head2)/(ycord(i+1)-ycord(i))*100.
      vflux1=dv1*(pv2-pv1)/(ycord(i)-ycord(i-1))*100
      vflux2=dv2*(pv3-pv2)/(ycord(i+1)-ycord(i))*100
      delwat=((wflux2-wflux1)+(vflux2-vflux1))  !m/sec
   1        /(ycord(i+1)-ycord(i-1))*200.  !m/m /sec


     endif

C    ====================================================================
C    Calculate change in ice content: d(ice)=d(total_water)-d(unfrozen)
C    ====================================================================

    m2i=-fn_slope(type,points8,ytem,yvoluwc,splinsl8,abs(avetemp))

      delice(i)=(delwat*deltat-m2i*deltemp)/rhoice

    else
      delice(i)=0.
    endif !avetemp.ne.99

    if(nodvolice(i)+delice(i).le.0.) then  ! make sure node ice content  not negative
     delice(i)=-nodvolice(i)
    endif
    if(newtemp(i).ge.-0.05) then   ! make sure no ice above freezing
      delice(i)=-nodvolice(i)
    endif

    enddo

    return
    end
```

subroutine init_graph(ivid,tta,da,pa,ea,ra,ia,ta,wa,sa)

```
c  ==========================================================================
c    This routine initializes the display for the display subroutine
c  ==========================================================================
     include 'SPPCOMMO.fi'            ! Declares some SPP parameters
     INCLUDE 'FUNCTION.FT'
     INCLUDE 'DECLARE.FT'              ! Contains all declarations
c  ------------------------------------------------
     character fname*10
     character gname*10
     character message*20
     integer  ivid
     real     tta(4),da(4),pa(4),ea(4),ra(4),ia(4),ta(4),wa(4),sa(4)
c  ==========================================================================
     mon  =  16     ! MCA Graphics Display
     ifore =  15    ! White Foreground Color
     iback =  16    ! Blue  Background Color
     nprin =   8
     mode =   5
     isave =  -1
     fname='memory'
     call vsinit(mon,10.,8.,isave,fname,iunitv,ivid,ifore,iback,iunitm)
     call linwid(0,.01)
     call setasp(1.0)
c  ==========================================================================
c    Initialize Deltat Scales
c  ==========================================================================
     call color(15)
     da(1) = MIN_DELTAT
     da(2) = MAX_DELTAT
     tta(1) = 0.0
     tta(2) = 24.0
     call scale(tta,5.,2,1)
     call lgscal(da,2.,2,1)
     call axis (1.2,1.,'Time (hrs)',10,-1,1,5.21,0.0,
    1         tta(3),tta(4),.1,1)
     call lgaxis(1.2,7.,'Time Step (seconds)',
    1         19,1,-1,2.01,90.,da(3),da(4),.1)
c  ==========================================================================
c    Initialize Suction Scale
c  ==========================================================================
     call color(12)
     sa(1) = 0.1
     sa(2) = 1000000.0
     tta(1) = 0.0
     tta(2) = 24.0
     call lgscal(sa,5.6,2,1)
     call lgaxis(1.2,1.,'Surface Suction (kPa)',
    1         21,1,-1,5.61,90.,sa(3),sa(4),.1)
c  ==========================================================================
c    Initialize PE & AE Scales
c  ==========================================================================
     pa(1) = -14.0
     pa(2) = +22.0
     call color(11)
     call scale(pa,8.,2,1)
     do i = 1,4
       ea(i) = pa(i)
       ra(i) = pa(i)
       ia(i) = pa(i)
     enddo
     call axis (.5,1.,'Surface Flux (mm/day)',
    1         21,1,-1,8.01,90.,pa(3),pa(4),.1,1)
c  ==========================================================================
c    Initialize Temperature Scale
c  ==========================================================================
     ta(1) =-10.0
```

```
      ta(2) = 40.0
      call color(14)
      call scale(ta,8.,2,1)
      call axis (6.5,1.,'Surface Temperature (Celcius)',
     1        29,-1,1,8.01,90.,ta(3),ta(4),.1,1)
c  ==================================================================
c  Initialize Water Balance Scale
c  ==================================================================
      wa(1) = -1.5
      wa(2) = +1.5
      call color(4)
      call scale(wa,8.,2,1)
      call axis (7.2,1.,'Water Balance (mm)',
     1        18,-1,1,8.01,90.,wa(3),wa(4),.1,1)
c  ==================================================================
      call color(15)
      write(message,'(A16,I4)') 'Running Day ',NDAY
      call msg(0.,.1,.2,message,0.,0,1)
      call origin(1.2,1.,0)
c  ==================================================================
      return
      end
```

```
C  ==================================================================
```

SUBROUTINE ITERATE

```
C  ==================================================================
C    This subroutine performs an iterative loop until the solution
C  has converged or the maximum number of iterations has been performed.
C  The element, global, system stiffness and storage storage matrices
C  are developed as well as the system load vectors. The coupled system
C  of simultaneous equations is solved to determine the new nodal suctions
C  and temperatures.
C     Subroutines called:
C          - CNICOL    :solver for a transient analysis
C          - SOLVE     :solver for a steady state analysis
C          - ELEM1Q1   :sets up the elemental stiffness and mass matrices.
C          - FLUX      :calculates the evaporative flux
C          - JSHAPE    :the shape function, used to interpolate properties to gauss pts.
C          - RELAXATION:implements a relaxation scheme for the iterative loop.
C             - DICE :calculates the nodal change in ice content over previous time step
C  ==================================================================
      IMPLICIT NONE
      INCLUDE 'DECLARE.FT'            ! All include files in here
      INCLUDE 'FUNCTION.FT'           ! All functions defined here
      REAL    Calc_gFlux             ! Function to calculate fluxes at element boundaries
C  ==================================================================
      REAL    avesuc                 ! Suction at Gauss Point at the half time step
      REAL    aau                    ! Temporary Storage for Water Contents/Suctions
      LOGICAL converged              ! Logical flag to indicate when system has converged
      REAL    dv                     ! Diffusion Coeffecient of Water Through Soil
      REAL    gbtbh  (MAX_NODES ,MAX_NODES ) ! Global Heat Stiffness Matrix
      REAL    gbtbhw (MAX_NODES ,MAX_NODES ) ! Global Heat Coupled to Moisture Stiffness Matrix
      REAL    gbtbw  (MAX_NODES ,MAX_NODES ) ! Global Moisture Stiffness Matrix
      REAL    gbtbwh (MAX_NODES ,MAX_NODES ) ! Global Moisture Coupled to Heat Stiffness Matrix
      REAL    gcord  (MAX_GAUSS)     ! Gaussian Coordinates
      REAL    glh    (MAX_NODES )    ! Global Heat Load Vector
      REAL    glw    (MAX_NODES )    ! Global Moisture Load Vector @ t + dt
      REAL    gstsh  (MAX_NODES ,MAX_NODES ) ! Global Heat Mass Storage Matrix
      REAL    gstsw  (MAX_NODES ,MAX_NODES ) ! Global Moisture Mass Storage Matrix
      REAL    gtemp  (MAX_GAUSS)     ! (K) Temperature at Gauss Pts
      INTEGER i,j,k,l,m              ! Loop counters
      REAL    lastsuc                ! Suction at last Gauss Point at the half time step
      REAL    lastpv                 ! Vapour Pressure at Gauss Pts.
      INTEGER type                   ! Current Layer
      LOGICAL maxd_out               ! Logical switch set when ITER =MXITER
      INTEGER iteration              ! Current iteration number
      REAL    pv                     ! Vapour Pressure at Gauss Pts.
      REAL    rd1,rd2                ! Multiplier for Isothermal & Thermal Vapour
```

```
      REAL   rh                    ! Relative Humidity at Gauss Pts.
      REAL   rm2w                  ! d(Suction)/d(VolWc) at Gauss Pts.
      REAL   suc0  (MAX_GAUSS)     ! Suction at Gauss Points @ t
      REAL   suc1  (MAX_GAUSS)     ! Suction at Gauss Points @ t+dt
      REAL   slpot                 ! Slope of Sat VP vs Temp Curve at Gauss Pts.
      REAL   volwc                 ! Volumetric Water Content at Gauss Points @ t + dt
      REAL   xkk                   ! Permeability at Gauss Pts. @ t + dt/2
      REAL   xkk1                  ! Permeability at Gauss Pts. @ t + dt
      REAL   xlamda                ! Thermal Conductivity at Gauss Points
      REAL   xsheat                ! Specific Heat at Gauss Points
      REAL   avesuck
      REAL   suca  (MAX_GAUSS)     ! Suction at Gauss Points @ t
      REAL   sucb  (MAX_GAUSS)     ! Suction at Gauss Points @ t+dt

      REAL   goldtemp(max_gauss), tgrad
      REAL   newtemp(max_nodes), temptemp(max_nodes)
      REAL   gnewtemp(max_gauss)
      REAL   tempvolwc             ! Temporary nodal vol. w/c
      REAL   crit_temp,avetemp
      REAL   m2l,GG
      REAL   gdelice(max_gauss),oldsuc(max_nodes),tempsuc(max_nodes)
      REAL   oldtem(max_nodes),gnodevolice(max_gauss)
      REAL   latent, mass_freeze
C     ======================================================================
      open(unit=27,file='test.dat',status='new')
      iteration = -1
      converged = FALSE
      maxd_out  = FALSE

C     ======================================================================
C     ********************************************************************
C     Entering the iteration loop
C     ********************************************************************
C     ======================================================================
      DO WHILE( .NOT.(converged.OR.maxd_out) )
      if(soiltype(1).ne.1)soiltype(1)=1
      iteration = iteration + 1
      IF(iteration.EQ.MXITER)THEN
       maxd_out     = TRUE
       MAXD_OUT_TODAY = MAXD_OUT_TODAY + DELTAT
      ENDIF
      IF( STEADYSTATE.AND.(iteration.gt.1) )THEN
        WRITE(*,2)iteration
 2      FORMAT(' Iteration ',I5,$)
      ENDIF
C     ======================================================================
C     Initializing the global matrices
C     ======================================================================
      DO j = 1,NNODES
       DO i = 1,NNODES
        gbtbw (i,j) = 0.0 ! stiffness matrix associated with suctions
        gbtbwh(i,j) = 0.0 ! stiffness matrix associated with temperature coupling
        gbtbh (i,j) = 0.0 ! stiffness matrix associated with temperatures
        gbtbhw(i,j) = 0.0 ! stiffness matrix associated with suction coupling
        gstsw (i,j) = 0.0 ! mass storage matrix associated with suctions
        gstsh (i,j) = 0.0 ! heat storage matrix associated with temperatures
  ENDDO
       glw (j) = 0.0       ! moisture load vector @ t + dt
       glh (j) = 0.0       ! heat load vector
      ENDDO

C     ======================================================================
C     AVERAGING NODAL TEMPERATURES TO GET AVERAGE PROPERTIES
C     ======================================================================

      do i=1,nnodes
        j=i+nnodes
        OLDTEM(i)=PHIA(j)   ! oldtemp=newtemp on first iteration
        OLDSUC(i)=PHIA(i)
        newtemp(i)=tem(i)       ! new temp is needed for freeze analysis
        tem(i)=(tem(i)+phia(j))/2.   ! tem(i) is now average over dt
```

```
      enddo
       call dice(oldtem,newtemp,oldsuc,sucnod)  ! calculated change in ice content
C    ===========================================================================
C       COMPUTING THE INITIAL PROPERTIES K,LAMBDA,SP HEAT,DV,RM2W
C                    AT GAUSS PTS
C    ===========================================================================
      DO 172 i = 1,NELEM
C     -------------------------------------
       type = SOILTYPE(NELCON(PNODES,i)) ! Locate the Current Layer
       CALL JSHAPE(i,PHIA  ,suc0)       ! Interpolate gaussian suctions @ t
       CALL JSHAPE(i,SUCNOD,suc1)       ! Interpolate gaussian suctions @ t+dt
c       CALL JSHAPE2(i,PHIA  ,suca)      ! Interpolate gaussian suctions @ t
c       CALL JSHAPE2(i,SUCNOD,sucb)      ! Interpolate gaussian suctions @ t+dt
       CALL JSHAPE(i,TEM,gtemp)         ! Interpolate gaussian temperatures @ t+dt
       CALL JSHAPE(i,YCORD,gcord)       ! Interpolate gaussian coordinates (constant)
       CALL JSHAPE(i,oldtem,goldtemp)
       CALL JSHAPE(i,newtemp,gnewtemp)
       call jshape(i,delice,gdelice)
       CALL JSHAPE(i,nodvolice,gnodevolice)

C    ======================================================================
       DO m = 1,AGAUSS

       gnodevolice(m)=gnodevolice(m)+gdelice(m)
       tempvolwc=calc_volwc(type,(oldsuc(i)+oldsuc(i+1)/2.)) ! based on start of time step suction
       crit_temp=-fn_point(type,points7,xvoluwc,xtem,splinsl7,tempvolwc)
       if(gnodevolice(m).gt.0) crit_temp=goldtemp(m)
C    ===========================================================================
C    Calculate average temp. on freezing curve and if freezing or thawing occured
C    ===========================================================================
       if(gnewtemp(m).le.crit_temp.and.goldtemp(m).lt.-0.05) then ! freezing
        avetemp=(gnewtemp(m)+crit_temp)/2.
       elseif(gnewtemp(m).gt.crit_temp.and.gnewtemp(m).lt.-0.05 ! thawing
    1       .and.gnodevolice(m).gt.0)then
        avetemp=(gnewtemp(m)+crit_temp)/2.
       elseif(gnewtemp(m).ge.-0.05.and.gnodevolice(m).gt.0)then
        avetemp=(-0.001+crit_temp)/2.
       else
        avetemp=99.  ! no freezing or thawing happening at this node
       endif

       avesuc = ( suc0(m) + suc1(m) )/2.0    ! in unfrozen soil ( kPa ) Suction at this Guass Pt. @ t+dt/2
       gtemp(m) = gtemp(m) + 273.16               ! ( K ) Convert Temperatures to Kelvin
       if(gtemp(m).eq.273.16) gtemp(m)=273.15
       gcord(m) = gcord(m)/100.0                 ! ( m ) Convert Coordinates to meters.

c       avesuck = ( suca(m) + sucb(m) )/2.0
       volwc  = Calc_VolWc(type,avesuc)              ! ( dec ) Volumetric Water Content
       aau   = volwc/(GS(type)*(1-PORS(type)))       ! ( dec ) Gravimetric Water Content
c       IF(QW(1,2).GT.0)THEN
c        xkk   = Calc_K(type,avesuck,gnodevolice(m))     ! ( m/s ) Hydraulic Conductivity @ t + dt/2
c        xkk1  = Calc_K(type,sucb(m),gnodevolice(m))     ! ( m/s ) Hydraulic Conductivity @ t + dt
c       ELSE
       xkk   = Calc_K(type,avesuc,gnodevolice(m))       ! ( m/s ) Hydraulic Conductivity @ t + dt/2
       xkk1  = Calc_K(type,suc1(m),gnodevolice(m))      ! ( m/s ) Hydraulic Conductivity @ t + dt
c       ENDIF
       rh    = Calc_RH(avesuc,gtemp(m))              ! ( dec ) Relative Humidity
       pv    = Calc_SatVp(gtemp(m))*rh*0.1           ! ( kPa ) Vapour pressure
       dv    = Calc_Vapour_Diff(gtemp(m),volwc,
    1             gnodevolice(m),PORS(type))  ! ( s ) Coefficient of Vapour Diffusion
       xlamda = Calc_Thermal_Cond(type,aau,gtemp(m),
    1             gnodevolice(m))          ! ( W/mC) Thermal Conductivity
       xsheat = Calc_Specific_Heat(type,aau,gnodevolice(m))  ! (J/m^3-C)Coefficient of Specific Heat

C    ======================================================================
       IF( TRANSIENT )THEN
        rm2w = Calc_RM2W(type,avesuc)               ! ( 1/kPa) d(volwc)/d(suction)
        slpot = 0.1*(0.00815*(gtemp(m)-273.0) + 0.8912)**7  ! ( kPa/C) Emperical method to calculate the slope of the satvp - temp
curve (Kpa/cel)
        IF(gtemp(m).LT.273.16) slpot=0.05475*exp(0.0802*(gtemp(m)-273.16))  ! Slope if negative temps.
```

```
      IF( iteration.GT.1 )THEN          ! Calculate only if there is a chance the system will converge
       IF(m.EQ.2 .OR. m.EQ.AGAUSS )THEN
         VFLUX(i+1) = Calc_gFlux(type,i,m,lastsuc,avesuc,
     1              lastpv,pv,gtemp,gcord,gnodevolice)  ! (mm/s) Total flux across element boundary
       ENDIF
       lastsuc = avesuc                    ! save the last suction
       lastpv  = pv                        ! last vapour pressure
      ENDIF
     ELSE
      rm2w    = 0.0  ! No storage in steady state solutions
      slpot   = 0.0  ! No storage in steady state solutions
      VFLUX(i+1) = 0.0D0  ! No flux sections required in steady state
     ENDIF
C    -----------------------------------------------
C       ENTERING THE ELEMENT MATRIX LOOP
C    -----------------------------------------------
C       COMPUTING THE ELEMENT PROPERTY COEFFS AT THE GAUSS PNTS
C    -----------------------------------------------
C    -----------------------------------------------
      rd1 = (dv*pv*2.1674E-03)/(RHOWAT*gtemp(m))     ! (D1)  See pg. 105 of JOSHI's thesis
      rd2 = rh*slpot
      rd2 = rd2 + (pv*avesuc*2.1674E-03)/(gtemp(m)**2) ! (D2)  See pg. 105 of JOSHI's thesis
      rd2 = rd2 * (dv/RHOWAT)
      CWSTIFF (m) = (xkk/(GRAV*RHOWAT))+rd1       ! [Kw]  See pg. 105 of JOSHI's thesis
      CWHSTIFF(m) = rd2                           ! [Kwh] See pg. 105 of JOSHI's thesis
      CWK    (m) = -xkk1  ! for Jame Test only =0          ! Vector related to gravity See pg. 105 of JOSHI's thesis @ t + dt
      CHSTIFF (m) = xlamda+(RLATENT*rd2*RHOWAT)     ! [Kh]  See pg. 105 of JOSHI's thesis
      CHWSTIFF(m) = +RLATENT*rd1*RHOWAT           ! [Khw] See pg. 105 of JOSHI's thesis
      CWMASS  (m) = rm2w                   ! [C1]  See pg. 105 of JOSHI's thesis
      CHMASS  (m) = xsheat                 ! [C2]  See pg. 105 of JOSHI's thesis

C     ==============================================================
C     This section is added for freeze thaw to modify the
C     element stiffness and mass matrices to solve for T
C     ==============================================================

      if(avetemp.ne.99) then !  at a phase change gauss point

      gg=fn_point(type,points9,gvoluwc,xgg,splinsl9,tempvolwc)  ! see Equation 2.13 of Greg's thesis.

C     COMPUTE THE LATENT HEAT TERMS
C     -------------------------------------------------------
      if(avetemp.eq.0) avetemp=-.05          !  average temp. can not be 0 in slope function.
      m2i=-fn_slope(type,points8,ytem,yvoluwc,splinsl8,abs(avetemp))  ! slope of soil freezing curve

      latent=rhowat*m2i*Flatent/rhoice
      mass_freeze=xkk*Flatent*gg/grav/rhoice   ! mass component inModified heat transfer  equation
     1          -(Rlatent-Flatent/rhoice)*gg*rd1
     1          -(Rlatent-Flatent/rhoice)*rd2

C     THE HEAT MASS matrices is modified
c     ---------------------------------------
      CHMASS(m)= CHMASS(m)+latent


C     THE H and W Stiffness matrices are modified
C     ----------------------------------------------------
      tgrad=abs((tem(i+1)-tem(i))/(ycord(i+1)-ycord(i)))
      if(tgrad.lt.0.3)then
       mass_freeze=mass_freeze*10**(-10*(0.3-tgrad))
      endif
      CHSTIFF(m)= xlamda + mass_freeze  ! add mass transfer freezing
      CHWSTIFF(m)= 0.                          ! de-couple heat and mass transfer equations
      CWSTIFF(m)= 0.
      CWHSTIFF(m)= 0.

      endif ! if avetemp.ne.99

      ENDDO  ! DO m = 1,AGAUSS
C     ==============================================================
```

```
C    ================================================================
C     Forming the element stiffness (BTB) and mass storage (STS) matices
C    ================================================================
     CALL ELEM1Q1(i)

C    ================================================================
C      Forming the global stiffness and mass storage matices
C    ================================================================
C
C       SYSTIFF              SYSMAS          SYSF
C    ___________          _________       ___
C    |      |      |     |      | |
C
C    _      _      _      _     _ _
C    |      |{  }  |      |{ * } | |
C    | Kw   Kwh |{ Suc }  | C1  C3 |{ Suc } |Fw|
C    |      |{  } + |     |{ * } =|  |
C    | Khw  Kh |{ Tem }   | 0  C2 |{ Tem } |Fh|
C    |_     _|{  } |_     _|{  } |_ _|
C
C    ================================================================
     DO j = 1,PNODES
      l    = NELCON(j,i)
      glw (l) = glw (l) + DSTK (j)
      DO k = 1,PNODES
        m = NELCON(k,i)
C         Adding the element matrices to the global matrices
        gbtbw (l,m) = gbtbw (l,m) + BTBW (j,k)
        gbtbwh(l,m) = gbtbwh(l,m) + BTBWH(j,k)
        gbtbh (l,m) = gbtbh (l,m) + BTBH (j,k)
        gbtbhw(l,m) = gbtbhw(l,m) + BTBHW(j,k)
        gstsw (l,m) = gstsw (l,m) + STSW (j,k)
        gstsh (l,m) = gstsh (l,m) + STSH (j,k)
        gstswh(l,m) = gstswh(l,m) + STSWH(j,k)
      ENDDO
     ENDDO
C    ------------------------------------------
 172 CONTINUE      ! END OF 'DO 172 i = 1,NELEM' Loop
C   ================================================================
C   ================================================================
C   CONSTRUCTING the system stiffness and storage matrices.
C   ================================================================

    DO j = 1,NNODES
     l = j+NNODES
     DO i = 1,NNODES
       k = i+NNODES
       SYSTIF(i,j) = gbtbw (i,j)
       SYSTIF(i,l) = gbtbwh(i,j)
       SYSTIF(k,j) = gbtbhw(i,j)
       SYSTIF(k,l) = gbtbh (i,j)
       SYSMAS(i,j) = gstsw (i,j)
       SYSMAS(k,j) = 0.0
       SYSMAS(k,l) = gstsh (i,j)
     ENDDO

      SYSF (j) = glw (j)
      SYSF (l) = glh (j)
    ENDDO

C   ================================================================
    CALL CNICOL(iteration)          ! Solve for Nodal Suctions and Temperatures
C   ================================================================

C   Modify suctions based on newly solved temperature below freezing when ice has formed.

    CALL REVERSE_SPLINES ! Reverse the splines order so the it is in ascending suction order
     DO i=1,MAX_TYPES
      CALL WtSplin2(i,POINTS1,XVOLWC,XSUC,SPLINSL1)
     ENDDO
    DO i = 1,NNODES
```

```
      type = SOILTYPE(i)
      tempvolwc=calc_volwc(type,phia(i))    !
      crit_temp=fn_point(type,points7,xvoluwc,xtem,splinsl7,tempvolwc)
      if(tem(i).le.crit_temp .or. nodvolice(i).gt.0) then
        tempvolwc=fn_point(type,points8,ytem,yvoluwc,splinsl8,abs(tem(i)))
        SUCNOD(i) = FN_POINT(type,POINTS1,XVOLWC,XSUC,SPLINSL1,tempvolwc)
      endif
     ENDDO ! i = 1,nnodes
      CALL REVERSE_SPLINES ! Re-Establish the original spline order
      DO i=1,MAX_TYPES
       CALL WtSplin2(i,POINTS1,XSUC,XVOLWC,SPLINSL1)
      ENDDO

C    Calculate a the appropriate time step on the first two iterations
C    ================================================================
     IF( iteration.LT.2 )THEN
      IF( TTIME.GT.0.0D0 )THEN    ! Use specified time step as first time step
        CALL CALCULATE_TIME_STEP  ! Adjust time step
      ENDIF
     ELSE
      converged = Convergence()   ! Check to see if System has converged
     ENDIF
C    ================================================================
C    IF NOT CONVERGED, USE RELAXATION TO HELP CONVERGE MORE RAPIDLY
C    ================================================================
     IF(.NOT.(converged.OR.maxd_out) )THEN
      CALL RELAXATION    ! Implements relaxation scheme
      DO i = 1,NNODES
        PRESNOD(i) = SUCNOD(i) ! Save current suctions for next iteration
        PRETNOD(i) = TEM   (i) ! Save current temp.s for next iteration
      ENDDO
     ENDIF


     ENDDO ! End of DO WHILE( .NOT.(converged.OR.maxd_out) )
C    ================================================================
C    ****************************************************************
C    End of the iteration loop
C    ****************************************************************
C    ================================================================
C    Modify Node Vol Ice Storage After Converged at this Time Step
C    ================================================================
     do i=1,nnodes
      oldtem(i)=phia(i+nnodes)
      oldsuc(i)=phia(i)
     enddo

     if(ice)then
     call dice(oldtem,tem,oldsuc,sucnod)     !  calculated change in nodal ice content over last time step
     do i=1,nnodes
      nodvolice(i)=nodvolice(i)+delice(i)    ! add change in ice content to nodal ice content array
     enddo

     RETURN
     END
```

```
C    ================================================================
      SUBROUTINE JSHAPE(Element,NP,GP)
C    ================================================================
C     This subroutine determines the shape functions for two or three
C    noded elements  (see page 103 JOSHI thesis)
C    ================================================================
     IMPLICIT NONE                 ! Ensure that all variables have been correctly defined
     INCLUDE 'FUNCTION.FT'           ! Contains all function declarations
     INCLUDE 'DECLARE.FT'            ! Contains all common block declarations
C    ================================================================
```

```
      INTEGER Element              ! Current element
      REAL  GP  (MAX_GAUSS)          ! Gauss Point Property
      INTEGER i                    ! Current Gauss Point
      REAL  NP  (MAX_NODES)          ! Nodal Property
      INTEGER N1                   ! First  node of current element
      INTEGER N2                   ! Second node of current element
      INTEGER N3                   ! Third  node of current element
      REAL  sl1                    ! Temporary Variable
      REAL  sl2                    ! Temporary Variable
      REAL  sl3                    ! Temporary Variable
C  ==========================================================
      IF(PNODES.EQ.2)THEN  ! 2 nodes per element
        N1  = NELCON(1,Element)
        N2  = NELCON(2,Element)
        DO i = 1,AGAUSS
          sl1  = 0.5*(1.0-AX(i))
          sl2  = 0.5*(1.0+AX(i))
          GP(i) = NP(N1)*sl1+NP(N2)*sl2
        ENDDO
      ELSE             ! 3 nodes per element
        N1  = NELCON(1,Element)
        N2  = NELCON(2,Element)
        N3  = NELCON(3,Element)
        DO i = 1,AGAUSS
          sl1  =  0.5*(AX(i)*   (AX(i)-1.0))
          sl2  = -1.0*(AX(i)+1.0)*(AX(i)-1.0)
          sl3  =  0.5*(AX(i)*   (AX(i)+1.0))
          GP(i) = NP(N1)*sl1+NP(N2)*sl2+NP(N3)*sl3
        ENDDO
      ENDIF
C  ==========================================================
      RETURN
      END
```

```
C  ==========================================================
      SUBROUTINE JSHAPE2(Element,NP,GP)
C  ==========================================================
C     This subroutine determines the shape functions for two or three
C    noded elements  (see page 103 JOSHI thesis)
C  ==========================================================
      IMPLICIT NONE                ! Ensure that all variables have been correctly defined
      INCLUDE 'FUNCTION.FT'          ! Contains all function declarations
      INCLUDE 'DECLARE.FT'           ! Contains all common block declarations
C  ==========================================================
      INTEGER Element              ! Current element
      REAL  GP  (MAX_GAUSS)          ! Gauss Point Property
      INTEGER i                    ! Current Gauss Point
      REAL  NP  (MAX_NODES)          ! Nodal Property
      INTEGER N1                   ! First  node of current element
      INTEGER N2                   ! Second node of current element
      INTEGER N3                   ! Third  node of current element
      REAL  sl1                    ! Temporary Variable
      REAL  sl2                    ! Temporary Variable
      REAL  sl3                    ! Temporary Variable
C  ==========================================================
      IF(PNODES.EQ.2)THEN  ! 2 nodes per element
        N1  = NELCON(1,Element)
        N2  = NELCON(2,Element)
        DO i = 1,AGAUSS
          sl1  = 0.5*(1.0-AX(i))
          sl2  = 0.5*(1.0+AX(i))
          GP(i) = NP(N1)*sl1+(NP(N1)*SL2*.8+NP(N2)*sl2*.2)
        ENDDO
      ELSE             ! 3 nodes per element
        N1  = NELCON(1,Element)
        N2  = NELCON(2,Element)
        N3  = NELCON(3,Element)
        DO i = 1,AGAUSS
```

```
        sl1  = 0.5*(AX(i)*    (AX(i)-1.0))
        sl2  = -1.0*(AX(i)+1.0)*(AX(i)-1.0)
        sl3  = 0.5*(AX(i)*    (AX(i)+1.0))
        GP(i) = NP(N1)*sl1+NP(N2)*sl2+NP(N3)*sl3
      ENDDO
    ENDIF
C  ==================================================================
    RETURN
    END
```

```
C  This subroutine calculates the Leaf Area Index
C  ==================================================================
```

SUBROUTINE LeafAreaIndex

```
C  ==================================================================
    IMPLICIT NONE                    ! Ensure that all variables have been correctly defined
    INCLUDE 'FUNCTION.FT'
    INCLUDE 'DECLARE.FT'
C  ==================================================================
    REAL   rday                      ! defines nday in real value terms
C  ==================================================================
    if(VEGETATION)then
     rday = NDAY*1.0                 ! Change nday to a real number
C    ***** Green LAI *****
     IF(rday.LT.(EXP(XLAIDAY(1,1))-0.5))THEN
      LAI = 0.0                      ! if spec. but before grow. seas.
     ELSEIF(rday.GT.(EXP(XLAIDAY(POINTS5(1),1))+0.5))THEN
      LAI = 0.0                      ! if spec. but after grow. season
     ELSE
      LAI = FN_POINT(1,POINTS5,XLAIDAY,XLAI,SPLINSL5,
   1          rday)                  ! if spec. and in grow. season
     ENDIF
C    ***** Mulch LAI *****
     IF(POINTS6(1).NE.0)THEN         ! if mulch is specified
      IF(rday.LT.(EXP(XMULCHDAY(1,1))-0.5))THEN
       MULCH = 0.0                   ! if spec. but before first spec
       ELSEIF(rday.GT.(EXP(XMULCHDAY(POINTS6(1),1))+0.5))THEN
       MULCH = 0.0                   ! if spec. but after first spec
      ELSE
       MULCH = FN_POINT(1,POINTS6,XMULCHDAY,XMULCH,SPLINSL6,
   1          rday)                  ! if spec. and in grow. season
      ENDIF
     ELSE                            ! if veget. spec. but mulch not spec.
      MULCH = 0.0
     ENDIF
    else
     LAI = 0.0                       ! If veget. not specified
     MULCH = 0.0
    endif
C  ==================================================================
    RETURN
    END
```

SUBROUTINE mprove(au,n,m1,m2,np,mp,al,mpl,indx,x,a,b)

```
C  ==================================================================
    IMPLICIT NONE
    INTEGER m1,m2,mp,mpl,n,np,indx(n)
    DOUBLE PRECISION a(np,mp),au(np,mp),al(np,mpl),b(n),x(n)
C  ==================================================================
C  Uses banbks
C   Improves a solutin vector x(1:n) of the linear set of equations
C  A*X=B.  The matrix a(1:n,1:n), and the vectors b(1:n) and x(1:n)
C  are input, as is the dimension n.  Also input are a and alud, the LU
C  decomposition of a as returned by bandec, and the vector  indx
C  also returned by that routine.  On output, only x(1:n) is modified,
```

```
C   to an improved set of values.
C   ================================================================
    INTEGER        i,j,k,mm,NMAX
    PARAMETER      (NMAX=210)
    DOUBLE PRECISION r(NMAX)
    DOUBLE PRECISION sdp
C   ================================================================
    mm = m1 + m2 + 1
    do 12 i=1,n
      sdp = -b(i)
      k  = i-m1-1
      do 11 j=max(1,1-k),min(mm,n-k)
        sdp = sdp + a(i,j)*x(j+k)
 11     enddo
      r(i) = sdp
 12   enddo
    CALL banbks(au,n,m1,m2,np,mp,al,mpl,indx,r)
    do 13 i=1,n
      x(i) = x(i) - r(i)
 13   enddo
C   ================================================================
    RETURN
    END
```

SUBROUTINE PREPROCESSOR

```
C   ================================================================
C     This subroutine obtains the run time information, the data input
C   file to define the problem, and writes the initial conditions
C   and properties to the output file.
C     Subroutines called:
C        get_run_time: obtains the run time information from the user
C        set_initial_suction: determines the initial suctions and
C            water contents based on initial conditions.
C        write_out: writes detailed information to the output file.
C        write_node: writes non-detailed info. to the output file.
C   ================================================================
    IMPLICIT NONE
    INCLUDE 'FUNCTION.FT'          ! Contains all function declarations
    INCLUDE 'DECLARE.FT'           ! Contains all common block declarations
C   ----------------------------------------
    INTEGER    i,j,l               ! Loop counters
C   ================================================================

C   ================================================================
C             Start Timer for total run time
C   ================================================================
    TIME0 = SECNDS(0.0)

C   ================================================================
C          Call GetRun to get the run time parameters
C   ================================================================
    CALL Get_Run_Time
    call set_init_suction
C   ================================================================
C   Calculate which nodes correspond to which elements
C   ================================================================
    l = 1
    DO i = 1,NELEM                 ! Finding the corresponding node for each gauss point
     j = 0
     DO WHILE (j.LT.PNODES)
       j = j + 1
       NELCON(j,i) = l
       IF(j.LT.PNODES) l = l + 1
     ENDDO
    ENDDO

C   ================================================================
    OPEN (UNIT=48,FILE=OUTPUT,STATUS='UNKNOWN')      ! Open the Output File
```

```
C  ==========================================================
C   Write out program information header to output file
C  ==========================================================
   write(48,*) '                                          '
   write(48,*) '                                          '
   WRITE(48,*) '              SoilCover  Version 1.2     '
   WRITE(48,*) '                  June 1994               '
   write(48,*) '         Department of Civil Engineering '
   write(48,*) '           University of Saskatchewan    '
   write(48,*) '            Saskatoon, Saskatchewan      '
   write(48,*) '               Canada  S7N 0W0           '
   write(48,*) '                                          '
   write(48,*) '                                          '
C==============================================================


C  ==========================================================
C       Print starting values to output file
C  ==========================================================
    NDAY = 0
    CALL DAILY_INPUT
    IF( TRANSIENT )THEN
    IF(PrintTime.EQ.1)THEN
     write(48,*)'            *** Noon output ***    '
    ELSE
     write(48,*)'          *** Midnight output ***  '
    ENDIF
    write(48,*)' '
    IF(DETAILED)THEN
     CALL WRITE_OUT(0.0)
    ELSE
     CALL WRITE_NOD(0.0)
    ENDIF
   ENDIF

C  ==========================================================
c   End of PREPROCESSOR
C  ==========================================================
   RETURN
   END
```

```
C  *************************************************************
C  *                                            *
C  *           MAIN INPUT ROUTINE                           *
C  *                                            *
C  *************************************************************
        SUBROUTINE MAIN_IN(InFile,Debug_Splines)
C  *************************************************************
C    This subroutine reads all the input data supplied in the input
C   data file.
C    Subroutines called:
C      Err_Msg: checks that the data falls within array bounds
C      splines: splines the soil property data
C  ==========================================================
        IMPLICIT NONE              ! Ensure that all variables have been correctly defined
        INCLUDE 'FUNCTION.FT'         ! Contains all function declarations
        INCLUDE 'DECLARE.FT'          ! Contains all common block declarations
C  ==========================================================
        CHARACTER*80  aline           ! Used to skip over file comments
        integer    analysis_code     ! Used to read type of analysis
        CHARACTER*(*) InFile          ! The name of the main input file
        CHARACTER*14  DayFile          ! The name of the Daily Input File
        LOGICAL     Debug_Splines     ! Flag to indicate splines are to be graphed to screen
        CHARACTER*14  PrpFile          ! The name of the Property Input File
        CHARACTER*14  CnstFile         ! The name of the Constants Input File
        CHARACTER*14  MeshFile          ! The name of the Mesh Input File
        CHARACTER*14  IceFile         ! The name of the freeze/thaw Input File
        INTEGER     IceFlag          ! flag used to determine if freeze/thaw is to be modelled
```

```
      INTEGER    VegFlag            ! flag used to determine if veget is to be modelled
      CHARACTER*14 VgtFile            ! The name of the Vegetation Input File
      integer    namelength         ! Temporary Variable
C =================================================================
      OPEN(UNIT=10,FILE=InFile,STATUS='OLD')
C ---------------------------------------------
      READ(10,FMT='(A80)',ERR=999)aline  ! "Main Input File for SoilCover"
      READ(10,FMT='(A80)',ERR=999)aline  ! "******************************"
      READ(10,FMT='(A80)',ERR=999)aline  ! A Blank Line
C ---------------------------------------------
      READ(10,FMT='(A80)',ERR=999)aline  ! "Analysis Type "
C Analysis Types:
C   0 = SteadyState
C   1 = DarcyFlux
C   2 = SoilCover
      READ(10,*,ERR=999)analysis_code
      IF(analysis_code.EQ.0)THEN
        STEADYSTATE = TRUE
        ICE       = TRUE
        TRANSIENT  = FALSE
        DFLUX     = FALSE
        VEGETATION  = FALSE ! Not currently supported in steady state
      ELSEIF(analysis_code.EQ.1)THEN
        STEADYSTATE = FALSE
        ICE       = TRUE
        TRANSIENT  = TRUE
        DFLUX     = TRUE
      ELSEIF(analysis_code.EQ.2)THEN
        STEADYSTATE = FALSE
        TRANSIENT  = TRUE
        ICE       = TRUE
        DFLUX     = FALSE
      ELSE
        WRITE(*,*)' Analysis Type ',analysis_code,' is not supported.'
        stop
      ENDIF
      READ(10,FMT='(A80)',ERR=999)aline  ! A Blank Line
C ---------------------------------------------
      READ(10,FMT='(A80)',ERR=999)aline  ! "Output File Name"
      READ(10,FMT='(A80)',ERR=999)aline  ! Name of Output File
      namelength = 1               ! determine location of "."
      DO WHILE(aline(namelength:namelength).NE.".")
       namelength = namelength + 1
      ENDDO
      namelength = namelength + 3      ! add spaces for "out"
      OUTPUT = aline(1:namelength)
      READ(10,FMT='(A80)',ERR=999)aline  ! A Blank Line
C ---------------------------------------------
      READ(10,FMT='(A80)',ERR=999)aline  ! "Output Data Corresponding to(1-noon,2-mid"
      READ(10,*,ERR=999)PrintTime
      CALL ERR_MSG('Print Time',PrintTime,2)
      READ(10,FMT='(A80)',ERR=999)aline  ! A Blank Line"
C ---------------------------------------------
      READ(10,FMT='(A80)',ERR=999)aline  ! "Constants Input FileName"
      READ(10,FMT='(A80)',ERR=999)aline  ! Name of Constants Input File
      namelength = 1               ! determine location of "."
      DO WHILE(aline(namelength:namelength).NE.".")
       namelength = namelength + 1
      ENDDO
      namelength = namelength + 3      ! add spaces for file name extension
      CnstFile = aline(1:namelength)
      READ(10,FMT='(A80)',ERR=999)aline  ! A Blank Line
C ---------------------------------------------
      READ(10,FMT='(A80)',ERR=999)aline  ! "Soil Property Input FileName"
      READ(10,FMT='(A80)',ERR=999)aline  ! Name of Soil Property File
      namelength = 1               ! determine location of "."
      DO WHILE(aline(namelength:namelength).NE.".")
       namelength = namelength + 1
      ENDDO
      namelength = namelength + 3      ! add spaces for file name extension
      PrpFile = aline(1:namelength)
```

```
      READ(10,FMT='(A80)',ERR=999)aline   ! A Blank Line
C  ----------------------------------------------------------
      READ(10,FMT='(A80)',ERR=999)aline   ! "Mesh Data Input FileName"
      READ(10,FMT='(A80)',ERR=999)aline   ! Name of Mesh Input File
      namelength = 1                    ! determine location of "."
      DO WHILE(aline(namelength:namelength).NE.".")
       namelength = namelength + 1
      ENDDO
      namelength = namelength + 3       ! add spaces for file name extension
      MeshFile = aline(1:namelength)
      READ(10,FMT='(A80)',ERR=999)aline   ! A Blank Line
C  ----------------------------------------------------------
      READ(10,FMT='(A80)',ERR=999)aline   ! "Daily Input FileName"
      READ(10,FMT='(A80)',ERR=999)aline   ! Name of Daily InputFile
      namelength = 1                    ! determine location of "."
      DO WHILE(aline(namelength:namelength).NE.".")
       namelength = namelength + 1
      ENDDO
      namelength = namelength + 3       ! add spaces for file name extension
      DayFile = aline(1:namelength)
      READ(10,FMT='(A80)',ERR=999)aline   ! A Blank Line
C  ----------------------------------------------------------
      IF( TRANSIENT )THEN   ! The rest of this only makes sense for a transient analysis
        READ(10,FMT='(A80)',ERR=999)aline   ! "Will Vegetation be Modelled? (1=Yes,2=No)"
        READ(10,*,ERR=999)VegFlag         ! Read flag
        IF(VegFlag.EQ.1)THEN
               VEGETATION = TRUE
        ELSE
               VEGETATION = FALSE
        ENDIF
        READ(10,FMT='(A80)',ERR=999)aline   ! A Blank Line
C  ----------------------------------------------------------
        IF( VEGETATION ) THEN
         READ(10,FMT='(A80)',ERR=999)aline   ! "Vegetation Input FileName"
         READ(10,FMT='(A80)',ERR=999)aline   ! Name of Vegetation InputFile
         namelength = 1                    ! determine location of "."
         DO WHILE(aline(namelength:namelength).NE.".")
               namelength = namelength + 1
         ENDDO
         namelength = namelength + 3       ! add spaces for file name extension
         VgtFile = aline(1:namelength)
        ELSE
               VgtFile = ' '
         READ(10,FMT='(A80)',ERR=999)aline   ! "Vegetation Input FileName"
         READ(10,FMT='(A80)',ERR=999)aline   ! Name of Vegetation InputFile
         READ(10,FMT='(A80)',ERR=999)aline   ! A Blank Line
        ENDIF

      ENDIF
C  ----------------------------------------------------------
        READ(10,FMT='(A80)',ERR=999)aline   ! "Will Freeze/thaw be Modelled? (1=Yes,2=No)"
        READ(10,*,ERR=999)IceFlag         ! Read flag
        IF(IceFlag.EQ.1)THEN
               ICE = TRUE
        ELSE
               ICE = FALSE
        ENDIF
        READ(10,FMT='(A80)',ERR=999)aline   ! A Blank Line
C  ----------------------------------------------------------
        IF( ICE ) THEN
         READ(10,FMT='(A80)',ERR=999)aline   ! "Freeze/Thaw Input FileName"
         READ(10,FMT='(A80)',ERR=999)aline   ! Name of Freeze/Thaw InputFile
         namelength = 1                    ! determine location of "."
         DO WHILE(aline(namelength:namelength).NE.".")
               namelength = namelength + 1
         ENDDO
         namelength = namelength + 3       ! add spaces for file name extension
         IceFile = aline(1:namelength)
        ELSE
               IceFile = ' '
        ENDIF
```

```
      CLOSE(UNIT=10) ! Closing the main input file
C     ------------------------------------------------
      OPEN(UNIT=12,FILE=CnstFile,STATUS='OLD')
      CALL CONSTANT_INPUT
      CLOSE(UNIT=12)
C     ------------------------------------------------
      OPEN(UNIT=12,FILE=PrpFile,STATUS='OLD')
      CALL PROPERTY_INPUT
      CLOSE(UNIT=12)
C     ------------------------------------------------
      IF( VEGETATION ) THEN
       OPEN(UNIT=10,FILE=VgtFile,STATUS='OLD')
       CALL VEGETATION_INPUT
      ENDIF
C     ------------------------------------------------
      OPEN(UNIT=12,FILE=MeshFile,STATUS='OLD')
      CALL MESH_INPUT
      CLOSE(UNIT=12)
C     ------------------------------------------------
      IF( ICE ) THEN
       OPEN(UNIT=14,FILE=IceFile,STATUS='OLD')
       CALL ICE_INPUT
       CLOSE(UNIT=14)
      ENDIF
C     ------------------------------------------------
      CALL SPLINES(Debug_Splines)      ! Spline the Soil Property Data
C     ------------------------------------------------

C     ------------------------------------------------
      OPEN(UNIT=12,FILE=DayFile,STATUS='OLD')
      CALL INITIAL_DAILY_INPUT
C     ================================================================
      RETURN
998   WRITE(*,*) aline
      STOP 'Error in Daily Input data file'
999   WRITE(*,*) aline
      STOP 'Error in Main Input data file'
      END
C     ****************************************************************
```

```
C     ****************************************************************
C     *                                         *
C     *         SOIL PROPERTY INPUT ROUTINE             *
C     *                                         *
C     ****************************************************************

C     ================================================================
      SUBROUTINE PROPERTY_INPUT
C     ================================================================
C     This subroutine reads all the soil property input data supplied
C     in the soil property input data file.
C     Subroutines called:
C       Err_Msg: checks that the data falls within array bounds
C     ================================================================
      IMPLICIT NONE              ! Ensure that all variables have been correctly defined
      INCLUDE 'FUNCTION.FT'          ! Contains all function declarations
      INCLUDE 'DECLARE.FT'           ! Contains all common block declarations
C     ================================================================
      CHARACTER*80  aline            ! Used to skip over file comments
      INTEGER    j                ! Loop Counter
      integer    type             ! the soil type number
      integer    wctype             ! specifies whether user inputs soil properties in Grav. or Vol. w/c
      real      xwc              ! temporary water content variable
C     ================================================================
      READ(12,FMT='(A80)',ERR=999)aline   ! "SoilProperty InputFile for SoilCover"
      READ(12,FMT='(A80)',ERR=999)aline   ! "*************************************"
```

```
      READ(12,FMT='(A80)',ERR=999)aline   ! A Blank Line
C  -------------------------------------------------
      DO type = 1,MAX_TYPES
        READ(12,FMT='(A80)',ERR=999)aline   ! "Soil Type #"
        READ(12,FMT='(A80)',ERR=999)aline   ! "========"
        READ(12,FMT='(A80)',ERR=999)aline   ! "Porosity  Specific"
        READ(12,FMT='(A80)',ERR=999)aline   ! "          Gravity"
        READ(12,*,ERR=999) PORS(type),GS(type)
        READ(12,FMT='(A80)',ERR=999)aline   ! A Blank Line
C  -------------------------------------------------
        READ(12,FMT='(A80)',ERR=999)aline   ! "Moisture Characterist"
        READ(12,FMT='(A80)',ERR=999)aline   ! "-------------------"
        READ(12,FMT='(A80)',ERR=999)aline   ! "NumberOf  Mv  WaterC"
        READ(12,FMT='(A80)',ERR=999)aline   ! "DataPoints (1/kPa)"
        READ(12,*,ERR=999)POINTS1(type),RM2WA(type),wctype
        READ(12,FMT='(A80)',ERR=999)aline   ! "Suction    WaterCont"
        READ(12,FMT='(A80)',ERR=999)aline   ! "(kPa)      (dec)"
        DO j=1,POINTS1(type)
         IF(wctype.EQ.1)THEN
              READ(12,*,ERR=999)XSUC(j,type),XWC
              XVOLWC(j,type)=XWC*GS(type)*(1.0E0-PORS(type))
         ELSE
              READ(12,*,ERR=999)XSUC(j,type),XVOLWC(j,type)
         ENDIF
        ENDDO
        XSUC1  (type) = XSUC  (1,type)
        SUCT_INT(type) = XVOLWC(1,type) + XSUC(1,type)*RM2WA(type)
        READ(12,FMT='(A80)',ERR=999)aline   ! A Blank Line
C  -------------------------------------------------
        READ(12,FMT='(A80)',ERR=999)aline   ! "Hydraulic Conductivity"
        READ(12,FMT='(A80)',ERR=999)aline   ! "-------------------"
        READ(12,FMT='(A80)',ERR=999)aline   ! "NumberOf  SatHydCond"
        READ(12,FMT='(A80)',ERR=999)aline   ! "DataPoints (cm/s)"
        READ(12,*,ERR=999)POINTS2(type),SATK(type),impfact(type)
        READ(12,FMT='(A80)',ERR=999)aline   ! "Suction    HydCond"
        READ(12,FMT='(A80)',ERR=999)aline   ! "(kPa)      (cm/s)"
        DO j=1,POINTS2(type)
              READ(12,*,ERR=999)XKSUC(j,type),XK(j,type)
        ENDDO
        READ(12,FMT='(A80)',ERR=999)aline   ! A Blank Line
C  -------------------------------------------------
        READ(12,FMT='(A80)',ERR=999)aline   ! "Thermal Conductivity"
        READ(12,FMT='(A80)',ERR=999)aline   ! "-------------------"
        READ(12,FMT='(A80)',ERR=999)aline   ! "NumberOf   WaterCont"
        READ(12,FMT='(A80)',ERR=999)aline   ! "DataPoints  Type"
        READ(12,*,ERR=999)POINTS3(type),wctype
        READ(12,FMT='(A80)',ERR=999)aline   ! "Water     Thermal"
        READ(12,FMT='(A80)',ERR=999)aline   ! "Content   Conduct"
        READ(12,FMT='(A80)',ERR=999)aline   ! "(dec)     (W/m^2)"
        DO j=1,POINTS3(type)
              IF(wctype.EQ.1)THEN
               READ(12,*,ERR=999)XLAMDWC(j,type),XLAMD(j,type)
              ELSE
               READ(12,*,ERR=999)XWC,XLAMD(j,type)
               XLAMDWC(j,type)=XWC/(GS(type)*(1.0E0-PORS(type)))
              ENDIF
        ENDDO
        READ(12,FMT='(A80)',ERR=999)aline   ! A Blank Line
C  -------------------------------------------------
        READ(12,FMT='(A80)',ERR=999)aline   ! "Specific Heat Function"
        READ(12,FMT='(A80)',ERR=999)aline   ! "-------------------"
        READ(12,FMT='(A80)',ERR=999)aline   ! "NumberOf   WaterCont"
        READ(12,FMT='(A80)',ERR=999)aline   ! "DataPoints  Type"""
        READ(12,*,ERR=999)POINTS4(type),wctype
        READ(12,FMT='(A80)',ERR=999)aline   ! "Water     Specific"
        READ(12,FMT='(A80)',ERR=999)aline   ! "Content    Heat"
        READ(12,FMT='(A80)',ERR=999)aline   ! "(dec)     (J/m^3-C)"
        DO j=1,POINTS4(type)
              IF(wctype.EQ.1)THEN
               READ(12,*,ERR=999)XSHWC(j,type),XSH(j,type)
              ELSE
```

```
                READ(12,*,ERR=999)XWC,XSH(j,type)
                XSHWC(j,type)=XWC/(GS(type)*(1.0E0-PORS(type)))
              ENDIF
        ENDDO
        READ(12,FMT='(A80)',ERR=999)aline   ! A Blank Line
      ENDDO
C  ==================================================================
      RETURN
999   WRITE(*,*) aline
      STOP 'Error in soil property data file'
      END
C  ****************************************************************
```

```
C  ****************************************************************
C  *                                          *
C  *             MESH INPUT ROUTINE                    *
C  *                                          *
C  ****************************************************************

C  ==================================================================
      SUBROUTINE MESH_INPUT
C  ==================================================================
C    This subroutine reads all the input data supplied in the mesh
C  input data file.
C    Subroutines called:
C      Err_Msg: checks that the data falls within array bounds
C  ==================================================================
      IMPLICIT NONE                 ! Ensure that all variables have been correctly defined
      INCLUDE 'FUNCTION.FT'              ! Contains all function declarations
      INCLUDE 'DECLARE.FT'              ! Contains all common block declarations
C  ==================================================================
      CHARACTER*80 aline              ! Used to skip over file comments
      INTEGER     element_type
      INTEGER     i                ! Loop Counter
      INTEGER     junk               ! Used to skip over integer in input file
      integer     type              ! the layer value of each node
C  ==================================================================
      READ(12,FMT='(A80)',ERR=999)aline   ! "Soil Mesh Data File For SoilCover"
      READ(12,FMT='(A80)',ERR=999)aline   ! "================================="
      READ(12,FMT='(A80)',ERR=999)aline   ! A Blank Line
C  ------------------------------------------
      READ(12,FMT='(A80)',ERR=999)aline   ! "Convergence Criteria"
      READ(12,FMT='(A80)',ERR=999)aline   ! "--------------------"
      READ(12,FMT='(A80)',ERR=999)aline   ! "Max.    Max.Change   Max.Change"
      READ(12,FMT='(A80)',ERR=999)aline   ! "Iterations Suction    Temperature"
      READ(12,FMT='(A80)',ERR=999)aline   ! "           (%)          (%)"
      READ(12,*,ERR=999)MXITER,PUSNORM,PUTNORM,SUC_DAMP,TEM_DAMP
      PUSNORM  = PUSNORM/100.0  ! Convert from % to decimal
      PUTNORM  = PUTNORM/100.0  ! Convert from % to decimal
      SUC_DAMP = SUC_DAMP/100.0  ! Convert from % to decimal
      TEM_DAMP = TEM_DAMP/100.0  ! Convert from % to decimal
      READ(12,FMT='(A80)',ERR=999)aline   ! A Blank Line
C  ------------------------------------------
      READ(12,FMT='(A80)',ERR=999)aline   ! "Time Step Control"
      READ(12,FMT='(A80)',ERR=999)aline   ! "-----------------"
      READ(12,FMT='(A80)',ERR=999)aline   ! "Max.Change  Max.Change Min."
      READ(12,FMT='(A80)',ERR=999)aline   ! "Suction    Temp     Time"
      READ(12,FMT='(A80)',ERR=999)aline   ! "(%)  (%) (secnds) (secnds) "
      IF( TRANSIENT )THEN
        READ(12,*,ERR=999)TOLS,TOLT,MIN_DELTAT,FIRST_DELTAT,MAX_DELTAT
        DELTAT = FIRST_DELTAT
        TOLS   = TOLS/100.0        ! Convert from % to decimal
        TOLT   = TOLT/100.0        ! Convert from % to decimal
      ELSE  ! This is a steady state analysis
        READ(12,FMT='(A80)',ERR=999)aline   ! Not interested in these values
        TOLS       = 0.0
        TOLT       = 0.0
```

```
            DELTAT      = 86400.0
            MIN_DELTAT  = 0.0
            FIRST_DELTAT = 86400.0
            MAX_DELTAT  = 0.0
          ENDIF
          READ(12,FMT='(A80)',ERR=999)aline    ! A Blank Line
C ------------------------------------------------------------
          READ(12,FMT='(A80)',ERR=999)aline    ! "Soil Profile Data"
          READ(12,FMT='(A80)',ERR=999)aline    ! "-----------------"
          READ(12,FMT='(A80)',ERR=999)aline    ! "NumberOf Element NumberOf"
          READ(12,FMT='(A80)',ERR=999)aline    ! " Nodes   Type       GaussPts"
          READ(12,*,ERR=999)NNODES,element_type,AGAUSS
C ****************************************************************
          IF( element_type.EQ.1 )THEN
            PNODES = 2
            NELEM  = NNODES - 1
          ELSEIF( element_type.EQ.2 )THEN
            PNODES = 3
            NELEM  = NNODES - 1
            NNODES = 2*(NNODES-1) + 1
          ELSE
            WRITE(*,*) 'Unsupported Element Type'
            stop
          ENDIF
          CALL GAUSS_DATA  ! Read in the Gauss Wgts & Locations
C ****************************************************************
          CALL ERR_MSG('NNODES',NNODES,MAX_NODES)
          CALL ERR_MSG('NELEM',NELEM,MAX_ELEM)
          CALL ERR_MSG('AGAUSS',AGAUSS,MAX_GAUSS)
          CALL ERR_MSG('PNODES',PNODES,MAX_PNODES)
          READ(12,FMT='(A80)',ERR=999)aline    ! A Blank Line
C ------------------------------------------------------------
          READ(12,FMT='(A80)',ERR=999)aline    ! "Initial Moisture Conditions"
          READ(12,FMT='(A80)',ERR=999)aline    ! "---------------------------"
          READ(12,FMT='(A80)',ERR=999)aline    ! "Specified by -> 1=GWC,2=Suct,3=VWC"
          IF( TRANSIENT )THEN
           READ(12,*,ERR=999)MOISCODE
          ELSE
           READ(12,FMT='(A80)',ERR=999)aline    ! Not required for steady state
           MOISCODE = 2
          ENDIF
          READ(12,FMT='(A80)',ERR=999)aline    ! A Blank Line
C ------------------------------------------------------------
          READ(12,FMT='(A80)',ERR=999)aline    ! "Mesh Data"
          READ(12,FMT='(A80)',ERR=999)aline    ! "---------"
          READ(12,FMT='(A80)',ERR=999)aline    ! "Node   Soil  Elevation  Moist"
          READ(12,FMT='(A80)',ERR=999)aline    ! "Node   Type  (cm)     (dec."
          IF( (MOISCODE.EQ.1) .OR. (MOISCODE.EQ.3) )THEN
           DO i = 1,NNODES,(PNODES-1)
            READ(12,*,ERR=999)junk,SOILTYPE(i),YCORD(i),WTWC(i),TEM(i)
           ENDDO
          ELSEIF( MOISCODE.EQ.2 )THEN
           IF( TRANSIENT )THEN
            DO i = 1,NNODES,(PNODES-1)
                  READ(12,*,err=999)junk,SOILTYPE(i),YCORD(i),SUCNOD(i),TEM(i)
            ENDDO
           ELSE ! Initial conditions not required for steady state
            DO i = 1,NNODES,(PNODES-1)
                  READ(12,*,err=999)junk,SOILTYPE(i),YCORD(i)
                  SUCNOD(i) = 0.0
                  TEM   (i) = 20.0
            ENDDO
           ENDIF
          ELSE
            WRITE(*,*) 'Invalid Initial Moisture Condition Specifier',
     1      ' in the Mesh Data File'
          ENDIF
          IF( PNODES.EQ.3 )THEN ! Quadratic Element
           DO i = 2,NNODES,2  ! Insert Middle Nodes
                  SOILTYPE(i) = SOILTYPE(i+1)
                  YCORD   (i) = ( YCORD (i-1) + YCORD (i+1) )/2.0
```

```
            SUCNOD  (i) = ( SUCNOD(i-1) + SUCNOD(i+1) )/2.0
            WTWC    (i) = ( WTWC  (i-1) + WTWC  (i+1) )/2.0
            TEM     (i) = ( TEM   (i-1) + TEM   (i+1) )/2.0
        ENDDO
      ENDIF
      DO i = 1,NNODES
       IF(SOILTYPE(i).GT.MAX_TYPES)THEN
         WRITE(*,*) 'Invalid Soil Type at Node ',i,', in Mesh Input'
         stop
       ENDIF
      ENDDO
C ==================================================================
      RETURN
999   WRITE(*,*) aline
      STOP 'Error in Mesh data file'
      END
```

```
C ****************************************************************
C *                                                              *
C *          VEGETATION INPUT ROUTINE                            *
C *                                                              *
C ****************************************************************

C ==================================================================
      SUBROUTINE VEGETATION_INPUT
C ==================================================================
C   This subroutine reads all the supplied vegetation input data
C   Subroutines called:
C     Err_Msg: checks that the data falls within array bounds
C ==================================================================
      IMPLICIT NONE               ! Ensure that all variables have been correctly defined
      INCLUDE 'DECLARE.FT'          ! Contains all common block declarations
C ==================================================================
      CHARACTER*80  aline          ! Used to skip over file comments
      INTEGER    i                 ! Loop Counter
C ==================================================================
      aline = ' '
      READ(10,FMT='(A80)',ERR=999)aline  ! "Vegatation Input File for SoilCover"
      READ(10,FMT='(A80)',ERR=999)aline  ! "************************************"
      READ(10,FMT='(A80)',ERR=999)aline  ! A Blank Line
C ------------------------------------------
      READ(10,FMT='(A80)',ERR=999)aline ! "moisture    Moisture"
      READ(10,FMT='(A80)',ERR=999)aline ! "LimitingPt  WiltingPt"
      READ(10,FMT='(A80)',ERR=999)aline ! "  (kPa)       (kPa)"
      READ(10,*,ERR=999)LimitingPt,WiltingPt
      READ(10,FMT='(A80)',ERR=999)aline  ! A Blank Line
C ------------------------------------------
      READ(10,FMT='(A80)',ERR=999)aline ! "Green Leaf Area Index"
      READ(10,FMT='(A80)',ERR=999)aline ! "---------------------"
      READ(10,FMT='(A80)',ERR=999)aline ! "NumberOf"
      READ(10,FMT='(A80)',ERR=999)aline ! "DataPnts"
      READ(10,*,ERR=999)POINTS5(1)
      READ(10,FMT='(A80)',ERR=999)aline ! "DAY   LAI.."
      DO i=1, POINTS5(1)
        READ(10,*,ERR=999)XLAIDAY(i,1),XLAI(i,1)
      ENDDO
      READ(10,FMT='(A80)',ERR=999)aline  ! A Blank Line
C ------------------------------------------
      READ(10,FMT='(A80)',ERR=999)aline ! "Mulch Leaf Area Index"
      READ(10,FMT='(A80)',ERR=999)aline ! "---------------------"
      READ(10,FMT='(A80)',ERR=999)aline ! "NumberOf"
      READ(10,FMT='(A80)',ERR=999)aline ! "DataPnts"
      READ(10,*,ERR=999)POINTS6(1)
      READ(10,FMT='(A80)',ERR=999)aline ! "DAY   LAI.."
      DO i=1, POINTS6(1)
        READ(10,*,ERR=999)XMULCHDAY(i,1),XMULCH(i,1)
```

```
      ENDDO
      READ(10,FMT='(A80)',ERR=999)aline  ! A Blank Line
      READ(10,FMT='(A80)',ERR=999)aline  ! "Daily Root Depth Data"
      READ(10,FMT='(A80)',ERR=999)aline  ! "----------------"
      READ(10,FMT='(A80)',ERR=999)aline  ! "Day  TopNode  BottomNode"
C ================================================================
      RETURN
999  WRITE(*,*) aline
      STOP 'Error in Vegetation data file'
      END
```

```
C ******************************************************************
C *                                     *
C *          FREEZE/THAW INPUT ROUTINE            *
C *                                     *
C ******************************************************************

C ================================================================
      SUBROUTINE ICE_INPUT
C ================================================================
C    This subroutine reads all the supplied ice input data
C    Subroutines called:
C      Err_Msg: checks that the data falls within array bounds
C ================================================================
      IMPLICIT NONE                ! Ensure that all variables have been correctly defined
      INCLUDE 'DECLARE.FI'            ! Contains all common block declarations
C ================================================================
      CHARACTER*80  aline            ! Used to skip over file comments
      REAL      xuwc
      INTEGER    i, j             ! Loop Counter
      INTEGER    junk,type
C ================================================================
      READ(14,FMT='(A80)',ERR=999)aline   ! "Freeze/Thaw Input File for SoilCover"
      READ(14,FMT='(A80)',ERR=999)aline   ! "************************************"
      READ(14,FMT='(A80)',ERR=999)aline   ! A Blank Line
C ----------------------------------------------------------
C ----------------------------------------------------------
      READ(14,FMT='(A80)',ERR=999)aline   ! "Density of Ice..."
      READ(14,*,ERR=999)RHOICE
C ----------------------------------------------------------
      READ(14,FMT='(A80)',ERR=999)aline   ! " Latent heat of Fusion..."
      READ(14,*,ERR=999)FLATENT
C ----------------------------------------------------------
C ----------------------------------------------------------
C ----------------------------------------------------------
      READ(14,FMT='(A80)',ERR=999)aline  ! "Node   Initial vol Ice.Content (dec)"
      DO i=1,NNODES
       READ(14,*,ERR=999)junk,NODVOLICE(i)
c    type=SOILTYPE(i)
c     NODVOLICE(i)=NODVOLICE(i)*GS(type)*(1.0e0-PORS(type))
       oldnodvolice(i)=nodvolice(i)
      ENDDO
C ----------------------------------------------------------
      DO type = 1,MAX_TYPES
       READ(14,FMT='(A80)',ERR=999)aline   ! A Blank Line
       READ(14,FMT='(A80)',ERR=999)aline   ! "Soil Type #"
       READ(14,FMT='(A80)',ERR=999)aline   ! "NUMBER OF DATA POINTS IN ..."
       READ(14,*,ERR=999)POINTS7(type)
       points8(type)=points7(type)
       READ(14,FMT='(A80)',ERR=999)aline   ! "GRAV W/C...NEGATIVE TEMP "
        DO j=1,POINTS7(type)
            READ(14,*,ERR=999)xuwc,XTEM(j,type)
            if(XTEM(j,type).LT.0.) XTEM(j,type)=0.-XTEM(j,type)
            XVOLUWC(j,type)=xuwc*GS(type)*(1.0E0-PORS(type))
            ytem(points7(type)+1-j,type)=xtem(j,type)
            yvoluwc(points7(type)+1-j,type)=xvoluwc(j,type)
```

```
          ENDDO
          READ(14,FMT='(A80)',ERR=999)aline   ! A Blank Line

          READ(14,FMT='(A80)',ERR=999)aline   ! "NUMBER OF DATA POINTS IN ..."
          READ(14,*,ERR=999)POINTS9(type)

          READ(14,FMT='(A80)',ERR=999)aline   ! "vol W/C....dsuc / dtem "
           DO j=1,POINTS9(type)
                  READ(14,*,ERR=999)GVOLUWC(j,type),XGG(j,type)
           ENDDO

         ENDDO
         RETURN
999  WRITE(*,*) aline
         STOP 'Error in freeze/thaw data file'
         END
C  -----------------------------------------------------------
```

```
C  ****************************************************************
C  *                                                              *
C  *      INITIAL DAILY DATA INPUT ROUTINE                        *
C  *                                                              *
C  ****************************************************************

C  ================================================================
```

SUBROUTINE INITIAL_DAILY_INPUT

```
C  ================================================================
C    This subroutine reads the climate and boundary condition data
C    on a daily basis.
C     Subroutines called:
C       Err_Msg: checks that the data falls within array bounds
C  ================================================================
         IMPLICIT NONE                 ! Ensure that all variables have been correctly defined
         INCLUDE 'FUNCTION.FT'            ! Contains all function declarations
         INCLUDE 'DECLARE.FT'             ! Contains all common block declarations
C  ================================================================
         CHARACTER*80  aline              ! Used to skip over file comments
         integer     temperature_code     ! Used to read if surface temperatures are specified
C  ================================================================
         READ(12,FMT='(A80)',ERR=998)aline   ! "Daily Data Input File For"
         READ(12,FMT='(A80)',ERR=998)aline   ! "*************************"
         READ(12,FMT='(A80)',ERR=998)aline   ! A Blank Line
C  ------------------------------------------
         READ(12,FMT='(A80)',ERR=998)aline   ! "Should SoilCover Use Spec"
         READ(12,*,ERR=998)temperature_code
         CALL ERR_MSG('Surface Temperature Code',temperature_code,1)
         IF( temperature_code.EQ.1 )THEN
           IF( TRANSIENT )THEN
                  CALCULATE_TEMPS = TRUE
           ELSE
                  STOP 'Surface temp must be specified for Steady State Analysis'
           ENDIF
         ENDIF
         READ(12,FMT='(A80)',ERR=998)aline   ! A Blank Line
C  ------------------------------------------
         READ(12,FMT='(A80)',ERR=998)aline   ! "Total  Temp   RelHum  Lat"
         READ(12,FMT='(A80)',ERR=998)aline   ! "DaysData Lag    Lag"
         IF( TRANSIENT )THEN
           READ(12,*,ERR=998)DAYS,Temperature_Lag,Rh_Lag,LAT,NSTART
         ELSE
           READ(12,FMT='(A80)',ERR=998)aline   ! Values are not used
           DAYS         = 1
           Temperature_Lag = 0.0
           Rh_Lag       = 0.0
           LAT          = 0.0
           NSTART       = 0
         ENDIF
```

```
      CALL ERR_MSG('DAYS',DAYS,256000)
      CALL ERR_MSG('Days Past Jan.',NSTART,365)
      READ(12,FMT='(A80)',ERR=998)aline   ! A Blank Line
      READ(12,FMT='(A80)',ERR=998)aline   ! "Daily Data"
      READ(12,FMT='(A80)',ERR=998)aline   ! "-------------------"
      READ(12,FMT='(A80)',ERR=998)aline   ! Headings Row #1 Line
      READ(12,FMT='(A80)',ERR=998)aline   ! Headings Row #2 Line
      READ(12,FMT='(A80)',ERR=998)aline   ! Headings Unit Line
C ===================================================================
      RETURN
998   STOP 'Error in initial part of daily input data file'
      END
```

```
C *****************************************************************
C *                                              *
C *            DAILY DATA INPUT ROUTINE                    *
C *                                              *
C *****************************************************************

C ===================================================================
      SUBROUTINE DAILY_INPUT
C ===================================================================
C   This subroutine reads the climate and boundary condition data
C   on a daily basis.
C    Subroutines called:
C       Err_Msg: checks that the data falls within array bounds
C ===================================================================
      IMPLICIT NONE             ! Ensure that all variables have been correctly defined
      INCLUDE 'FUNCTION.FT'        ! Contains all function declarations
      INCLUDE 'DECLARE.FT'         ! Contains all common block declarations
C ===================================================================
      integer    bot_type        ! specifies whether top node has vol. wc head input
      real       bot_value
      INTEGER    i               ! Loop Counter
      INTEGER    j               ! Loop Counter
      INTEGER    junk              ! Used to skip over integer in input file
      integer    top_type        ! specifies whether bottom head boundary input as vol. wc
      real       top_value       ! the actual boundary condition
C ===================================================================
      DO i = 1,2
        NODEB  (1,i) = 1
        NODEB  (2,i) = NNODES
        NODEN  (1,i) = 1
        NODEN  (2,i) = NNODES
        TEMPAMAX (i) = TEMPAMAX (i+1)
        TEMPAMIN (i) = TEMPAMIN (i+1)
        SOLAR   (i) = SOLAR   (i+1)
        RH_MAX  (i) = RH_MAX  (i+1)
        RH_MIN  (i) = RH_MIN  (i+1)
        WIND    (i) = WIND    (i+1)
        DURATION (i) = DURATION (i+1)
        RootTop  (i) = RootTop  (i+1)
        RootDepth(i) = RootDepth(i+1)
        EBW   (1,i) = EBW   (1,i+1)
        EBW   (2,i) = EBW   (2,i+1)
        EBH   (1,i) = EBH   (1,i+1)
        EBH   (2,i) = EBH   (2,i+1)
        QW    (1,i) = QW    (1,i+1)
        QW    (2,i) = QW    (2,i+1)
        VFLUXPAN (i) = VFLUXPAN (i+1)
      ENDDO
C ---------------------------------------------
      IF( NDAY.LT.DAYS )THEN ! If there is more data to read
        READ(12,*,ERR=999)junk,TEMPAMAX(3),TEMPAMIN(3),SOLAR(3),
     1  RH_MAX(3),RH_MIN(3),WIND(3),top_type,top_value,DURATION(3),
     1  bot_type,bot_value,EBH(1,3),EBH(2,3),nexttoptemp ! nexttoptemp is next days user defined surface temp.
        IF( CALCULATE_TEMPS )THEN
```

```
            EBH(1,3) = TEMPAMIN(3)
      ENDIF
      IF(VEGETATION)THEN  ! If modelling vegetation, readin next days values"
            READ(10,*,ERR=998)junk,RootTop(3),RootDepth(3)
      ELSE
            RootTop  (3) = 1
            RootDepth(3) = 0
      ENDIF
C -------------------------------------------
      TOP_MOIS_BNDRY(3) = FALSE
      EBW      (1,3) = 1.00E+10
      QW       (1,3) = 1.00E+20
      BCOEF      (3) = 0.0
      VFLUXPAN   (3) = 2.0
      IF(top_type.EQ.0)THEN           ! Pressure Head Boundary Condition
       EBW(1,3)       = top_value
      ELSEIF(top_type.EQ.1)THEN       ! Gravimetric Water Content BC
       TOP_MOIS_BNDRY(3) = TRUE
       EBW(1,3)       = top_value * GS(1) * (1.0E0 - PORS(1))
       CALL VWC_TO_HEAD
      ELSEIF(top_type.EQ.2)THEN       ! Volumetric Water Content BC
       TOP_MOIS_BNDRY(3) = TRUE
       EBW(1,3)       = top_value
       CALL VWC_TO_HEAD
      ELSEIF(top_type.EQ.3)THEN       ! Flux Boundary Condition
       IF(top_value.NE.1.00E+20)THEN
             QW      (1,3) = top_value/86400000.0 ! change units from mm/day to m/sec"
       ENDIF
      ELSEIF(top_type.EQ.4.AND.DFLUX)THEN  ! Potential Evaporation for DFLUX
       VFLUXPAN    (3) = -top_value/86400.0 ! change units from mm/day to mm/sec
      ELSEIF(top_type.EQ.5.AND.DFLUX)THEN  ! Potential Evaporation for DFLUX
       BCOEF      (3) = top_value
      ELSE
       WRITE(*,*)' Bad Top Boundary Condition on Day',NDAY+1
      ENDIF
C   -------------------------------------------
      IF(bot_type.EQ.0)THEN           ! Pressure Head Boundary Condition
       BOT_MOIS_BNDRY(3) = FALSE
       EBW(2,3)       = bot_value
       QW(2,3)        = 1.00E+20
      ELSEIF(bot_type.EQ.1)THEN       ! Gravimetric Water Content BC
       BOT_MOIS_BNDRY(3) = TRUE
       EBW(2,3)       = bot_value * GS(SOILTYPE(NNODES))
     1              * (1.0E0-PORS(SOILTYPE(NNODES)))
       CALL VWC_TO_HEAD
       QW(2,3)        = 1.00E+20
      ELSEIF(bot_type.EQ.2)THEN       ! Volumetric Water Content BC
       BOT_MOIS_BNDRY(3) = TRUE
       EBW(2,3)       = bot_value
       CALL VWC_TO_HEAD
       QW(2,3)        = 1.00E+20
      ELSEIF(bot_type.EQ.3)THEN       ! Flux Boundary Condition
       BOT_MOIS_BNDRY(3) = FALSE
       EBW(2,3)       = 1.00E+10
       IF(QW(2,3).NE.1.00E+20)THEN
             QW(2,3)       = bot_value/86400000.0 ! change units from mm/day to m/sec"
       ENDIF
      ELSE
       WRITE(*,*)' Bad Bottom Boundary Condition on Day',NDAY+1
      ENDIF
     ENDIF
C ==========================================================
     IF( NDAY.EQ.0 )THEN
      j = 1
     ELSEIF(NDAY.EQ.DAYS)THEN
      j = 2
     ELSE
      j = 3
     ENDIF
     DO i = 2,j,-1
            NODEB  (1,i) = 1
```

```
                NODEB  (2,i) = NNODES
                NODEN  (1,i) = 1
                NODEN  (2,i) = NNODES
                TEMPAMAX (i) = TEMPAMAX (3)
                TEMPAMIN (i) = TEMPAMIN (3)
                SOLAR   (i) = SOLAR   (3)
                RH_MAX  (i) = RH_MAX  (3)
                RH_MIN  (i) = RH_MIN  (3)
                WIND    (i) = WIND    (3)
                DURATION (i) = DURATION (3)
                RootTop  (i) = RootTop  (3)
                RootDepth(i) = RootDepth(3)
                EBW   (1,i) = EBW   (1,3)
                EBW   (2,i) = EBW   (2,3)
                EBH   (1,i) = EBH   (1,3)
                EBH   (2,i) = EBH   (2,3)
                QW    (1,i) = QW    (1,3)
                QW    (2,i) = QW    (2,3)
                VFLUXPAN (i) = VFLUXPAN (3)
      ENDDO
C ===========================================================
      RETURN
998   STOP 'Error in daily Root Depth Data File'
999   STOP 'Error in daily input data file'
      END
```

```
C ******************************************************************
C *                                            *
C *          CONSTANTS INPUT ROUTINE                    *
C *                                            *
C ******************************************************************

C ===========================================================
      SUBROUTINE CONSTANT_INPUT
C ===========================================================
C   This subroutine reads all the input data supplied in the input
C   data file.
C   Subroutines called:
C      Err_Msg: checks that the data falls within array bounds
C      Gauss_Data: reads the gauss weight and location data files.
C ===========================================================
      IMPLICIT NONE              ! Ensure that all variables have been correctly defined
      INCLUDE 'DECLARE.FT'          ! Contains all common block declarations
C ===========================================================
      CHARACTER*80 aline             ! Used to skip over file comments
C ===========================================================
      READ(12,FMT='(A80)',ERR=999)aline  ! "Constants Input File for SoilCover"
      READ(12,FMT='(A80)',ERR=999)aline  ! "****************************"
      READ(12,FMT='(A80)',ERR=999)aline  ! A Blank Line
C -----------------------------------------
      READ(12,FMT='(A80)',ERR=999)aline  ! "Acceleration  Density  Latent"
      READ(12,FMT='(A80)',ERR=999)aline  ! "  Due To      of   Heat of"
      READ(12,FMT='(A80)',ERR=999)aline  ! " Gravity     Water  Vaporization"
      READ(12,FMT='(A80)',ERR=999)aline  ! "  (m/s)     (Kg/m^3)  (J/Kg)"
      READ(12,*,ERR=999)GRAV,RHOWAT,RLATENT
      READ(12,FMT='(A80)',ERR=999)aline  ! A Blank Line
C -----------------------------------------
      READ(12,FMT='(A80)',ERR=999)aline  ! "Gauss Pt..."
      READ(12,FMT='(A80)',ERR=999)aline  ! Reading in entire line
      GaussLcFile = aline(1:10)          ! assign file name
      READ(12,FMT='(A80)',ERR=999)aline  ! Reading in entire line
      GaussWtFile = aline(1:10)          ! assign file name
C ===========================================================
      RETURN
999   WRITE(*,*) aline
      STOP 'Error in Constants data file'
```

```
      END
```

```
C ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C This routine outputs an error message if the value read is larger than
C the limit set by array bounds
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      SUBROUTINE ERR_MSG(Message,Value_found,Max_value)
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      CHARACTER*(*) Message
      INTEGER    Value_found,Max_value
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      IF( Value_found.GT.Max_value )THEN
            WRITE(*,20)Message,Value_found,Max_value
20    FORMAT(' ',A20,' was read as ',I3,' LIMIT is ',I3)
            STOP
      ENDIF
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      RETURN
      END
```

SUBROUTINE RELAXATION

```
C  ====================================================================
C    This subroutine implements a relaxation scheme which is meant to
C   help the 'ITERATE' subroutine to achieve convergence more rapidly.
C  ====================================================================
   IMPLICIT NONE
   INCLUDE 'FUNCTION.FT'          ! Contains all function declarations
   INCLUDE 'DECLARE.FT'           ! Contains all common block declarations
C  ====================================================================
   INTEGER i,j                    ! Loop counters
C  -----------------------------------------
   DO i = 1,NNODES
     SUCNOD(i) = SUCNOD(i) + SUC_DAMP*( PRESNOD(i)-SUCNOD(i) )
     TEM(i)  = TEM  (i) + TEM_DAMP*( PRETNOD(i)-TEM  (i) )
   ENDDO
C  ====================================================================
   RETURN
   END
```

SUBROUTINE REVERSE_SPLINES ! suction vs water content

```
C  ====================================================================
C    This subroutine swaps the dependant and independant variables
C   of a calculated spline and modifies the spline weights to
C   accomodate this change.
C  ====================================================================
   IMPLICIT NONE
   INCLUDE 'FUNCTION.FT'          ! Contains all function declarations
   INCLUDE 'DECLARE.FT'           ! Contains all common block declarations
C  -----------------------------------------
   REAL     data1          ! Temporary Var to get initial nodal suctions
   REAL     data2          ! Temporary Var to get initial nodal suctions
   INTEGER  i,j            ! Loop counters
   INTEGER  soil           ! The current soil type
C  ====================================================================
   do soil=1,MAX_TYPES
    do i = 1,POINTS1(soil)/2      ! Reverse the splines order so the it is in ascending volwc order
     j = POINTS1(soil) - i + 1
     data1       = exp(XVOLWC(i,soil))
     data2       = exp(XSUC  (i,soil))
     XVOLWC (i,soil) = exp(XVOLWC(j,soil))
     XSUC   (i,soil) = exp(XSUC  (j,soil))
     XVOLWC (j,soil) = data1
```

```
      XSUC    (j,soil) = data2
    enddo
    if( (POINTS1(soil)/2) .EQ. (POINTS1(soil)-1)/2 )then
      j = POINTS1(soil)/2 + 1
      XVOLWC  (j,soil) = exp(XVOLWC(j,soil))
      XSUC    (j,soil) = exp(XSUC  (j,soil))
    endif
  enddo
C =================================================================
  return
  end
```

SUBROUTINE REVERSE_SPLINES2   ! temperature vs water content

```
C =================================================================
C    This subroutine swaps the dependant and independant variables
C  of a calculated spline and modifies the spline weights to
C  accomodate this change.
C =================================================================
  IMPLICIT NONE
  INCLUDE 'FUNCTION.FT'          ! Contains all function declarations
  INCLUDE 'DECLARE.FT'           ! Contains all common block declarations
C ---------------------------------------------------------
  REAL      data1            ! Temporary Var to get initial nodal suctions
  REAL      data2            ! Temporary Var to get initial nodal suctions
  INTEGER   i,j              ! Loop counters
  INTEGER   soil             ! The current soil type
C =================================================================
  do soil=1,MAX_TYPES
    do i = 1,POINTS7(soil)/2        ! Reverse the splines order so the it is in ascending volwc order
      j = POINTS7(soil) - i + 1
      data1        = exp(XVOLUWC(i,soil))
      data2        = exp(XTEM  (i,soil))
      XVOLUWC  (i,soil) = exp(XVOLUWC(j,soil))
      XTEM   (i,soil) = exp(XTEM  (j,soil))
      XVOLUWC  (j,soil) = data1
      XTEM   (j,soil) = data2
    enddo
    if( (POINTS7(soil)/2) .EQ. (POINTS7(soil)-1)/2 )then
      j = POINTS7(soil)/2 + 1
      XVOLUWC  (j,soil) = exp(XVOLUWC(j,soil))
      XTEM   (j,soil) = exp(XTEM  (j,soil))
    endif
  enddo
C =================================================================
  return
  end
```

SUBROUTINE SET_INITIAL_SETTINGS

```
C =================================================================
C    This subroutine calculates the air temperature, the air relative
C  humidity, the slope of the saturated vapour pressure function,
C  and the net solar radiation, and saves the starting suctions and
C  temperatures for the current time step.
C    Subroutines called: LeafAreaIndex  ! Calculates daily green LAI
C =================================================================
  IMPLICIT NONE                  ! Ensure that all variables have been correctly defined
  INCLUDE 'FUNCTION.FT'            ! Contains all function declarations
  INCLUDE 'DECLARE.FT'             ! Contains all common block declarations
C =================================================================
  REAL   curTime           ! Current Time
  INTEGER i,j,k,soil       ! Loop Counters
  REAL   limitFactor       ! Plant Limiting Factor
  REAL   dayleng           ! Daylight hours in the current day(hrs)
C ---------------------------------------------
  curTime = TTIME
  CALL LeafAreaIndex                 ! Calculates the daily green LAI
  dayleng = Calc_dayleng(NSTART+NDAY-1)        ! Calculate the length of the current day
```

```
      TEMPAIR = Calc_AIRTemp(dayleng,curTime)          ! (K) AIR TEMP
      RHAIR1  = Calc_AirRH(dayleng,curTime)           ! Relative Humidity of the air
      SLPOT1  = 0.1*(0.00815*(TEMPAIR-273.0) + 0.8912)**7  ! this is a emperical method to calculate the slope of the satvp - temp
curve (Kpa/cel)
      PVAIR1  = Calc_SatVp(TEMPAIR)*RHAIR1/10.0          ! Vapour Pressure of the air
      QSTAR   = Calc_NETRAD(dayleng)                ! Net solar radiation in mm/day
C  ==========================================================================
C   FORMING THE NODAL VECTOR OF KNOWNS PHIA or {x} at t
C  ==========================================================================
      DO i = 1,NNODES
       j = i + NNODES
       PHIA(i) = SUCNOD(i)     ! {  Suction  } @ t
       PHIA(j) = TEM  (i)      ! {Temperature} @ t
      ENDDO

c  -----------------------------------------
      RETURN
      END
```

SUBROUTINE SET_INIT_SUCTION

```
C  ==========================================================================
C     This subroutine determines the initial suctions, water contents,
C   and air entry value based on the initial input conditions.
C     Subroutines called:
C        reverse_splines: swaps the dependant and independant variables
C           and modifies the spline weights to accomodate this change.
C  ==========================================================================
      IMPLICIT NONE
      INCLUDE 'FUNCTION.FT'          ! Contains all function declarations
      INCLUDE 'DECLARE.FT'           ! Contains all common block declarations
C  -----------------------------------------
      INTEGER  i               ! Loop counters
      integer  Soil            ! The current soil type
      REAL     tx              ! Temporary X variable
      REAL     t2              ! Temporary variable

C  ==========================================================================
C           Calculate initial nodal suctions
C  ==========================================================================
      IF(MOISCODE.EQ.0) THEN  ! If Initial Nodal Water Contents were not specified
        DO i = 1,NNODES
          soil   = SOILTYPE(i)
          SUCNOD(i)= YCORD(i)*RHOWAT*GRAV/100.0E0

          WTWC(i)  = Calc_VolWc( soil,SUCNOD(i))
     1     /GS(soil) /(1.0E0-PORS(soil))
        ENDDO
C  -----------------------------------------
      ELSEIF(MOISCODE.EQ.1) THEN       ! Initial water contents were specified

        CALL REVERSE_SPLINES ! Reverse the splines order so the it is in ascending suction order
        DO i=1,MAX_TYPES
         CALL WtSplin2(i,POINTS1,XVOLWC,XSUC,SPLINSL1)
        ENDDO
        DO i = 1,NNODES
         soil = SOILTYPE(i)
         t2   = exp( XVOLWC(POINTS1(soil),soil) )
         tx   = WTWC(i)*GS(soil)*(1.0e0-PORS(soil))
         IF( tx.lt.t2 ) THEN
          SUCNOD(i) = FN_POINT(soil,POINTS1,XVOLWC,XSUC,SPLINSL1,tx)
         ELSE
          SUCNOD(i) = (SUCT_INT(soil)-tx)/RM2WA(soil)
         ENDIF
        ENDDO  ! i = 1,nnodes
        CALL REVERSE_SPLINES ! Re-Establish the original spline order
        DO i=1,MAX_TYPES
         CALL WtSplin2(i,POINTS1,XSUC,XVOLWC,SPLINSL1)
        ENDDO
C  -----------------------------------------
```

```
      ELSEIF( MOISCODE.EQ.2 ) THEN          ! Initial Pressures were specified

        DO i = 1,NNODES
          soil   = SOILTYPE(i)
          WTWC(i) = Calc_VolWc(soil,SUCNOD(i))
     1              /GS(soil)/(1.0E0-PORS(soil))
        ENDDO
C -----------------------------------------------
      ELSEIF( MOISCODE.EQ.3) THEN           ! Initial Volumetric Water Contents were specified

        CALL REVERSE_SPLINES ! Reverse the splines order so the it is in ascending suction order
        DO i=1,MAX_TYPES
          CALL WtSplin2(i,POINTS1,XVOLWC,XSUC,SPLINSL1)
        ENDDO
        DO i = 1,NNODES
          soil = SOILTYPE(i)
          t2  = exp( XVOLWC(POINTS1(soil),soil) )
          tx = WTWC(i)  ! These values are VolWc's already
          IF( tx.LT.t2 ) THEN
            SUCNOD(i)=FN_POINT(soil,POINTS1,XVOLWC,XSUC,SPLINSL1,tx)
          ELSE
            SUCNOD(i)=(SUCT_INT(soil)-tx)/RM2WA(soil)
          ENDIF
          WTWC(i) = WTWC(i)/(GS(soil)*(1.0E0-PORS(soil)))  ! Convert VolWc to GravWC
        enddo  ! i = 1,nnodes
        CALL REVERSE_SPLINES ! Re-Establish the original spline order
        DO i=1,MAX_TYPES
          CALL WtSplin2(i,POINTS1,XSUC,XVOLWC,SPLINSL1)
        ENDDO
      ENDIF  ! IF(MOISCODE...
C -----------------------------------------------
      RETURN
      END
```

```
C   This function changes the head boundary condition of the bottom node
C   of the top node from a water content to matric suction.
C   ==================================================================
```

SUBROUTINE VWC_TO_HEAD

```
C   ==================================================================
      IMPLICIT NONE                 ! ENSURE ALL VARIABLES HAVE BEEN CORRECTLY DEFINED
      INCLUDE 'DECLARE.FT'
      INCLUDE 'FUNCTION.FT'
C   ==================================================================
      INTEGER   i                   ! Loop counter
      REAL      tx                  ! Temporary X variable
      REAL      t2                  ! Temporary variable
C   ==================================================================
      IF(TOP_MOIS_BNDRY(3).OR.BOT_MOIS_BNDRY(3))THEN
       CALL REVERSE_SPLINES ! Reverse the splines order so the it is in ascending suction order
       DO i=1,MAX_TYPES
        CALL WtSplin2(i,POINTS1,XVOLWC,XSUC,SPLINSL1)
       ENDDO
       IF(TOP_MOIS_BNDRY(3).AND.(EBW(1,3).NE.1E10))THEN
         t2 = exp( xvolwc(POINTS1(SOILTYPE(1)),SOILTYPE(1)) )
         tx = EBW(1,3)
         if( tx.lt.t2 ) then
          EBW(1,3) = -1 * fn_point(SOILTYPE(1),POINTS1,xvolwc,xsuc,
     1                SPLINSL1,tx)
         else
          EBW(1,3) = -1*(suct_int(SOILTYPE(1))-tx)/RM2WA(SOILTYPE(1))
         endif
       ENDIF
       IF(BOT_MOIS_BNDRY(3).AND.(EBW(2,3).NE.1E10))THEN
         t2 = exp(xvolwc(POINTS1(SOILTYPE(NNODES)),SOILTYPE(NNODES)))
         tx = EBW(2,3)
         if( tx.lt.t2 ) then
```

```
      EBW(2,3) = -1 * fn_point(SOILTYPE(NNODES),POINTS1,xvolwc,xsuc,
   1                  SPLINSL1,tx)
     else
      EBW(2,3) = -1 * (suct_int(SOILTYPE(NNODES))-tx)
   1               /RM2WA(SOILTYPE(NNODES))
     endif
    ENDIF
    CALL REVERSE_SPLINES ! Re-Establish the original spline order
    DO i=1,MAX_TYPES
     CALL WtSplin2(i,POINTS1,XSUC,XVOLWC,SPLINSL1)
    ENDDO
   ENDIF
C  ===========================================================================
   RETURN
   END
```

```
C  This subroutine smoothes user supplied points.
C  ===================================================================
```

SUBROUTINE Smooth(soil,pnts,X,Y,order,times,type)

```
C  ===================================================================
   IMPLICIT NONE                 ! Ensure that all variables have been correctly defined
   INCLUDE 'CONSTANT.FT'
C  ===================================================================
   integer i,j,t                 ! Loop counters
   integer soil                   ! The current layer
   integer order                  ! The order of smoothing required
   INTEGER pnts   (MAX_TYPES)          ! Number of Data Pts. in VolWc vs Suction
   integer times                  ! Number of times to smooth the data
   integer type                   ! Type of Smoothing
   REAL   X    (MAX_POINTS,MAX_TYPES) ! Suction Data for Suction vs. Perm.
   REAL   Y    (MAX_POINTS,MAX_TYPES) ! Volumetric Water Content for Suct. vs WC.
   real   ty(max_points)
C  ===================================================================
   if(type.eq.semi_log)then
    do i=1,pnts(soil)
     X(i,soil) = log(X(i,soil)) ! Smooth with Logarithmic X scale
    enddo
   else if(type.eq.logarithmic)then
    do i=1,pnts(soil)
     X(i,soil) = log(X(i,soil)) ! Smooth with Logarithmic X scale
     Y(i,soil) = log(Y(i,soil)) ! Smooth with Logarithmic Y scale
    enddo
   endif
C  ===================================================================
   do t=1,times
c  ------------------------------------------
    do i=2,order
     ty(i) = Y(i,soil)
     do j=1,i-1
      ty(i) = ty(i) + Y(i-j,soil) + (X(i,soil)-X(i-j,soil))
   1        *(Y(i+j,soil)-Y(i-j,soil))/(X(i+j,soil)-X(i-j,soil))
     enddo
     ty(i) = ty(i)/i
    enddo
c  ------------------------------------------
    do i=(order+1),pnts(soil)-order
     ty(i) = Y(i,soil)
     do j=1,order
      ty(i) = ty(i) + Y(i-j,soil) + (X(i,soil)-X(i-j,soil))
   1        *(Y(i+j,soil)-Y(i-j,soil))/(X(i+j,soil)-X(i-j,soil))
     enddo
     ty(i) = ty(i)/(order+1)
    enddo
c  ------------------------------------------
    do i=pnts(soil)-order+1,pnts(soil)-1
     ty(i) = Y(i,soil)
     do j=1,pnts(soil)-i
      ty(i) = ty(i) + Y(i-j,soil) + (X(i,soil)-X(i-j,soil))
```

```
  1        *(Y(i+j,soil)-Y(i-j,soil))/(X(i+j,soil)-X(i-j,soil))
     enddo
     ty(i) = ty(i)/(pnts(soil)-i+1)
    enddo
c  ------------------------------------------------
    do i=2,pnts(soil)-1
     Y(i,soil) = ty(i)
    enddo
   enddo
C  ================================================================
   if(type.eq.semi_log)then
    do i=1,pnts(soil)
     X(i,soil) = exp(X(i,soil))
    enddo
   else if(type.eq.logarithmic)then
    do i=1,pnts(soil)
     X(i,soil) = exp(X(i,soil))
     Y(i,soil) = exp(Y(i,soil))
    enddo
   endif
C  ================================================================
   RETURN
   END
```

```
C  ================================================================
    SUBROUTINE SPLINES(Debug_Splines)
C  ================================================================
C    This subroutine splines, smooths, graphs, and stores the soil
C   property data.
C    Subroutines called:
C      wtsplin2: calculates the spline weights
C      smooth: smooths the spline data
C      graph: graphs the splines to the display
C      writesplines: writes the splined data to a data file.
C  ================================================================
    IMPLICIT NONE                  ! Ensure that all variables have been correctly defined
    INCLUDE 'FUNCTION.FT'            ! Contains all function declarations
    INCLUDE 'DECLARE.FT'             ! Contains all common block declarations
C  ================================================================
    CHARACTER*80 aline               ! Used to skip over file comments
    LOGICAL Debug_Splines            ! Flag to indicate splines are to be graphed to screen
    INTEGER i,j,show                 ! Loop Counters
    integer nskip                ! Number of layers which are not being used
    integer order1  (MAX_TYPES)      ! Order for Smoothing Corresponding Curve
    integer order2  (MAX_TYPES)      ! Order for Smoothing Corresponding Curve
    integer order3  (MAX_TYPES)      ! Order for Smoothing Corresponding Curve
    integer order4  (MAX_TYPES)      ! Order for Smoothing Corresponding Curve
    integer order5  (MAX_TYPES)      ! Order for Smoothing Corresponding Curve
    integer order6  (MAX_TYPES)      ! Order for smoothing corresponding curve
    integer order7  (MAX_TYPES)      ! Order for smoothing corresponding curve
    integer order8  (max_types)
    integer order9  (max_types)
    INTEGER times1  (MAX_TYPES)        ! Number of times to smooth corresponding curve
    INTEGER times2  (MAX_TYPES)        ! Number of times to smooth corresponding curve
    INTEGER times3  (MAX_TYPES)        ! Number of times to smooth corresponding curve
    INTEGER times4  (MAX_TYPES)        ! Number of times to smooth corresponding curve
    INTEGER times5  (MAX_TYPES)        ! Number of times to smooth corresponding curve
    INTEGER times6  (MAX_TYPES)        ! Number of times to smooth corresponding curve
    INTEGER times7  (MAX_TYPES)        ! Number of times to smooth corresponding curve
    integer times8  (max_types)
    integer times9  (max_types)
    REAL   x0    (MAX_POINTS,MAX_TYPES) ! Temporary Data Points
    REAL   y0    (MAX_POINTS,MAX_TYPES) ! Temporary Data Points
    REAL   z0    (MAX_POINTS,MAX_TYPES) ! Temporary Spline Wgts.

C  ================================================================
C              OPEN SPLINE DATA FILE
```

```
C ==============================================================
      OPEN(UNIT=14,FILE='SPLINE.DAT',STATUS='old')
C --------------------------------------------------------
C     Read in the data spline smoothing settings
C --------------------------------------------------------
      DO i=1,MAX_TYPES
       READ(14,FMT='(A80)',ERR=999)aline ! "-------------------------"
       READ(14,FMT='(A80)',ERR=999)aline ! "Spline Smoothing Settings"
       READ(14,FMT='(A80)',ERR=999)aline ! "-------------------------"
       READ(14,FMT='(A80)',ERR=999)aline ! "Suction vs Volumetric WC"
       READ(14,FMT='(A80)',ERR=999)aline ! "Order    #times"
       READ(14,*,ERR=999)order1(i),times1(i)
       READ(14,FMT='(A80)',ERR=999)aline ! "Suction vs Hydraulic Cond"
       READ(14,FMT='(A80)',ERR=999)aline ! "Order    #Times"
       READ(14,*,ERR=999)order2(i),times2(i)
       READ(14,FMT='(A80)',ERR=999)aline ! "WTWC vs Thermal Cond"
       READ(14,FMT='(A80)',ERR=999)aline ! "Order    #Times"
       READ(14,*,ERR=999)order3(i),times3(i)
       READ(14,FMT='(A80)',ERR=999)aline ! "WTWC vs Specific Heat"
       READ(14,FMT='(A80)',ERR=999)aline ! "Order    #Times"
       READ(14,*,ERR=999)order4(i),times4(i)
       READ(14,FMT='(A80)',ERR=999)aline ! "Unfrozen w/c vs Temperature"
       READ(14,FMT='(A80)',ERR=999)aline ! "Order    #Times"
       READ(14,*,ERR=999)order7(i),times7(i)
       order8(i)=order7(i)
       times8(i)=times7(i)
       order9(i)=order7(i)
       times9(i)=times7(i)
      ENDDO
      READ(14,FMT='(A80)',ERR=999)aline ! "-------------------------"
      READ(14,FMT='(A80)',ERR=999)aline ! "Spline Smoothing Settings(GREEN)"
      READ(14,FMT='(A80)',ERR=999)aline ! "-------------------------"
      READ(14,FMT='(A80)',ERR=999)aline ! "GREEN LAI vs day"
      READ(14,FMT='(A80)',ERR=999)aline ! "Order    #times"
      READ(14,*,ERR=999)order5(1),times5(1)
      READ(14,FMT='(A80)',ERR=999)aline ! "-------------------------"
      READ(14,FMT='(A80)',ERR=999)aline ! "Spline Smoothing Settings(MULCH)"
      READ(14,FMT='(A80)',ERR=999)aline ! "-------------------------"
      READ(14,FMT='(A80)',ERR=999)aline ! "MULCH LAI vs day"
      READ(14,FMT='(A80)',ERR=999)aline ! "Order    #times"
      READ(14,*,ERR=999)order6(1),times6(1)

C ==============================================================


C ==============================================================
C          SPLINING OF THE SUCTION VS WC DATA
C ==============================================================
      DO i=1,MAX_TYPES
       CALL WtSplin2(i,POINTS1,XSUC,XVOLWC,SPLINSL1)
      ENDDO
c --------------------------------------------------------
c                Smooth the curve
c --------------------------------------------------------
      DO j=1,MAX_TYPES
       DO i=1,POINTS1(j)
        x0(i,j) = exp(XSUC(i,j))
        y0(i,j) = exp(XVOLWC(i,j))
       ENDDO
       CALL SMOOTH(j,POINTS1,x0,y0,order1(j),times1(j),semi_log)
      ENDDO
      DO i=1,MAX_TYPES
       CALL WtSplin2(i,POINTS1,x0,y0,z0)
      ENDDO
      IF(Debug_Splines)THEN
c --------------------------------------------------------
c                Graph the curves
c --------------------------------------------------------
       write(*,*)'Enter the Soil Type to graph'
       read(*,*) show
```

```
c     DO i=1,MAX_TYPES
      i=show
      call graph(i,POINTS1,XSUC,XVOLWC,SPLINSL1,y0,z0,
     1   'Suction (kPa)','Volumetric Water Content (dec.)',semi_log)
c     ENDDO
      ENDIF
c     ------------------------------------------------
c                Store Smoothed Curves
c     ------------------------------------------------
      DO j=1,MAX_TYPES
       DO i=1,POINTS1(j)
        XVOLWC(i,j)  = y0(i,j)
        SPLINSL1(i,j) = z0(i,j)
       ENDDO
      ENDDO

C     ==================================================================
C          SPLINING OF THE SUCTION VS K DATA
C     ==================================================================
      DO i=1,MAX_TYPES
       CALL WtSplin2(i,POINTS2,XKSUC,XK,SPLINSL2)
      ENDDO
c     ------------------------------------------------
c                Smooth the curve
c     ------------------------------------------------
      DO j=1,MAX_TYPES
       DO i=1,POINTS2(j)
        x0(i,j) = exp(XKSUC(i,j))
        y0(i,j) = exp(XK(i,j))
       ENDDO
       CALL SMOOTH(j,POINTS2,x0,y0,order2(j),times2(j),logarithmic)
      ENDDO
      DO i=1,MAX_TYPES
       CALL WtSplin2(i,POINTS2,x0,y0,z0)
      ENDDO
      IF(Debug_Splines)THEN
c     ------------------------------------------------
c                Graph the curves
c     ------------------------------------------------
c     DO i=1,MAX_TYPES
      i=show
      call graph(i,POINTS2,XKSUC,XK,SPLINSL2,y0,z0,
     1   'Suction (kPa)','Hydraulic Conductivity (cm/s)',logarithmic)
c     ENDDO
      ENDIF
c     ------------------------------------------------
c                Store Smoothed Curves
c     ------------------------------------------------
      DO j=1,MAX_TYPES
       DO i=1,POINTS2(j)
        XK(i,j)     = y0(i,j)
        SPLINSL2(i,j) = z0(i,j)
       ENDDO
      ENDDO

C     ==================================================================
C        SPLINING OF WC VS THERMAL CONDUCTIVITY DATA
C     ==================================================================
      DO i=1,MAX_TYPES
       CALL WtSplin2(i,POINTS3,XLAMDWC,XLAMD,SPLINSL3)
      ENDDO
c     ------------------------------------------------
c                Smooth the curve
c     ------------------------------------------------
      DO j=1,MAX_TYPES
       DO i=1,POINTS3(j)
        x0(i,j) = exp(XLAMDWC(i,j))
        y0(i,j) = exp(XLAMD(i,j))
       ENDDO
       CALL SMOOTH(j,POINTS3,x0,y0,order3(j),times3(j),linear)
      ENDDO
```

```
      DO i=1,MAX_TYPES
       CALL WtSplin2(i,POINTS3,x0,y0,z0)
      ENDDO
      IF(Debug_Splines)THEN
c     ------------------------------------------
c                 Graph the curves
c     ------------------------------------------
c       DO i=1,MAX_TYPES
        i=show
        call graph(i,POINTS3,XLAMDWC,XLAMD,SPLINSL3,y0,z0,
     1   'Grav. Water Content (dec.)','Thermal Conductivity (W/m^2)',
     1   linear)
c       ENDDO
      ENDIF
c     ------------------------------------------
c                 Store Smoothed Curves
c     ------------------------------------------
      DO j=1,MAX_TYPES
       DO i=1,POINTS3(j)
        XLAMD(i,j)   = y0(i,j)
        SPLINSL3(i,j) = z0(i,j)
       ENDDO
      ENDDO

C     ==============================================================
C      SPLINING OF WC VS VOL SPECIFIC HEAT DATA
C     ==============================================================

      DO i=1,MAX_TYPES
       CALL WtSplin2(i,POINTS4,XSHWC,XSH,SPLINSL4)
      ENDDO
c     ------------------------------------------
c                 Smooth the curve
c     ------------------------------------------
      DO j=1,MAX_TYPES
       DO i=1,POINTS4(j)
        x0(i,j) = exp(XSHWC(i,j))
        y0(i,j) = exp(XSH(i,j))
       ENDDO
       CALL SMOOTH(j,POINTS4,x0,y0,order4(j),times4(j),linear)
      ENDDO
      DO i=1,MAX_TYPES
       CALL WtSplin2(i,POINTS4,x0,y0,z0)
      ENDDO
      IF(Debug_Splines)THEN
c     ------------------------------------------
c                 Graph the curves
c     ------------------------------------------
c       DO i=1,MAX_TYPES
        i=show
        call graph(i,POINTS4,XSHWC,XSH,SPLINSL4,y0,z0,
     1   'Grav. Water Content (dec.)','Specific Heat (J/m^3-C)',
     1   linear)
c       ENDDO
      ENDIF
c     ------------------------------------------
c                 Store Smoothed Curves
c     ------------------------------------------
      DO j=1,MAX_TYPES
       DO i=1,POINTS4(j)
        XSH(i,j)     = y0(i,j)
        SPLINSL4(i,j) = z0(i,j)
       ENDDO
      ENDDO


C     ==============================================================
C      SPLINING OF TEMPERATURE vs UWC
C     ==============================================================
      IF( ICE ) THEN
      DO i=1, MAX_TYPES
```

```
    CALL WtSplin2(i,POINTS7,XVOLUWC,XTEM,SPLINSL7)
   ENDDO
c  ---------------------------------------------------------------
c               Smooth the curve
c  ---------------------------------------------------------------
    DO j=1,MAX_TYPES
     DO i=1,POINTS7(j)
      x0(i,j) = exp(XVOLUWC(i,j))
      y0(i,j) = exp(XTEM(i,j))
     ENDDO
     CALL SMOOTH(j,POINTS7,x0,y0,order7(j),times7(j),logarithmic)
    ENDDO
    DO i=1, MAX_TYPES
      CALL WtSplin2(i,POINTS7,x0,y0,z0)
    ENDDO
    IF(Debug_Splines)THEN
c  ---------------------------------------------------------------
c               Graph the curves
c  ---------------------------------------------------------------
     i=show
c     DO i=1,MAX_TYPES
      call graph(i,POINTS7,XVOLUWC,XTEM,SPLINSL7,y0,z0,
   1    'Unfrozen Water Content (dec.)', 'Negative Temperature (C)',
   1    logarithmic)
c     ENDDO
    ENDIF
c  ---------------------------------------------------------------
c               Store Smoothed Curves
c  ---------------------------------------------------------------
    DO j=1,MAX_TYPES
     DO i=1,POINTS7(j)
      XTEM(i,j)     = y0(i,j)
      SPLINSL7(i,j) = z0(i,j)
     ENDDO
    ENDDO
   ENDIF

C  ==========================================================================
C   SPLINING OF UWC vs TEMPERATURE
C  ==========================================================================
   IF( ICE ) THEN
   DO i=1, MAX_TYPES
    CALL WtSplin2(i,POINTS8,YTEM,YVOLUWC,SPLINSL8)
   ENDDO
c  ---------------------------------------------------------------
c               Smooth the curve
c  ---------------------------------------------------------------
    DO j=1,MAX_TYPES
     DO i=1,POINTS8(j)
      x0(i,j) = exp(ytem(i,j))
      y0(i,j) = exp(yvoluwc(i,j))
     ENDDO
     CALL SMOOTH(j,POINTS8,x0,y0,order8(j),times8(j),semi_log)
    ENDDO
    DO i=1, MAX_TYPES
      CALL WtSplin2(i,POINTS8,x0,y0,z0)
    ENDDO
    IF(Debug_Splines)THEN
c  ---------------------------------------------------------------
c               Graph the curves
c  ---------------------------------------------------------------
c     DO i=1,MAX_TYPES
      i=show
      call graph(i,POINTS8,ytem,yvoluwc,SPLINSL8,y0,z0,
   1    'Negative Temperature (C)','Unfrozen Water Content (dec.)',
   1    semi_log)
c     ENDDO
    ENDIF
c  ---------------------------------------------------------------
c               Store Smoothed Curves
c  ---------------------------------------------------------------
```

```
      DO j=1,MAX_TYPES
       DO i=1,POINTS8(j)
        yvoluwc(i,j)     = y0(i,j)
        SPLINSL8(i,j) = z0(i,j)
       ENDDO
      ENDDO
     ENDIF

C   ==========================================================
C     SPLINING OF dSUC/dTEM vs. unfrozen volumetric water content
C   ==========================================================
     IF( ICE ) THEN
     DO i=1, MAX_TYPES
       CALL WtSplin2(i,POINTS9,GVOLUWC,xgg,SPLINSL9)
     ENDDO
c   ----------------------------------------------------------
c                  Smooth the curve
c   ----------------------------------------------------------
      DO j=1,MAX_TYPES
       DO i=1,POINTS9(j)
        x0(i,j) = exp(gvoluwc(i,j))
        y0(i,j) = exp(xgg(i,j))
       ENDDO
       CALL SMOOTH(j,POINTS9,x0,y0,order9(j),times9(j),logarithmic)
      ENDDO
      DO i=1, MAX_TYPES
        CALL WtSplin2(i,POINTS9,x0,y0,z0)
      ENDDO
      IF(Debug_Splines)THEN
c   ----------------------------------------------------------
c                  Graph the curves
c   ----------------------------------------------------------
c      DO i=1,MAX_TYPES
        i=show
        call graph(i,POINTS9,gvoluwc,xgg,SPLINSL9,y0,z0,
     1    'Unfrozen Water Content (dec.)','dSUC/dTEM',
     1    logarithmic)
c      ENDDO
      ENDIF
c   ----------------------------------------------------------
c                  Store Smoothed Curves
c   ----------------------------------------------------------
      DO j=1,MAX_TYPES
       DO i=1,POINTS9(j)
        xgg(i,j)     = y0(i,j)
        SPLINSL9(i,j) = z0(i,j)
       ENDDO
      ENDDO
     ENDIF

C   ==========================================================
C          SPLINING OF THE GREEN LAI VS DAY DATA
C   ==========================================================
      IF(VEGETATION)THEN
      CALL WtSplin2(1,POINTS5,XLAIDAY,XLAI,SPLINSL5)
c   ----------------------------------------------------
c                  Smooth the curve
c   ----------------------------------------------------
      DO i=1,POINTS5(1)
       x0(i,1) = exp(XLAIDAY(i,1))
       y0(i,1) = exp(XLAI(i,1))
      ENDDO
      CALL SMOOTH(1,POINTS5,x0,y0,order5(1),times5(1),linear)
      CALL WtSplin2(1,POINTS5,x0,y0,z0)
c   ----------------------------------------------------
c                  Graph the curves
c   ----------------------------------------------------
      IF(Debug_Splines)THEN
        call graph(1,POINTS5,XLAIDAY,XLAI,SPLINSL5,y0,z0,
     1    'Day','Green Leaf Area Index',linear)
```

```
      ENDIF
c   -------------------------------------------------
c              Store Smoothed Curves
c   -------------------------------------------------
      DO i=1,POINTS5(1)
       XLAI(i,1)     = y0(i,1)
       SPLINSL5(i,1) = z0(i,1)
      ENDDO

C   =====================================================================
C        SPLINING OF THE MULCH LAI VS DAY DATA
C   =====================================================================
      IF(POINTS6(1).NE.0)THEN
       CALL WtSplin2(1,POINTS6,XMULCHDAY,XMULCH,SPLINSL6)
c   -------------------------------------------------
c              Smooth the curve
c   -------------------------------------------------
       DO i=1,POINTS6(1)
        x0(i,1) = exp(XMULCHDAY(i,1))
        y0(i,1) = exp(XMULCH(i,1))
       ENDDO
       CALL SMOOTH(1,POINTS6,x0,y0,order6(1),times6(1),linear)
       CALL WtSplin2(1,POINTS6,x0,y0,z0)
       IF(Debug_Splines)THEN
c   -------------------------------------------------
c              Graph the curves
c   -------------------------------------------------
        call graph(1,POINTS6,XMULCHDAY,XMULCH,SPLINSL6,y0,z0,
     1   'Day','Dead Mulch Leaf Area Index',linear)
       ENDIF
c   -------------------------------------------------
c              Store Smoothed Curves
c   -------------------------------------------------
       DO i=1,POINTS6(1)
        XMULCH(i,1)   = y0(i,1)
        SPLINSL6(i,1) = z0(i,1)
       ENDDO
      ENDIF
     ENDIF
C   =====================================================================
C        Call Subroutine to Write Splines to a Data File
C   =====================================================================
     CALL WRITESPLINES
C   =====================================================================

     CLOSE(UNIT=14)
C   =====================================================================
     RETURN
999  WRITE(*,*) aline
     STOP 'Error in splines data file'
     END
```

SUBROUTINE WRITE_NOD(Water)

```
C   =====================================================================
C      This subroutine writes out the abreviated version of the
C    daily output to the output file.
C   =====================================================================
     IMPLICIT NONE              ! Ensure that all variables have been correctly defined
     INCLUDE 'FUNCTION.FT'          ! Contains all function declarations
     INCLUDE 'DECLARE.FT'           ! Contains all common block declarations
C   -------------------------------------------------
     REAL    gravwc             ! (dec)   Gravimetric water content at the node points
     INTEGER i                  !         Loop Counter
     INTEGER IOFLUSH                !         Intrinsic Function to flush file buffer
     INTEGER IORESULT                !         Holds return value for ioflush
     INTEGER soil               !         The current soil type
     REAL    satnode            ! (cm/s)  Saturated Hydraulic Conductivity
     REAL    specif             ! (mm/day) the specified rainfall
```

```
      REAL   volWatCon            ! (dec)   Volumetric Water Content
      REAL   press_head           ! (kPa)   Pressure Head
      REAL   total_head           ! (kPa)   Total Head
      REAL   avail_poros          ! (%)     Available Porosity
      REAL   Water                ! (mm/day) The Change in Water in the System
      REAL   k,lamda     ! (cm/s) , (W/mC)
C  ==================================================================
      WRITE(48,7)NDAY
      write(48,*) ttime
7     FORMAT(' Elpsd time = ',I3,' days')
      IF( MAXD_OUT_TODAY.GT.0.0 )THEN
       WRITE(48,*)'WARNING: The system failed to converge within the '
       WRITE(48,*)'         specified maximum number of iterations'
       WRITE(48,*)'         one or more times during the current day!'
       WRITE(48,*)' Total Non-Covergence Time = ',MAXD_OUT_TODAY
      ENDIF

      IF(QW(1,2).EQ.1.00E+20)THEN
       specif = 0.00E0
      ELSEIF(NDAY.EQ.0)THEN
       specif = 0.00E0
      ELSE
       specif = QW(1,2) * 24 * 3600 * 1000
      ENDIF
      write(48,8) PEsum
      write(48,9) AEsum
      write(48,10) PTsum
      write(48,11) ATsum
      write(48,12) ( AEsum + ATsum )
      write(48,13) Water
      write(48,14) specif
      write(48,15) Runoff
      IF(VEGETATION)THEN
       write(48,16) SFLUX(RootDepth(2))
      ELSE
       write(48,17) SFLUX(1)
      ENDIF
      write(48,18) LAI,MULCH
      IF(STEADYSTATE)THEN
       WRITE(*,*)' Converged'
      ELSEIF(.NOT.GRAPHICS)THEN
       WRITE(*,8) PEsum
       WRITE(*,9) AEsum
       WRITE(*,10) PTsum
       WRITE(*,11) ATsum
       WRITE(*,12) ( AEsum + ATsum )
       WRITE(*,13) Water
       WRITE(*,14) specif
       WRITE(*,15) Runoff
       IF(VEGETATION)THEN
        WRITE(*,16) SFLUX(RootDepth(2))
       ELSE
        WRITE(*,17) SFLUX(1)
       ENDIF
       WRITE(*,18) LAI,MULCH
      ENDIF
8     FORMAT(' Pot. Evap.     = ',G9.3,' mm/day  ')
9     FORMAT(' Actual Evap.   = ',G9.3,' mm/day  ')
10    FORMAT(' Pot. Transp.    = ',G9.3,' mm/day  ')
11    FORMAT(' Actual Transp.   = ',G9.3,' mm/day  ')
12    FORMAT(' Actual Evapotrans.   = ',G9.3,' mm/day  ')
13    FORMAT(' Water Balance = ',G9.3,' mm/day  ')
14    FORMAT(' Specified Rainfall = ',G9.3,' mm/day ')
15    FORMAT(' Total Runoff  = ',G9.3,' mm/day  ')
16    FORMAT(' Net Infiltr. = ',G9.3,' mm/day (at root base) ')
17    FORMAT(' Net Infiltr. = ',G9.3,' mm/day (at soil surface) ')
18    FORMAT(' Leaf AI  = ',G9.3,' (Green) ',G9.3,' (Mulch) ')
      IF(NDAY.EQ.0)THEN
       write(48,*) ' Root system extends from n/a to n/a cm depth'
      ELSEIF(VEGETATION)THEN
       WRITE(48,19) (YCORD(1)-YCORD(RootTop(2))),
```

```
1          (YCORD(1)-YCORD(RootDepth(2)))
   IF(.NOT.GRAPHICS)THEN
    write(*,19) (YCORD(1)-YCORD(RootTop(2))),
1           (YCORD(1)-YCORD(RootDepth(2)))
   ENDIF
  ELSE
   WRITE(48,*) ' Root system extends from n/a to n/a cm depth'
  ENDIF
19 FORMAT(' Root system extends from ',F6.2,' to ',F6.2,' cm depth')
C ==========================================================================
  WRITE(48,*)' Y    Water   Temp Grav.Ice  (Ua-Uw)  k'
  WRITE(48,*)' Coord. Content  -   Content    -     '
  WRITE(48,*)' (m)   (%)   (C)   (%)      (kPa)  (m/s)'
C ==========================================================================
  DO i=1,NNODES,(PNODES-1)
   soil     = SOILTYPE(i)
   volWatCon = Calc_VolWc(soil,SUCNOD(i))
   gravwc    = volWatCon/GS(soil)/(1.0E0-PORS(soil))
   satnode   = volWatCon/PORS(soil)
   IF( satnode.GT.1.0E0 )THEN
    satnode = 1.0E0
   ENDIF
   total_head  = -SUCNOD(i)/GRAV + YCORD(i)/100.0E0
   avail_poros = 100.0E0*(1-satnode)
   k = Calc_K(soil,SUCNOD(i),nodvolice(i))
   lamda = Calc_Thermal_Cond(soil,gravwc,tem(i),
1                    nodvolice(i))          ! ( W/mC) Thermal Conductivity

   WRITE(48,20)(YCORD(i)/100.0),(100.0*gravwc),TEM(i),
1          100.0*NODVOLICE(i)*rhoice/gs(soil)/(1.0-pors(soil))
1          ,SUCNOD(i),k,lamda

20    FORMAT(' ',F6.3,F8.3,F6.1,F8.3,G10.3,E11.3,f5.3)
  ENDDO
C ------------------------------------------------
  IORESULT = IOFLUSH(48)
  RETURN
  END
```

SUBROUTINE WRITE_OUT(Water)

```
C ==========================================================================
C   This subroutine writes out the detailed daily output.
C ==========================================================================
  IMPLICIT NONE         ! Ensure that all variables have been correctly defined
  INCLUDE 'FUNCTION.FT'  ! Contains all function declarations
  INCLUDE 'DECLARE.FT'   ! Contains all common block declarations
C ==========================================================================
  REAL      dv        ! Diffusion Coeffecient of Water Through Soil
  REAL      heat      ! Specific Heat at Node
  INTEGER   i         ! Loop Counter
  INTEGER   IOFLUSH   ! Intrinsic Function to flush file buffer
  INTEGER   IORESULT  ! Holds return value for the ioflush function
  REAL      k         ! Hydraulic Conductivity at Node
  REAL      lamda     ! Thermal Conductivity at Node
  REAL      Lapsed    ! Total run time in minutes
  INTEGER   Soil      ! The current soil type
  REAL      satvapour ! Saturated vapour Pressure at Node
  REAL      satCond   ! Hydraulic Conductivity for the Saturated Condition
  REAL      specif    ! (mm/day) the specified rainfall
  CHARACTER*1 t       ! A tab character
  REAL      vapour    ! vapour Pressure at Node
  REAL      volWatCon ! Volumetric Water Content at Gauss Points
  REAL      Water     ! (mm/day) The Change in Water in the System
C ==========================================================================
  t = CHAR(9)
  IF( (QW(1,2).EQ.1.00E+20) .OR. (NDAY.EQ.0) )THEN
   specif = 0.00E0
```

```
      ELSE
       specif = QW(1,2) * 24 * 3600 * 1000 ! convert units from m/s to mm/day
      ENDIF
      WRITE(48,*)'Elapsed',t,'Pot',t,'Act',t,'Pot',t,'Act',t,'Tot',t,
     1        'Water',t,'Spec',t,'Runoff',t,'LAI',t,'LAI',t,'Not'
      WRITE(48,*)'Time',t,'Evap',t,'Evap',t,'Tran',t,'Tran',t,'ET',t,
     1        'Bal',t,'Flux',t,t,'Green',t,'Mulch',t,'Converged'
      WRITE(48,*)'days',t,'(mm)',t,'(mm)',t,'(mm)',t,'(mm)',t,'(mm)',t,
     1        '(mm)',t,'(mm)',t,'(mm)',t,'(mm)',t,t,'(secnds)'
      WRITE(48,1)NDAY,t,PEsum,t,AEsum,t,PTsum,t,ATsum,t,AEsum+ATsum,t,
     1        Water,t,specif,t,Runoff,t,LAI,t,MULCH,t,
     1        MAXD_OUT_TODAY
1     FORMAT(' ',I3,A1,F9.3,A1,F9.3,A1,F9.3,A1,F9.3,A1,F9.3,A1,F9.3,A1,
     1    F9.3,A1,F9.3,A1,G9.3,A1,G9.3,A1,G9.3)
      IF(STEADYSTATE)THEN
        WRITE(*,*)' Converged'
      ELSEIF(.NOT.GRAPHICS)THEN
        IF( ((NDAY/22)*22).EQ.NDAY )THEN   ! Write Headings every 40 days
          WRITE(*,*)'DAY    PE      AE      AT      SF  ',
     1          '  Runoff   WB     Time'
          WRITE(*,*)'      (mm)    (mm)    (mm)    (mm)  ',
     1          '   (mm)    (mm)    (min)'
        ENDIF
        Lapsed = SECNDS(TIME0)/60.0 ! Determining the total elapsed time for run so far.
        WRITE(*,5) NDAY,PEsum,AEsum,ATsum,SFLUX(1),Runoff,Water,Lapsed
5       FORMAT(' ',I3,F8.2,' ',F8.2,' ',F8.2,' ',F8.2,' ',F8.2,
     1          ' ',F8.2,' ',F8.1)
      ENDIF
C ==================================================================
      WRITE(48,*)'Y',t,'GWC',t,'T',t,'Suc',t,'TtlHd',t,'LqF',t,'VpF',t,
     1        'TtlF',t,'PRU',t,'ARU',t,'VWC',t,'Sat',t,'HydCnd',t,
     1        'Dv',t,'VpP'
      WRITE(48,*)'(m)',t,'(%)',t,'(C)',t,'(m)',t,'(kPa)',t,'(mm)',t,
     1        '(mm)',t,'(mm)',t,'(m/m)',t,'(m/m)',t,'(%)',t,'(%)',t,
     1        '(m/s)',t,t,'(kPa)'
C ==================================================================
      DO i=1,NNODES,(PNODES-1)
        soil    = SOILTYPE(i)
        WTWC(i)  = Calc_VolWc(soil,SUCNOD(i))
     1                 /GS(soil)/(1.0E0-PORS(soil))
        volWatCon = WTWC(i)*GS(soil)*(1.0-PORS(soil))
        IF( PORS(soil).LT.volWatCon )THEN
         satCond = 1.0
        ELSE
         satCond = volWatCon/PORS(soil)
        ENDIF
        k = Calc_K(soil,SUCNOD(i))
        dv = Calc_vapour_Diff(TEM(i)+273.0,volWatCon,NODVOLICE(i),PORS(soil))
        satVapour = Calc_SatVp(TEM(i)+273.0)
        IF(SUCNOD(i).lt.0.0E0) THEN
           vapour = satVapour*0.1
        ELSE
           vapour = exp((-2.1674E-03*SUCNOD(i))/(TEM(i)+273.0))
     1          *satVapour*0.1
        ENDIF
        WRITE(48,20)YCORD(i)/100.0,t,100.0*WTWC(i),t,TEM(i),t,
     1     SUCNOD(i),t,YCORD(i)/100.0-SUCNOD(i)/GRAV,t,SFLUXL(i),t,
     1     SFLUXV(i),t,SFLUX(i),t,SFLUXPRU(i)/10.,t,SFLUXARU(i)/10.,
     1     t,100.0*volWatCon,t,100.0*satCond,t,k,t,dv,t,vapour
20      FORMAT(' ',F6.3,A1,F8.3,A1,F6.1,A1,G11.3,A1,G11.3,A1,G10.3,A1,
     1      G10.3,A1,G10.3,A1,G10.3,A1,G10.3,A1,F6.2,A1,F8.2,A1,
     1      G9.2,A1,G11.2,A1,F7.2)
      ENDDO
C ------------------------------------------
      IORESULT = IOFLUSH(48)
C ==================================================================
      RETURN
      END
```

```
C ================================================================
      SUBROUTINE WRITESPLINES
C ================================================================
C   This subroutine writes the splines to a data file as specified
C   in the spline.dat file.
C ================================================================
      IMPLICIT NONE               ! Ensure that all variables have been correctly defined
      INCLUDE 'FUNCTION.FT'       ! Contains all function declarations
      INCLUDE 'DECLARE.FT'        ! Contains all common block declarations
C ================================================================
      CHARACTER*80 aline          ! Used to skip over file comments
      INTEGER datapoints          ! Number of data points to generate
      INTEGER I                   ! Loop Counter
      INTEGER soil                ! Layer to generate splined data points from
      REAL   max_x                ! Maximum X to lookup
      REAL   min_x                ! Minimum X to lookup
      REAL   s                    ! Calculated slope of the curve at X.
      real   spline_min           ! The smallest suction in the spline
      REAL   x                    ! X coordinate of data point
      REAL   y                    ! Y coordinate of data point
C ================================================================

C ------------------------------------------
C   Skip over initial comment lines in data file
C ------------------------------------------
      READ(14,FMT='(A80)',ERR=999)aline  ! "--------------------"
      READ(14,FMT='(A80)',ERR=999)aline  ! "Raw Spline Data Output Sett"
      READ(14,FMT='(A80)',ERR=999)aline  ! "--------------------"
C ================================================================
C          WRITING OF THE SUCTION VS WC DATA
C ================================================================
      READ(14,FMT='(A80)',ERR=999)aline  ! "Suction vs VolWc"
      READ(14,FMT='(A80)',ERR=999)aline  ! "Layer #Points,Min,Max"
      READ(14,*)soil,datapoints,min_x,max_x
      IF(datapoints.GT.1)THEN
      OPEN(UNIT=15,FILE='SUC_WTWC.TXT',STATUS='UNKNOWN')
      spline_min = exp( xsuc(1,soil) )
      DO 20 i = 0,datapoints-1
      x = min_x + ( max_x-min_x )*I/( datapoints-1 )
      IF(X.le.spline_min)THEN
       y = (suct_int(soil)-RM2WA(soil)*X)/GS(soil)/(1.0E0-PORS(soil))
       s = RM2WA(soil)
      ELSE
       y = FN_POINT(soil,POINTS1,XSUC,XVOLWC,SPLINSL1,x)
     1       /GS(soil)/(1.0E0-PORS(soil))
       s = FN_SLOPE(soil,POINTS1,XSUC,XVOLWC,SPLINSL1,x)
      ENDIF
      WRITE(15,*)x,y,s
20    CONTINUE
      CLOSE(UNIT=15)
      ENDIF
C ------------------------------------------


C ================================================================
C          WRITING OF THE SUCTION VS K DATA
C ================================================================
      READ(14,FMT='(A80)',ERR=999)aline  ! "Suction vs Hyd Cond"
      READ(14,FMT='(A80)',ERR=999)aline  ! "Layer #Points,Min,Max"
      READ(14,*)soil,datapoints,min_x,max_x
      IF(datapoints.GT.1)THEN
      OPEN(UNIT=15,FILE='SUC_HYD.TXT',STATUS='UNKNOWN')
      DO 30 I = 0,datapoints-1
       x = min_x + ( max_x-min_x )*I/( datapoints-1 )
       y = FN_POINT(soil,POINTS2,XKSUC,XK,SPLINSL2,
     +          X)/100.0E0
       WRITE(15,*)x,y
30    CONTINUE
      CLOSE(UNIT=15)
      ENDIF
```

```
C  ------------------------------------------
C  ==========================================================
C       WRITING OF WC VS THERMAL CONDUCTIVITY DATA
C  ==========================================================
   READ(14,FMT='(A80)',ERR=999)aline  ! "WtWc vs Therm Cond"
   READ(14,FMT='(A80)',ERR=999)aline  ! "Layer #Points,Min,Max"
   READ(14,*)soil,datapoints,min_x,max_x
   IF(datapoints.GT.1)THEN
   OPEN(UNIT=15,FILE='WTWC_THC.TXT',STATUS='UNKNOWN')
   DO 40 I = 0,datapoints-1
   x = min_x + ( max_x-min_x )*I/( datapoints-1 )
   y = FN_POINT(soil,POINTS3,XLAMDWC,XLAMD,SPLINSL3,x)
    WRITE(15,*)x,y
40  CONTINUE
   CLOSE(UNIT=15)
   ENDIF
C  ------------------------------------------
C  ==========================================================
C       WRITING OF WC VS VOL SPECIFIC HEAT DATA
C  ==========================================================
   READ(14,FMT='(A80)',ERR=999)aline  ! "WtWc vs Spec. Heat"
   READ(14,FMT='(A80)',ERR=999)aline  ! "Layer #Points,Min,Max"
   READ(14,*)soil,datapoints,min_x,max_x
   IF(datapoints.GT.1)THEN
   OPEN(UNIT=15,FILE='WTWC_SPH.TXT',STATUS='UNKNOWN')
   DO 50 I = 0,datapoints-1
    x = min_x + ( max_x-min_x )*I/( datapoints-1 )
    y = FN_POINT(soil,POINTS4,XSHWC,XSH,SPLINSL4,x)
    WRITE(15,*)x,y
50  CONTINUE
   CLOSE(UNIT=15)
   ENDIF
C  ------------------------------------------
C  ==========================================================
C       WRITING OF TEMPERATURE vs UWC  DATA
C  ==========================================================
   READ(14,FMT='(A80)',ERR=999)aline  ! "Temperature vs Unfrozen Vol w/c"
   READ(14,FMT='(A80)',ERR=999)aline  ! "Layer #Points,Min,Max"
   READ(14,*)soil,datapoints,min_x,max_x
   IF(datapoints.GT.1)THEN
   OPEN(UNIT=15,FILE='TEM_UWC.TXT',STATUS='UNKNOWN')
   DO 55 I = 0,datapoints-1
    x = min_x + ( max_x-min_x )*I/( datapoints-1 )
    y = FN_POINT(soil,POINTS7,XVOLUWC,XTEM,SPLINSL7,x)
    WRITE(15,*)x,y
55  CONTINUE
   CLOSE(UNIT=15)
   ENDIF
C  ------------------------------------------
C  ==========================================================
C       WRITING OF UWC VS TEMPERATURE DATA
C  ==========================================================
   READ(14,FMT='(A80)',ERR=999)aline  ! "Unfrozen Vol w/c vs Temperature"
   READ(14,FMT='(A80)',ERR=999)aline  ! "Layer #Points,Min,Max"
   READ(14,*)soil,datapoints,min_x,max_x
   IF(datapoints.GT.1)THEN
   OPEN(UNIT=15,FILE='UWC_TEM.TXT',STATUS='UNKNOWN')
   DO 56 I = 0,datapoints-1
    x = min_x + ( max_x-min_x )*I/( datapoints-1 )
    y = FN_POINT(soil,POINTS8,ytem,yvoluwc,SPLINSL8,x)
    WRITE(15,*)x,y
56  CONTINUE
   CLOSE(UNIT=15)
   ENDIF

C  ==========================================================
C       WRITING OF dSUC/dTEM  VS UWC DATA
C  ==========================================================
   READ(14,FMT='(A80)',ERR=999)aline  ! "dSUC/dTEM vs. UWC"
   READ(14,FMT='(A80)',ERR=999)aline  ! "Layer #Points,Min,Max"
```

```
      READ(14,*)soil,datapoints,min_x,max_x
      IF(datapoints.GT.1)THEN
      OPEN(UNIT=15,FILE='dSdT_UWC.TXT',STATUS='UNKNOWN')
      DO 58 I = 0,datapoints-1
       x = min_x + ( max_x-min_x )*I/( datapoints-1 )
       y = FN_POINT(soil,POINTS9,gvoluwc,xgg,SPLINSL9,x)
       WRITE(15,*)x,y
58     CONTINUE
      CLOSE(UNIT=15)
      ENDIF

C ===================================================================
C         WRITING OF LAI VS DAY DATA
C ===================================================================
      IF(VEGETATION)THEN
       READ(14,FMT='(A80)',ERR=999)aline ! "LAI vs Day"
       READ(14,FMT='(A80)',ERR=999)aline ! "Layer #Points,Min,Max"
       READ(14,*)soil,datapoints,min_x,max_x
       IF(datapoints.GT.1)THEN
        OPEN(UNIT=15,FILE='LAI_DAY.TXT',STATUS='UNKNOWN')
        DO 60 I = 0,datapoints-1
         x = min_x + ( max_x-min_x )*I/( datapoints-1 )
         y = FN_POINT(1,POINTS5,XLAIDAY,XLAI,SPLINSL5,x)
         WRITE(15,*)x,y
60       CONTINUE
        CLOSE(UNIT=15)
       ENDIF
      ENDIF
C ------------------------------------
      RETURN
999   WRITE(*,*) aline
      STOP 'Error in splines data file'
      END
```

SUBROUTINE WtSplin2(soil,N,X,Y,SPLINE)

```
C ===================================================================
C    This subroutine calculates the spline weights given the number
C  of points, and the X and Y coordinates of each point.
C    This subroutine was originally supplied by geoslope and modified
C  slightly to meet our needs.
C ===================================================================
      IMPLICIT NONE                ! Ensure that all variables have been correctly defined
      INCLUDE 'FUNCTION.FT'          ! Contains all function declarations
      INCLUDE 'DECLARE.FT'           ! Contains all common block declarations
C ===================================================================
      REAL  a   (MAX_POINTS)        ! Temporary Matrixes used in Spline Calc's
      REAL  b   (MAX_POINTS)        ! Temporary Matrixes used in Spline Calc's
      REAL  beta              ! Temporary Variable
      REAL  deltaX(MAX_POINTS)        ! Temporary Variable
      REAL  deltaY(MAX_POINTS)        ! Temporary Variable
      REAL  c   (MAX_POINTS-1)       ! Temporary Matrix used in Spline Calc's
      INTEGER i                ! Loop Counter
      REAL  gam  (MAX_POINTS)        ! Temporary Vector for Spline
      INTEGER soil               ! Current Layer
      INTEGER N   (MAX_TYPES)        ! Number of Data Points
      REAL  wi,wi1             ! Temporary Calculation Variables
      REAL  r   (MAX_POINTS)        ! Temporary Matrixes used in Spline Calc's
      REAL  SPLINE(MAX_POINTS,MAX_TYPES) ! Slope Vector for Spline
      REAL  X   (MAX_POINTS,MAX_TYPES) ! X coordinates of data points
      REAL  Y   (MAX_POINTS,MAX_TYPES) ! Y coordinates of data points
C ===================================================================
      X   (1,soil) = LOG(X(1,soil))
      Y   (1,soil) = LOG(Y(1,soil))
      SPLINE(1,soil) = 0.0
      b   (1)    = 2.0
      DO i = 2,N(soil)
       X   (i,soil)  = LOG(X(i,soil))
       Y   (i,soil)  = LOG(Y(i,soil))
```

```
      IF ( X(i,soil).LT.X(i-1,soil) )THEN
       WRITE(*,*) ' X values not in ascending order '
       stop
      ENDIF
      SPLINE(i,soil)  = 0.0
      b   (i)    = 2.0
      deltaX(i-1)   = X(i,soil) - X(i-1,soil)
      deltaY(i-1)   = Y(i,soil) - Y(i-1,soil)
     ENDDO

     c(1)    = 1.0
     a(N(soil)) = 1.0
     r(1)    = 3.0*( deltaY(1)     /deltaX(1)     )
     r(N(soil)) = 3.0*( deltaY(N(soil)-1)/deltaX(N(soil)-1))
     wi1     = EXP(-3.0*LOG( 1.0+(deltaY(1)*deltaY(1))
    1                 /(deltaX(1)*deltaX(1)) ))
     DO i=2,N(soil)-1
      wi  = wi1
      wi1  = EXP(-3.0*LOG( 1.0+(deltaY(i)*deltaY(i))
    1                 /(deltaX(i)*deltaX(i)) ))
      a(i) = wi*deltaX(i)/(wi*deltaX(i)+wi1*deltaX(i-1))
      c(i) = 1.0 - a(i)
      r(i) = 3.0*a(i)*deltaY(i-1)/deltaX(i-1)
    1       +3.0*c(i)*deltaY(i) /deltaX(i)
     ENDDO
C    *****************************************************************
C                    TRIDIAG_SOLVER
C    *****************************************************************
     IF ( ABS(b(1)).LT.1.0E-15) THEN
      WRITE(*,*) ' Zero on diagonal; cannot solve Tridiag equations'
      stop
     END IF
     beta = b(1)
     spline(1,soil) = r(1)/beta
     DO i = 2,N(soil)
      gam(i) = c(i-1)/beta
      beta = b(i) - a(i) * gam(i)
      IF ( ABS(beta).LT.1.0E-8 ) THEN
       WRITE(*,*) ' Divide by zero; cannot solve Tridiag equations'
       stop
      END IF
      spline(i,soil) = ( r(i)-a(i)*spline(i-1,soil))/beta
     ENDDO
     DO i=n(soil)-1,1, -1
      spline(i,soil) = spline(i,soil) - gam(i+1) * spline(i+1,soil)
     ENDDO
C    *****************************************************************
     DO  i=1,N(soil)
      IF( r(i).EQ.0.0 ) SPLINE(i,soil)=0.0E0
     ENDDO
C    ================================================================
     RETURN
     END
```

APPENDIX E
SUB FUNCTION CODE

```
C   This function calculates a sin distribution for relative humidity
C ===========================================================================
      REAL FUNCTION Calc_AirRH(DayLeng,TimeScnds)
C ===========================================================================
      IMPLICIT NONE              ! Ensure that all variables have been correctly defined
      INCLUDE 'DECLARE.FI'
C ===========================================================================
      REAL   DayLeng             ! (HRS) Number of hours in a day
      REAL   sunrise             ! (HRS) Hour of the day in which the sun rises (ie 12:00am = 0)
      REAL   sunset              ! (HRS) Hour of the day in which the sun sets
      REAL   MaxRh               ! (K)  Max air temp
      REAL   MinRh               ! (K)  Min air temp of the current day
      REAL   MaxRh0              ! (K)  max air temp of the previous day
      REAL   MaxRh2              ! (K)  max air temp of the next day
      REAL   TimeScnds           ! (Seconds) current time
      REAL   timeHrs             ! (HRS) current time
C ===========================================================================
      IF(DFLUX)THEN
        Calc_AirRH = RH_Max(2)
      ELSE
       sunrise = 12.0 - (DayLeng/2.0) + Rh_Lag
       sunset  = 12.0 + (DayLeng/2.0) + Rh_Lag
       timeHrs = TimeScnds / 3600.0
       MinRh = RH_Min(2)
       MaxRh = RH_Max(2)
C ------------------------------------------
       maxRh0 = RH_Max(1)
       maxRh2 = RH_Max(3)
C ------------------------------------------
       IF((timeHrs.GT.sunrise).AND.(timeHrs.LT.sunset))THEN
        Calc_AirRH = MaxRh + (MinRh - MaxRh) *
     1   SIN(pi*(timeHrs - sunrise)/DayLeng)
       ELSEIF(timeHrs.LE.sunrise) THEN
        Calc_AirRH = ( MaxRh0 + MaxRh + (MaxRh-MaxRh0)
     1                   *timeHrs/sunrise )/2.0
       ELSE
        Calc_AirRH = ( MaxRh  +  (MaxRh2-MaxRh)/2.0
     1                   *(timeHrs-sunset)/(24.-sunset))
       ENDIF
      ENDIF
C ===========================================================================
      RETURN
      END
```

```
C   This function calculates a sin distribution for temperature
C ===========================================================================
      REAL FUNCTION Calc_AIRTemp(DayLeng,TimeScnds)
C ===========================================================================
      IMPLICIT NONE              ! Ensure that all variables have been correctly defined
      INCLUDE 'DECLARE.FI'
C ===========================================================================
      REAL   DayLeng       ! (HRS)  Number of hours in a day
      REAL   max_1         ! (K)    Max air temp
      REAL   min_1         ! (K)    Min air temp of the current day
      REAL   min_0         ! (K)    Min air temp of the previous day
      REAL   min_2         ! (K)    Min air temp of the next day
      REAL   sunrise       ! (HRS)  Hour of the day in which the sun rises (ie 12:00am = 0)
      REAL   sunset        ! (HRS)  Hour of the day in which the sun sets
      REAL   TimeScnds     ! (scnds) current time
      REAL   timehrs       ! (HRS)  current time
C ===========================================================================
      IF(DFLUX)THEN
        Calc_airtemp = TEMPAMAX(2) + 273.0
      ELSE
```

```
      sunrise = 12.0 - (DayLeng/2.0) + Temperature_Lag
      sunset  = 12.0 + (DayLeng/2.0) + Temperature_Lag
      timehrs = TimeScnds / 3600.0
      max_1   = TEMPAMAX(2)
      min_0   = TEMPAMIN(1)
      min_1   = TEMPAMIN(2)
      min_2   = TEMPAMIN(3)
C     ----------------------------------------
      IF((timehrs.GT.sunrise).AND.(timehrs.LT.sunset))THEN
        Calc_airtemp = min_1 + (max_1 - min_1) *
     1          SIN(pi*(timehrs - sunrise)
     1              /(sunset - sunrise) )      + 273.0   ! (K)
       ELSEIF(timehrs.LE.sunrise) THEN
        Calc_airtemp = ( min_0+min_1 + (min_1-min_0)
     1                  *timehrs/sunrise )/2.0 + 273.0   ! (K)
       ELSE
        Calc_airtemp = ( min_1 +  (min_2-min_1)/2.0
     1           *(timehrs-sunset)/(24.-sunset)) + 273.0   ! (K)
       ENDIF
      ENDIF
C     ==================================================================
      RETURN
      END
```

```
C      This function is used by the subroutine 'ITERATE' to determine
C     if convergence has been achieved.
C     ==================================================================
```

LOGICAL FUNCTION Convergence()

```
C     ==================================================================
      IMPLICIT NONE
      INCLUDE 'DECLARE.FT'             ! Contains all common block declarations
C     ==================================================================
      INTEGER i,soil                   ! Loop Counter
      LOGICAL converged                ! Logical flag to indicate when system has converged
      REAL   diff                  ! Relative change in Suction or Temperature
      REAL   volwc,voluwc
      REAL   fn_point
C     ==================================================================
      converged = TRUE
      i = 0
      DO WHILE( i.LT.NNODES .AND. converged )
        i = i + 1

C       ----------------------------------------
C       Convergence for suction is based upon a relative convergence
C         which is checked at every node.
C       ----------------------------------------
        IF(PRESNOD(i).NE.0.0E0)THEN      ! Prevent division by zero
          diff =ABS( SUCNOD(i)-PRESNOD(i) )/PRESNOD(i)
        ELSEIF(SUCNOD(i).NE.0.0E0)THEN
          diff = 1.0E0
        ENDIF

        IF(diff.GT.PUSNORM)THEN
         converged = FALSE
         IF( STEADYSTATE)THEN
           WRITE(*,1)i,diff*100.0,SUCNOD(i)
 1          FORMAT('  Node',I3,'  Change ',F6.2,
     1          '%   Suction',G17.4)
         ENDIF
        ENDIF

C       ----------------------------------------
C       Convergence for temperature is based upon a relative conv.
C         which is checked at every node.
C       ----------------------------------------
        IF(PRETNOD(i).NE.0.0E0)THEN      ! Prevent division by zero
          diff  = ABS( TEM(i)-PRETNOD(i) )/PRETNOD(i)
```

```
      ELSEIF(TEM(i).NE.0.0E0)THEN
        diff = 1.0E0
      ENDIF

      IF( diff.GT.PUTNORM )THEN
        IF( STEADYSTATE.AND.converged )THEN
          WRITE(*,2)i,diff*100.0,TEM(i)
2          FORMAT('  Node',I3,'  Change ',F6.2,
   1          '%  Temperature',F9.2)
        ENDIF
        converged = FALSE
      ENDIF

C    ----------------------------------------
C    Convergence for ice content compares the unfrozen water content
C    given by the soil freezing curve and soil water curve
C    ----------------------------------------
ENDDO
    Convergence = converged
C   ==========================================================
    RETURN
    END
```

```
C  This function calculates the time (HR) of the DAYLENG
C  ==========================================================
      REAL FUNCTION Calc_DAYLENG(N)      !(Hours)
C  ==========================================================
    INCLUDE 'DECLARE.FT'
C  ==========================================================
    REAL    degrees   ! Degrees
    REAL    ws        ! Daylight angle (degrees)
    INTEGER N         ! No. of days past Jan 1.

C  ==========================================================
    degrees     = 23.45*SIN((360.0*(284.0+N)/365)*PI/180)
    ws          = ACOS(-TAN(LAT*PI/180.0)*TAN(degrees*PI/180.0))
    Calc_DAYLENG = 2*(12.0/PI)*ws
C  ==========================================================
    RETURN
    END
```

```
C     This function does a spline lookup and returns the Y value
C   corresponding to the supplied X value.
C  ==========================================================
      REAL FUNCTION Fn_Point(Type,N,X,Y,Spline,Lookup)
C  ==========================================================
    IMPLICIT NONE                 ! Ensure that all variables have been correctly defined
    INCLUDE 'DECLARE.FT'          ! Contains all common block declarations
C  ==========================================================
    INTEGER Type                  ! Current Layer
    INTEGER N    (MAX_TYPES)      ! Number of Points in Spline
    REAL    X    (MAX_POINTS,MAX_TYPES)  ! X coordinates of data points
    REAL    Lookup                ! X value of point to lookup
    REAL    Y    (MAX_POINTS,MAX_TYPES)  ! Y coordinates of data points
    REAL    Spline(MAX_POINTS,MAX_TYPES) ! Calculated data from Spline
C  ==========================================================
    REAL    a,b,hi,phi,xi         ! Temporary Calculation Vars
    INTEGER k,khi,klo             ! Temporary Calculation Vars
    REAL    newX                  ! Log of X value of point to lookup
    REAL    newY                  ! Y value at newX
C  ----------------------------------------
    newX = LOG(Lookup)
    IF ( newX.LT.X(1,Type) ) THEN
      Fn_Point = EXP(Y(1,Type))
      RETURN
```

```
      END IF
      IF ( newX.GT.X(N(Type),Type) ) THEN
        Fn_Point = EXP(Y(N(Type),Type))
        RETURN
      END IF
C  ------------------------------------------------
C  Find the two points the value lies between ( binary search )
C  ------------------------------------------------
      klo = 1
      khi = N(Type)
      do while( (khi-klo).GT.1 )
        k = (khi+klo)/2
        if( X(k,Type).GT.newX)THEN
          khi = k
        else
          klo = k
        endif
      enddo
C  ------------------------------------------------
C  Calculate the lookup point
C  ------------------------------------------------
      a   = newX      - X(klo,Type)
      b   = newX      - X(khi,Type)
      hi  = X(khi,Type) - X(klo,Type)
      phi = 2.0/(hi**3)*(a+0.5*hi)*(b**2)
      xi  = 1.0/(hi**2)*(a      )*(b**2)

      newY = Y(klo,Type)*phi + Spline(klo,Type)*xi

      phi = -2.0/(hi**3)*(b-0.5*hi)*(a**2)
      xi  =  1.0/(hi**2)*(b      )*(a**2)

      newY = newY + Y(khi,Type)*phi + Spline(khi,Type)*xi
C  ------------------------------------------------
      Fn_Point = EXP(newY)
C  ================================================================
      RETURN
      END
```

```
C     This function does a spline lookup and returns the slope of
C  the function at the supplied X value.
C  ================================================================
        REAL FUNCTION Fn_Slope(Type,N,X,Y,Spline,Lookup)
C  ================================================================
      IMPLICIT NONE                    ! Ensure that all variables have been correctly defined
      INCLUDE 'DECLARE.FT'             ! Contains all common block declarations
C  ================================================================
      INTEGER Type                     ! Current Layer
      INTEGER N    (MAX_TYPES)         ! Number of Points in Spline
      REAL    X    (MAX_POINTS,MAX_TYPES)  ! X coordinates of data points
      REAL    Lookup                   ! X value of point to lookup
      REAL    Y    (MAX_POINTS,MAX_TYPES)  ! Y coordinates of data points
      REAL    Spline(MAX_POINTS,MAX_TYPES) ! Calculated data from Spline
C  ================================================================
      REAL    a,b,hi,phi,xi            ! Temporary Calculation Vars
      INTEGER k,khi,klo                ! Temporary Calculation Vars
      REAL    newX                     ! Log of X value of point to lookup
      REAL    newY                     ! Y value corresponding to the provided X
      REAL    slope                    ! Slope value at newX
C  ================================================================
C           Function Name Declarations
C  ================================================================
      REAL    Fn_Point                 ! Function which calculates the Y value of the Spline at a specified point

C  ================================================================
      newX = LOG(Lookup)
```

```
      IF ( newX.LT.X(1,Type) ) THEN
       newY    = EXP(Y(1,Type))
       Fn_Slope = (newY/Lookup)*Spline(1,Type)
       RETURN
      END IF
      IF ( newX.GT.X(N(Type),Type) ) THEN
       newY    = EXP(Y(N(Type),Type))
       Fn_Slope = (newY/Lookup)*Spline(N(Type),Type)
       RETURN
      END IF
C  -----------------------------------------------
C   Find the two points the value lies between ( binary search )
C  -----------------------------------------------
      klo = 1
      khi = N(Type)
      do while( (khi-klo).GT.1 )
       k = (khi+klo)/2
       if( X(k,Type).GT.newX)THEN
         khi = k
       else
         klo = k
       endif
      enddo
C  -----------------------------------------------
C   Calculate the lookup point
C  -----------------------------------------------
      a   = newX     - X(klo,Type)
      b   = newX     - X(khi,Type)
      hi  = X(khi,Type) - X(klo,Type)
      phi  = 2.0/(hi**3)*( b**2+(a+hi/2.0)*2.0*b )
      xi   = 1.0/(hi**2)*( b**2+ a      *2.0*b )

      slope = Y(klo,Type)*phi + Spline(klo,Type)*xi

      phi  = -2.0/(hi**3)*( a**2+(b-hi/2.0)*2.0*a )
      xi   =  1.0/(hi**2)*( a**2+ b      *2.0*a )

      slope = slope + Y(khi,Type)*phi + Spline(khi,Type)*xi
C  -----------------------------------------------
      newY    = Fn_Point(Type,N,X,Y,Spline,Lookup)
      Fn_Slope = (newY/Lookup)*slope
      RETURN
      END
```

```
C  ==================================================================
       REAL FUNCTION
Calc_gFlux(Type,E,C,Sc_0,Sc_1,Pv_0,Pv_1,Gtemp,Gcord,gvolice)
C  ==================================================================
C    This subroutine computes the liquid and vapour fluxes at the
C   element boundaries.
C  ==================================================================
      IMPLICIT NONE
      INCLUDE 'DECLARE.FT'          ! All include files in here
      INCLUDE 'FUNCTION.FT'         ! All function defined here
C  ==================================================================
      INTEGER C                     !        Current Gauss Point
      REAL    c_head                ! (m)    total head at last gauss point
      INTEGER E                     !        Current Element
      REAL    gsuc                  ! (kPa)  Average suction for two gauss points
      REAL    gtem                  ! (K)    Average gauss point temperature
      REAL    gice
      REAL    gvwc                  ! (dec)  Average volumetric water content
      REAL    gdv                   ! (?)    Average coefficient of vapour diffusion
      REAL    gkk                   ! (m/s)  Average hydraulic conductivity
      REAL    Gcord  (MAX_GAUSS)    ! (m)    Gaussian Coordinates
      REAL    Gtemp  (MAX_GAUSS)    ! (K) Temperature at Gauss Pts
```

```
      REAL   Gvolice (MAX_GAUSS)
      INTEGER l                  !       Last Gauss Point
      REAL   l_gradient          ! (m/m)  Head Gradient
      REAL   l_head              ! (m)    total head at last gauss point
      INTEGER Type               ! Current Soil Type
      REAL   Pv_0                ! (kPa)  Vapour pressure at last gauss point
      REAL   Pv_1                ! (kPa)  Vapour pressure at current gauss point
      REAL   Sc_0                ! (kPa)  Suction at last gauss point at the half time step
      REAL   Sc_1                ! (kPa)  Suction at current gauss point at the half time step
      REAL   v_gradient          ! (kPa/m) Vapour Pressure Gradient
C ==================================================================
      IF( C.EQ.AGAUSS .OR. (E.EQ.1 .AND. C.EQ.2) )THEN
        l = C - 1
        gsuc    = ( Sc_0+Sc_1 )/2.0              ! (m)    Suction
        gtem    = ( gtemp(l)+gtemp(C) )/2.0      ! (K)    Average gaussian temperature
        gice    = (gvolice(l)+gvolice(c))/2.0
        l_head  = gcord(l) - Sc_0/GRAV           ! (m)    total head @ l
        c_head  = gcord(C) - Sc_1/GRAV           ! (m)    total head @ m
        l_gradient = ( l_head-c_head )/( gcord(l)-gcord(C) )! (m/m)  total head gradient
        gkk     = Calc_K(Type,gsuc)              ! (m/s)  Hydraulic Conductivity @ t + dt/2
        v_gradient = (Pv_0-Pv_1)/( gcord(l)-gcord(C) )   ! (kPa/m) vapour pressure gradient
        gvwc    = Calc_VolWc(Type,gsuc)          ! (dec)  Volumetric Water Content
        gdv     = Calc_Vapour_Diff(gtem,gvwc,gice,PORS(Type)) ! (m/s)  Coefficient of Vapour Diffusion
        IF(C.EQ.2 .AND. E.EQ.1 )THEN             ! If first element & 2nd gauss pt.
          FLUX_L(1)  = gkk * l_gradient * 1000.0     ! (mm/s)  liquid flux
          FLUX_V(1)  = gdv * v_gradient * 1000.0     ! (mm/s)  vapour flux
          VFLUX (1)  = FLUX_L(1) + FLUX_V(1)         ! (mm/s)  total flux @ surface
        ENDIF
        IF(C.EQ.AGAUSS)THEN                      ! If last gauss pt. of element
          FLUX_L(E+1)= gkk * l_gradient * 1000.0     ! (mm/s)  liquid flux
          FLUX_V(E+1)= gdv * v_gradient * 1000.0     ! (mm/s)  vapour flux
          Calc_gFlux = ( FLUX_V(E+1)+FLUX_L(E+1) )   ! (mm/s)  total flux @ element boundary
        ELSE
          Calc_gFlux = 0.0
        ENDIF
      ENDIF
C ==================================================================
      RETURN
      END
```

```
C  This function calculates the specific heat
C ==================================================================
```

real FUNCTION Calc_Specific_Heat(Soil,WatCon,icecon)

```
C ==================================================================
      IMPLICIT NONE              ! Ensure that all variables have been correctly defined
      INCLUDE 'DECLARE.FT'
C ==================================================================
      real   Fn_Point            ! Function which calculates the Y value of the Spline at a specified point
      integer Soil               ! The current soil type
      real   WatCon,icecon       ! (dec) The water content
C ==================================================================
      integer i
       real   t1,t2,t3,t4
C ------------------------------------------

      t1 = exp(XSHWC(1,Soil))
      t2 = exp(XSHWC(POINTS4(Soil),Soil))
      if( WatCon.lt.t1 )then
       t2 = exp(XSH (1,Soil))
       t3 = exp(XSH (2,Soil))
       t4 = exp(XSHWC(2,Soil))
       Calc_Specific_Heat = t2-((t3-t2)/(t4-t1))*(t1-WatCon)
      elseif( WatCon.gt.t2 )then
       i = POINTS4(Soil) - 1
       t1 = exp(XSH(POINTS4(Soil),Soil))
       t3 = exp(XSH (i,Soil))
       t4 = exp(XSHWC(i,Soil))
       Calc_Specific_Heat = t1+((t1-t3)/(t2-t4))*(WatCon-t2)
```

```
      else
       Calc_Specific_Heat = Fn_Point(Soil,POINTS4,XSHWC,XSH,SPLINSL4,
     1                         WatCon)
      endif

      if(icecon.gt.0)then
      icecon=rhoice*icecon/(GS(soil)*(1-pors(soil)))

      Calc_specific_heat=Calc_specific_heat+1.85*1000000*2.1*icecon
      endif
C     ==================================================================
      RETURN
      END
```

```
C     This function calculates the hydraulic conductivity
C     ==================================================================
```

real FUNCTION Calc_K(Soil,Suction,nodice)

```
C     ==================================================================
      IMPLICIT NONE                 ! Ensure that all variables have been correctly defined
      INCLUDE 'DECLARE.FT'
C     ==================================================================
      real   Fn_Point, Fn_slope     ! Function which calculates the Y value of the Spline at a specified point
      integer Soil                  ! The current soil type
      real   Suction                ! (kpa) The corresponding matrix Suction
C     ==================================================================
      real   xkk,temp,nodice,volwc,eSat
C     ------------------------------------------
      temp = SATK(Soil)/100.0
      if( Suction.gt.0.0 )then
        xkk = Fn_Point(Soil,POINTS2,XKSUC,XK,SPLINSL2,Suction)*temp

        if( xkk.gt.temp )then
          xkk = temp
        endif
      else
       xkk = temp
      endif

      calc_k = xkk

      if(nodice.gt.0.0) calc_k=xkk*10**(-impfact(soil)*nodice)

C     ==================================================================
      RETURN
      END
```

DOUBLE PRECISION FUNCTION Lump(R,C,m1,m2,lsys)

```
C     ==================================================================
C       This subroutine lumps all the terms in the storage matrix
C     to the diagonal.
C     ==================================================================
      IMPLICIT NONE                 ! Ensure that all variables have been correctly defined
      INCLUDE 'DECLARE.FT'          ! Contains all common block declarations
C     ==================================================================
      INTEGER R             ! current row of storage matrix
      INTEGER C             ! current column of storage matrix
      INTEGER j1,j2,l       ! Loop Counters
      INTEGER m1            ! The width of the band below the diagnol
      INTEGER m2            ! The width of the band above the diagnol
      INTEGER lsys          ! = 2*NNODES
C     ==================================================================
      IF( R.EQ.C )THEN
        j1 = C - m1
        IF( j1.LT.1 )THEN
          j1 = 1
```

```
      ENDIF
      j2 = C + m2
      IF( j2.GT.lsys )THEN
       j2 = lsys
      ENDIF
      Lump = 0.0D0
      DO l = j1,j2
        Lump = Lump + SYSMAS(R,l)
      ENDDO
     ELSE
      Lump = 0.0D0
     ENDIF
C  ================================================================
     RETURN
     END
```

```
C  This function calculates a sin distribution for net radiation
C  ================================================================
```

REAL FUNCTION Calc_NETRAD(Dayleng)!(MM/DAY)

```
C  ================================================================
     IMPLICIT NONE                ! Ensure that all variables have been correctly defined
     INCLUDE 'DECLARE.FT'            ! Contains all common block declarations
C  ================================================================
     REAL   Dayleng   ! ( HRS  ) Number of hours in a day
     REAL   ea        ! (mm/day) fu*ea*(B-A)
     REAL   sunrise   ! ( HRS  ) Hour of the day in which the sun rises (ie 12:00am = 0)
     REAL   sunset    ! ( HRS  ) Hour of the day in which the sun sets
     REAL   qmax      ! ( W/m2 ) Max net radiation
     REAL   timehrs   ! ( HRS  ) current time
     REAL   gamma     ! (kPa/C ) Phsychrometer Constant
C  ================================================================
     IF(DFLUX.OR.STEADYSTATE)THEN   ! Calculate an equivalent NetRad
      Calc_NETRAD = 0.0
     ELSE
      sunrise = 12.0 - (Dayleng/2.0)
      sunset  = 12.0 + (Dayleng/2.0)
      timehrs = ttime / 3600.0
      qmax = PI * SOLAR(2) / (2 * (Dayleng * 3600)) * 1.0E+6
    1        * 0.0353    ! W/m2 --> mm/day
      IF((timehrs.GT.sunrise).AND.(timehrs.LT.sunset))THEN
        Calc_NETRAD = qmax * SIN(PI*(timehrs - sunrise)
    1                   /(sunset - sunrise))
      ELSE
        Calc_NETRAD = 0.0
      ENDIF
     ENDIF
C  ================================================================
     RETURN
     END
```

```
C  This source file provides the functions for the Cover Factors
C  ================================================================
```

REAL FUNCTION Calc_PlantLimitFactor(Suction)

```
C  ================================================================
     IMPLICIT NONE                ! Ensure that all variables have been correctly defined
     INCLUDE 'DECLARE.FT'            ! INCLUDES ALL COMMON BLOCK DECLARATIONS
C  ================================================================
     REAL  Suction               ! Nodal Suction at Root Centroid
C  ----------------------------------------
     IF(Suction.LT.LimitingPt) THEN
      Calc_PlantLimitFactor =  1.0
     ELSE IF(Suction.GT.WiltingPt) THEN
      Calc_PlantLimitFactor =  0.0
     ELSE
C        linear relationship
```

```
      Calc_PlantLimitFactor = 1.0 - (Suction-LimitingPt)
     1                /(WiltingPt-LimitingPt)
C          logarithmic relationship
C      Calc_PlantLimitFactor = 1.0 - (LOG(Suction)-LOG(LimitingPt))
C    1                /(LOG(WiltingPt)-LOG(LimitingPt))
      ENDIF
C     ====================================================================
      RETURN
      END
```

```
C    This function calculates the potential root uptake for each node
C     ====================================================================
      REAL FUNCTION Calc_PRU(node)          !(mm/sec)
C     ====================================================================
      IMPLICIT NONE                  ! Ensure that all variables have been correctly defined
      INCLUDE 'DECLARE.FT'              ! Contains all common block declarations
C     ====================================================================
      INTEGER  node        ! (HRS) Number of hours in a day
      REAL     maxRootFlux  ! maximum specific root flux, occurs at surface
      REAL     rootZone     ! depth of the root zone
      REAL     nodeCentroid ! centroid of the depth of influence of each node
C     ====================================================================
      rootZone = YCORD(RootTop(2)) - YCORD(RootDepth(2))
      maxRootFlux = 2 * VFLUXPT / rootZone
      IF(node.EQ.RootTop(2))THEN                 ! if surface node
       nodeCentroid = (YCORD(node) + YCORD(node+1))/2
       NodeContrib(node) = (YCORD(node) - YCORD(node+1))/2
      ELSEIF(node.EQ.RootDepth(2))THEN        ! if base of root zone node
       nodeCentroid = (YCORD(node-1) + YCORD(node))/2
       NodeContrib(node) = (YCORD(node-1) - YCORD(node))/2
      ELSE                              ! if inbetween top and bot node
       nodeCentroid = (YCORD(node-1) + YCORD(node+1))/2
       NodeContrib(node) = (YCORD(node-1) - YCORD(node+1))/2
      ENDIF
      Calc_PRU = maxRootFlux * (1 - (YCORD(RootTop(2)) - nodeCentroid)
     1        / rootZone) * NodeContrib(node)
C     ====================================================================
      RETURN
      END
```

```
C    This function calculates Rh given suction (kPa) and temperature (K)
C     ====================================================================
      REAL FUNCTION Calc_RH(Suction,Temperature)
C     ====================================================================
      IMPLICIT NONE                  ! Ensure that all variables have been correctly defined
C     ====================================================================
      REAL        Suction    ! (kPa) Suction at Node or Guass Point
      REAL        Temperature !  (K)  Temperature of Node or Guass Point
C     ====================================================================
      IF(Suction.GE.0.0E0) THEN
        Calc_RH = EXP( (-2.1674E-03*Suction)/Temperature )
      ELSE
        Calc_RH = 1.0E0
      ENDIF
C     ====================================================================
      RETURN
      END
```

```
C    This function calculates change in volumetric water content per
C    change in matric suction.
C     ====================================================================
      real FUNCTION Calc_RM2W(Soil,Suc)
```

```
C ====================================================================
   IMPLICIT NONE                    ! Ensure that all variables have been correctly defined
   INCLUDE 'DECLARE.FT'
C ====================================================================
   integer Soil              ! The current soil type
   real  Fn_Slope              ! Function which calculates the slope  of the Spline at a specified point
   real  Suc                ! (kpa) The matrix Suction @ t
C ====================================================================
   if(Suc.GT.XSUC1(Soil))then
     Calc_RM2W = -Fn_Slope(Soil,POINTS1,XSUC,XVOLWC,SPLINSL1,Suc)
   else
     Calc_RM2W = RM2WA(Soil)
   endif
C ====================================================================
   RETURN
   END
```

REAL FUNCTION Calc_SatVp(Temp)

```
C -----------------------------------------------
C     This function calculates the saturated vapour pressure given
C   the temperature in Kelvin.
C -----------------------------------------------
   REAL Temp        ! Temperature in Kelvin
C -----------------------------------------------
   IF(Temp.gt.273.16) then
   Calc_SatVp = ( ( ( ( (6.136820929E-11    !this is in [mBARS]
  1          *Temp-8.023923082E-08)
  1         *Temp+4.393587233E-05)
  1       *Temp-1.288580973E-02)
  1     *Temp+2.133357675E0)
  1   *Temp-188.903931E0)
  1  *Temp+6984.505294E0
   elseif(Temp.eq.273.16) then
    Calc_SatVp=6.108
   else
    Calc_SatVp=6.2617*exp(0.0894*(Temp-273.16))
   endif
C -----------------------------------------------
   RETURN
   END
```

```
C -----------------------------------------------
c Function to determine lapsed time from BASE
c A.E. Krause
c
C -----------------------------------------------
```

REAL FUNCTION SECNDS(BASE)

```
C -----------------------------------------------
    IMPLICIT NONE
    REAL BASE,TIME,LASTTIME
    SAVE LASTTIME               ! Must be saved to compare with next call
    INTEGER IHR,IMIN,ISEC,I100TH
C -----------------------------------------------
    CALL GETTIM(IHR,IMIN,ISEC,I100TH)
    TIME = (100.0*(60.0*(IMIN+60.0*IHR)+ISEC)+I100TH)/100.0
    SECNDS=TIME-BASE
    IF(BASE.EQ.0.0)THEN
      LASTTIME = 0.0            ! Initialize Lasttime
    ELSE
     IF(SECNDS.LT.LASTTIME)THEN    ! If a new day has started
      BASE   = BASE - 86400.0
      SECNDS  = TIME-BASE
     ENDIF
     LASTTIME = SECNDS
    ENDIF
C -----------------------------------------------
    RETURN
```

```
      END

C     This function calculates the soil temperature to be applied to the
C      top node.
C     ================================================================
      real FUNCTION Calc_SoilTemp(TheWind) ! Celcius
C     ================================================================
      IMPLICIT NONE          ! Ensure that all variables have been correctly defined
      INCLUDE 'DECLARE.FT'   ! Contains all common variable declarations
C     ================================================================
C     Calculation of surface temperature (Wilson, 1990)
C     ================================================================
      real  TheWind      ! (km/hr)   specified Average Daily Wind Speed
C     ================================================================
      real  air          ! (C)       Current Air Temperature
      real  fu           ! (mm/day/kPa) function of Wind at 2m
      real  gamma        ! (kPa/C)   Phsychrometer Constant
      real  qstar_mod    ! (dec.)    qstar modifier based upon LAI
C     ================================================================
      air = TEMPAIR - 273.0
C     CALCULATION OF CONSTANTS:
C     Calculation of the Psychrometer Constant
c     Linear Interpolation (Monteith, 1973)
      gamma = 6.41212e-05*air+0.064567273 ! kPa/C

C     Calculate the heat supplied by vapour in mm/day
c     volwc = Calc_VolWc(SOILTYPE(1),(SUCNOD(1)+SUCNOD(2))/2.) ! Volumetric Water Content
c     grvwc = volwc/(GS(SOILTYPE(1))*(1-PORS(SOILTYPE(1))))  ! Gravimetric Water Content
c     lamda = Calc_Thermal_Cond(SOILTYPE(1),grvwc)           ! Thermal Conductivity
c     qG   = lamda*(TEM(1)-TEM(2))/(YCORD(1)-YCORD(2))*0.0353*100.

C     Calculation of fu constant
      fu = 2.6252*(1+0.15*TheWind)          !(mm/day)/kPa

C     Calculation of the LAI - Qstar modifier
      IF(VEGETATION)THEN
       IF(LAI.LT.0.1)THEN
        qstar_mod = 1.0*EXP(-MULCH)
       ELSEIF(LAI.GT.2.7)THEN
        qstar_mod = 0.0
       ELSE
        qstar_mod = EXP(-0.4*LAI-MULCH)
       ENDIF
      ELSE
       qstar_mod = 1.0
      ENDIF

C     Calculation of surface temp
      Calc_soiltemp = (1.0/(gamma*fu))*
     1     (QSTAR*qstar_mod+PENMAN*86400.)+air ! (C) Note plus sign is for evap.
      IF( Calc_soiltemp.LT.TEMPAMIN(2) )THEN
        Calc_soiltemp = TEMPAMIN(2)
      ENDIF
c     if(calc_soiltemp.lt.0.) Calc_soiltemp=air+(0.1*snowdepth(nday+1))


C     ================================================================
      RETURN
      END

C     This function calculates the thermal conductivity
C     ================================================================
      real FUNCTION Calc_Thermal_Cond(Soil,WatCon,temp,nodeice)
C     ================================================================
      IMPLICIT NONE               ! Ensure that all variables have been correctly defined
```

```
      INCLUDE 'DECLARE.FI'
C    ================================================================
      REAL   Fn_Point               ! Function which calculates the Y value of the Spline at a specified point
      integer Soil                  ! The current soil type
      REAL   WatCon                 ! (dec) The water content
C    ================================================================
       integer  i
       real     t1, t2, t3, t4
       real     Ks,Ksat,Ke,Kdry,nodeice
       real     volwc,temp
       open(unit=88,file='lamda.dat',status='new')
C    ----------------------------------------------------------
      if (nodeice.le.0.) then     ! use therm. cond. graph above freezing
      t1 = exp(XLAMDWC(1,Soil))
      t2 = exp(XLAMDWC(POINTS3(Soil),Soil))
      if( WatCon.lt.t1 )then
       t2  = exp(XLAMD(1,Soil))
       t3  = exp(XLAMD(2,Soil))
       t4  = exp(XLAMDWC(2,Soil))
       Calc_Thermal_Cond = t2-((t3-t2)/(t4-t1))*(t1-WatCon)
      else if( WatCon.gt.t2 )then
       i  = POINTS3(Soil) - 1
       t1  = exp(XLAMD(POINTS3(Soil),Soil))
       t3  = exp(XLAMD(i,Soil))
       t4  = exp(XLAMDWC(i,Soil))
       Calc_Thermal_Cond = t1+((t1-t3)/(t2-t4))*(WatCon-t2)
      else
       Calc_Thermal_Cond = Fn_Point(Soil,POINTS3,XLAMDWC,XLAMD,
     1                     SPLINSL3,WatCon)
      endif
C    ================================================================
      else               ! Use Johansen's method below freezing
      if(soil.eq.1.or.soil.eq.2) Ks=5.05        ! for equity mine only.
      if(soil.eq.3) ks=4.05
      Kdry=.4
      volwc=watcon*GS(soil)*(1-PORS(soil))
      Ksat=2.2**PORS(soil)*Ks**(1-PORS(soil))*0.269**volwc
      Ke=(volwc+nodeice)/pors(soil)
      Calc_thermal_cond = (Ksat-Kdry)*Ke + Kdry

      endif


C    ================================================================
      RETURN
      END
```

```
C    This function calculates the diffusion coefficient of water vapour
C     through soil.
C    ================================================================
       real FUNCTION Calc_Vapour_Diff(Gtemp,VolWc,volice,Poros)
C    ================================================================
      IMPLICIT NONE                 ! Ensure that all variables have been correctly defined
      include 'declare.fi'
C    ================================================================
      real  Gtemp                   ! (K)  Temperature
      real  VolWc,volice            ! (dec.) Volumetric Water Content
      real  Poros                   ! (dec.) Porosity
C    ================================================================
      real  beta                    ! X-sectional area of soil available for vapour flow */
      real  dvap                    ! (Mg*m/(kN*s)) Coefficient of Vapour Diffusion */
      real  satn                    ! Saturated Hydraulic Conductivity at Node */
C    ----------------------------------------------------------
      dvap = 0.229E-04*(1.0+Gtemp/273.0)**1.75
      if(dvap.lt.0.229E-04)then
        dvap = 0.229E-04
      endif
      satn = VolWc/Poros
```

```
      if(satn.gt.1.0)then
        satn = 1.0
      endif
      beta = Poros-Volwc-volice
      if(beta.lt.0.) then
      beta=0.0
      endif
      Calc_Vapour_Diff = 2.1674E-03*(beta**1.667)*dvap/Gtemp
C   ==================================================================
      RETURN
      END
```

```
C   This function calculates the vertical flux to be applied to the
C     top node.
C   ==================================================================
      REAL FUNCTION Calc_Vflux(RhSoil)        ! (mm/sec)
C   ==================================================================
      IMPLICIT NONE              ! Ensure that all variables have been correctly defined
      INCLUDE 'DECLARE.FT'
C   ==================================================================
      real   ea        ! (mm/day)   fu*ea*(B-A)
      real   fu        ! (mm/day/kPa) function of wind at 2m
      REAL   gamma     ! (kPa/C)    Phsychrometer Constant
      REAL   qstar_mod ! (dec)      Modifies Rn by LAI function
      REAL   RhSoil    ! (dec)      Relative Humidity at Top Node
C   ==================================================================
      IF(QSTAR.EQ.0.0)THEN
       Calc_Vflux = 0.0
      ELSE
C   ==================================================================
C           Modified Penman Method
C   ==================================================================
C     CALCULATION OF CONSTANTS:
C     Calculation of the Psychrometer Constant
c     Method 1: Linear Interpolation (Monteith, 1973)
       gamma = 6.41212e-05*(TEMPAIR-273)+0.064567273

c     Method 2: Storr and Hartog gamma calculation (1975)
c       gamma = (6.5e-4*(PVAIR1*10)+6.35e-07*(PVAIR1*10)*(TEMPAIR-273)+
c   1    3.537e-07*((TEMPAIR-273)**2.0)*(RHAIR1*100.0))*0.1

C     Calculation of fu constant
C       fu = 2.63*(1.0 + (0.537/3.6)*WIND(2)) ! original penman formulation
       fu = 2.63*(1.0 + (0.864/3.6)*WIND(2)) !Doorenbos and Pruit (1977)

C     Calculation of ea (mm/day)
       if(rhair1.gt.0)then
       ea = PVAIR1*fu*(1/RHAIR1 - 1/RhSoil)
       endif

C     Calculating the LAI modifier
       IF(VEGETATION)THEN
        IF(LAI.LT.0.1)THEN
         qstar_mod=1.0*EXP(-MULCH)
        ELSEIF(LAI.GT.2.7)THEN
         qstar_mod=0.0
        ELSE
         qstar_mod=EXP(-0.4*LAI-MULCH)
        ENDIF
       ELSE
        qstar_mod=1.0
       ENDIF

C     Modified Penman Equation
       Calc_Vflux=-((SLPOT1 * QSTAR * qstar_mod + gamma*ea)/
     1          ((gamma*1/RhSoil) + SLPOT1))/86400.0
      ENDIF
C   ==================================================================
```

```
      RETURN
      END
```

```
C   This function calculates the vertical flux to be applied to the
C    top node using the mass transfer method.
C  ===========================================================
      REAL FUNCTION Calc_Dflux(E)               ! (mm/sec)
C  ===========================================================
      IMPLICIT NONE                 ! Ensure that all variables have been correctly defined
      INCLUDE 'DECLARE.FI'
C  ===========================================================
      REAL   E        ! Vapour pressure of soil at the soil surface
C  ===========================================================
C            Modified Mass Transfer Method
C  ===========================================================
      Calc_Dflux = -BCOEF(2)*(E-PVAIR1)
C  ===========================================================
      RETURN
      END
```

```
C   This function calculates the potential evaporation at the top node.
C  ===========================================================
      REAL FUNCTION Calc_VfluxPE()         ! (mm/sec)
C  ===========================================================
      IMPLICIT NONE                 ! Ensure that all variables have been correctly defined
      INCLUDE 'DECLARE.FI'
C  ===========================================================
c   Penman Method
C  ===========================================================
      REAL   ea        ! (mm/day)   fu*ea*(B-A)
      REAL   fu        ! (mm/day/kPa) function of wind at 2m
      REAL   gamma     ! (kPa/C)    Phsychrometer Constant
      REAL   satVpAir  ! (kPa)      Saturated vapour pressure of the air at Ta
C  ===========================================================
      IF( QSTAR.EQ.0.0)THEN
        Calc_VfluxPE = 0.0
      ELSE

C     CALCULATION OF CONSTANTS:
C     Calculation of the Psychrometer Constant
c     Method 1: Linear Interpolation (Monteith, 1973)
        gamma  = 6.41212e-05*(TEMPAIR-273)+0.064567273

c     Method 2: Storr and Hartog gamma calculation (1975)
c       gamma = (6.5e-4*(PVAIR1*10)+6.35e-07*(PVAIR1*10)*(TEMPAIR-273)+
c     1  3.537e-07*((TEMPAIR-273)**2.0)*(RHAIR1*100.0))*0.1

C     Conversion of Net Solar Radiation from W/m2 - day --> mm/day
C       QSTAR = solar * (1e6/86400) * (0.0353)

C     Calculation of fu constant
C       fu = 2.63*(1.0 + (0.537/3.6)*WIND(2)) ! original penman formulation
        fu = 2.63*(1.0 + (0.864/3.6)*WIND(2)) !Doorenbos and Pruit (1977)

C     Calculation of the saturated VP of the air (kPa)
        if(rhair1.gt.0)then
        satVpAir = PVAIR1/RHAIR1
        endif

C     Calculation of ea
        ea = fu*(satVpAir - PVAIR1)

C     Penman Equation:
        Calc_VfluxPE = -(( (SLPOT1 * QSTAR + gamma*ea)/
     1              (gamma + SLPOT1) )) /86400.0
```

```
      ENDIF
C===========================================================
      RETURN
      END
```

```
C   This function calculates the potential evaporation using the mass
C     transfer method.
C  ===========================================================
      REAL FUNCTION Calc_DfluxPE(E1,E2)     ! (mm/sec)
C  ===========================================================
      IMPLICIT NONE              ! Ensure that all variables have been correctly defined
      INCLUDE 'DECLARE.FT'
C  ===========================================================
      REAL  E1      ! Saturation Vapour pressure at pan's water temperature
      REAL  E2      ! Saturation Vapour pressure at soil surface temperature
C  ===========================================================
C          Mass Transfer Method
C  ===========================================================
C     Dalton's Mass Transfer Equation
      IF(VFLUXPAN(2).LE.0.0)THEN
        BCOEF(2)   = -VFLUXPAN(2)/(E1-PVAIR1)
      ENDIF
      Calc_DfluxPE = -BCOEF(2)*(E2-PVAIR1)
C  ===========================================================
      RETURN
      END
```

```
C   This function calculates the POTENTIAL TRANSPIRATION
C  ===========================================================
      REAL FUNCTION Calc_VfluxPT()      ! (mm/day)
C  ===========================================================
      IMPLICIT NONE              ! Ensure that all variables have been correctly defined
      INCLUDE 'DECLARE.FT'
C  ===========================================================
C          Modify Penman by LAI function
C  ===========================================================
      IF(LAI.GT.2.7)THEN
       Calc_VFLUXPT = VFLUXPE
      ELSE
       Calc_VFLUXPT = VFLUXPE * (-0.21 + 0.7 * SQRT(LAI))
      ENDIF
C  ===========================================================
      RETURN
      END
```

```
C   This function calculates the volumetric water content
C  ===========================================================
      REAL FUNCTION Calc_VolWc(soil,Suction)
C  ===========================================================
      IMPLICIT NONE              ! Ensure that all variables have been correctly defined
      INCLUDE 'DECLARE.FT'
C  ===========================================================
      REAL  Fn_Point              ! Function which calculates the Y value of the Spline at a specified point
      integer soil             ! Temporary x variable
      REAL  t2                 ! Temporary variable

      REAL  Suction              ! (kpa) The corresponding matrix Suction
C  ===========================================================

        if(Suction.lt.XSUC1(soil))then
         Calc_VolWc = SUCT_INT(soil) - RM2WA(soil)*Suction
        else
```

```
      Calc_VolWc = FN_POINT(soil,POINTS1,XSUC,XVOLWC,SPLINSL1,
   1              Suction)
      endif
C ===================================================================
   RETURN
   END
```

```
C  This function calculates the volumetric water content
C ===================================================================
```

REAL FUNCTION WaterBalance()

```
C ===================================================================
   IMPLICIT NONE              ! Ensure that all variables have been correctly defined
   INCLUDE 'FUNCTION.FT'
   INCLUDE 'DECLARE.FT'          ! All include files in this file
C ===================================================================
C  Functions
C ===================================================================
   REAL    bottom             ! The flux at the bottom node
   INTEGER i,j               ! loop counters
   INTEGER type               ! The current soil type
   INTEGER node               ! The current node
   REAL    specified          ! The user specified flux
   REAL    top               ! The flux at the top node
   REAL    v0                ! Initial volume of water
   REAL    v1                ! Final volume of water
   double precision vt,ft
   REAL    volwc_i             ! Volumetric water content at current node
   REAL    volwc_j             ! Volumetric water content at next node
   SAVE    v0,ft              ! Retain the volume of water at the beginning of the time step
C ===================================================================
   v1    = 0.0
   type  = SOILTYPE(1)
   volwc_j = Calc_VolWc(type,SUCNOD(1))
   DO i = 1,(NNODES-1)
     j     = i + 1
     volwc_i = volwc_j
     type  = SOILTYPE(j)
     volwc_j = Calc_VolWc(type,SUCNOD(j))
     v1    = v1 + (YCORD(i)-YCORD(j))*(volwc_i+volwc_j)/.2  !mm
  1          +(ycord(i)-ycord(j))*(nodvolice(i)+nodvolice(j))/.2
  1          *rhoice/rhowat
   ENDDO
C -------------------------------------------
   IF( TTIME.EQ.DELTAT )THEN
     v0 = v1
     ft = 0.0D0
   ENDIF
C -------------------------------------------
   top   = VFLUX(1)
   bottom = VFLUX(NNODES)
   DO i = 1,NNBC
     node    = NODEN(i,2)          ! Node to apply boundary condition to
     specified = QW(i,2)
     IF( node.EQ.NNODES )THEN
       IF( specified.NE.1.0E20 )THEN    ! Flag indicating atmospheric forcing
         bottom = specified*1000.0
       ENDIF
     ENDIF
   ENDDO
C -------------------------------------------
   vt = v1-v0
   ft = ft + (top-bottom+VFLUXAT)*DELTAT
   IF( NDAY.GT.0 )THEN
     WaterBalance = vt - ft
   ELSE
     WaterBalance = 0.0
   ENDIF
C ===================================================================
```

RETURN
END

APPENDIX F
SUPPORT FILES REQUIRED BY PROGRAM

```
c     COMMON.FI
C----------------------------------------------------C
C              Common Block Declarations               C
C----------------------------------------------------C
      common /AREA00/ AGAUSS
      common /AREA06/ AESUM
      common /AREA06/ ARU
      common /AREA06/ ATSUM
      common /AREA00/ AW
      common /AREA00/ AX
      common /AREA00/ BCOEF
      common /AREA07/ BOT_MOIS_BNDRY
      common /AREA01/ BTBH
      common /AREA01/ BTBHW
      common /AREA01/ BTBW
      common /AREA01/ BTBWH
      common /AREA07/ CALCULATE_TEMPS
      common /AREA02/ CHMASS
      common /AREA02/ CHSTIFF
      common /AREA02/ CHWSTIFF
      common /AREA02/ CWK
      common /AREA02/ CWMASS
      common /AREA02/ CWHSTIFF
      common /AREA02/ CWSTIFF
      common /AREA00/ DAYS
      common /AREA06/ DELTAT
      common /AREA00/ DELICE

      common /AREA07/ DETAILED
      common /AREA01/ DETJ
      common /AREA07/ DFLUX
      common /AREA01/ DINVJ
      common /AREA01/ DSTK
      common /AREA05/ DURATION
      common /AREA00/ EBH
      common /AREA00/ EBW
      common /AREA00/ FIRST_DELTAT
      common /AREA00/ FLATENT
      common /AREA00/ FLUX_L
      common /AREA00/ FLUX_V
      common /AREA08/ GaussLcFile
      common /AREA08/ GaussWtFile
      common /AREA07/ GRAPHICS
      common /AREA00/ GRAV
      common /AREA00/ GS
      common /AREA06/ INCRUNOFF
      common /AREA07/ ICE
      common /AREA05/ LAI
      common /AREA00/ LAT
      common /AREA05/ LIMITINGPT
      common /AREA04/ POINTS1
      common /AREA04/ POINTS2
      common /AREA04/ POINTS3
      common /AREA04/ POINTS4
      common /AREA04/ POINTS5
      common /AREA04/ POINTS6
      common /AREA04/ POINTS7
      common /AREA04/ POINTS8
      common /area04/ points9
      common /AREA00/ MAXD_OUT_TODAY
      common /AREA00/ MAX_DELTAT
      common /AREA00/ MIN_DELTAT
      common /AREA00/ MOISCODE
      common /AREA05/ MULCH
      common /AREA00/ MXITER
      common /AREA00/ NDAY
c     common /AREA00/ NEBC
```

```
      common /AREA00/ NELEM
      common /AREA00/ NELCON
c     common /AREA00/ NNBC
      common /AREA00/ NNODES
      common /AREA00/ NODEB
      common /AREA05/ NODECONTRIB
      common /AREA00/ NODEN
      common /AREA00/ NODVOLICE

      common /AREA00/ oldnodvolice
      common /AREA00/ NSTART
      common /AREA08/ OUTPUT
      common /AREA06/ PENMAN
      common /AREA06/ PESUM
      common /AREA06/ PHIA
      common /AREA06/ PLANTSUM
      common /AREA00/ PNODES
      common /AREA00/ PORS
      common /AREA00/ PRESNOD
      common /AREA00/ PRETNOD
      common /AREA05/ PRINTTIME
      common /AREA06/ PRU
      common /AREA05/ PTSUM
      common /AREA00/ PUSNORM
      common /AREA00/ PUTNORM
      common /AREA00/ PVAIR1
      common /AREA00/ QSTAR
      common /AREA00/ QW
      common /AREA05/ RAIN
      common /AREA00/ RH_LAG
      common /AREA00/ RH_MAX
      common /AREA00/ RH_MIN
      common /AREA00/ RHAIR1
      common /AREA00/ RHOWAT
      common /AREA00/ RHOICE
      common /AREA00/ RLATENT
      common /AREA00/ RM2WA
      common /AREA05/ ROOTDEPTH
      common /AREA05/ ROOTTOP
      common /AREA06/ RUNOFF
      common /AREA00/ SATK
      common /AREA00/ IMPFACT
      common /AREA06/ SFLUX
      common /AREA06/ SFLUXARU
      common /AREA06/ SFLUXL
      common /AREA06/ SFLUXPRU
      common /AREA06/ SFLUXV
      common /AREA07/ SHUTDOWN
      common /AREA00/ SLPOT1
      common /AREA00/ nexttoptemp
      common /AREA00/ SOLAR
      common /AREA00/ SOILTYPE
      common /AREA04/ SPLINSL1
      common /AREA04/ SPLINSL2
      common /AREA04/ SPLINSL3
      common /AREA04/ SPLINSL4
      common /AREA05/ SPLINSL5
      common /AREA06/ SPLINSL6
      common /AREA06/ SPLINSL7
      common /AREA06/ SPLINSL8
      common /area06/ splinsl9
      common /AREA07/ STEADYSTATE
      common /AREA00/ STSH
      common /AREA00/ STSW
      common /AREA00/ SUC_DAMP
      common /AREA00/ SUCNOD
      common /AREA00/ SUCT_INT
      common /AREA03/ SYSF
      common /AREA03/ SYSMAS
      common /AREA03/ SYSTIF
      common /AREA00/ TEM_DAMP
```

```
      common /AREA00/ TEM
      common /AREA00/ TEMPAIR
      common /AREA00/ TEMPAMAX
      common /AREA00/ TEMPAMIN
      common /AREA00/ TEMPERATURE_LAG
      common /AREA00/ TIME0
      common /AREA00/ TOLS
      common /AREA00/ TOLT
      common /AREA07/ TOP_MOIS_BNDRY
      common /AREA07/ TRANSIENT
      common /AREA06/ TTIME
      common /AREA07/ VEGETATION
      common /AREA06/ VFLUX
      common /AREA06/ VFLUXAE
      common /AREA06/ VFLUXAT
      common /AREA06/ VFLUXPAN
      common /AREA06/ VFLUXPE
      common /AREA06/ VFLUXPT
      common /AREA05/ WILTINGPT
      common /AREA00/ WIND
      common /AREA00/ WTWC
      common /AREA04/ XK
      common /AREA04/ XKSUC
      common /AREA04/ XLAI
      common /AREA05/ XLAIDAY
      common /AREA04/ XLAMD
      common /AREA04/ XLAMDWC
      common /AREA04/ XMULCH
      common /AREA04/ XMULCHDAY
      common /AREA04/ XSH
      common /AREA04/ XSHWC
      common /AREA04/ XSUC
      common /AREA04/ XSUC1
      common /AREA04/ XVOLWC
      common /AREA00/ YCORD
      common /AREA04/ XVOLUWC
      common /AREA04/ XTEM
      common /AREA04/ YVOLUWC
      common /AREA04/ YTEM
      common /area04/ xgg
      common /area04/ gvoluwc
```

```
C===================================================================C
C                  constant.fi                      C
C          This is an include file for Vap1.for.      C
C   This file provides the constants for the soilcover program      C           C
C===================================================================C

C---------------------------------------------C
C            Parameter Name Declarations        C
C---------------------------------------------C

      INTEGER MAX_DAYS      ! Number of Simulation Days
      INTEGER NEBC          ! In general this is a variable, but for our purposes, it is constant
      INTEGER NNBC          ! In general this is a variable, but for our purposes, it is constant
c  We will always have 2, one at the top node & one at the bottom node
      INTEGER MAX_EBC       ! Number of Essential Boundary Conditions (Head type)
      INTEGER MAX_ELEM      ! Number of Elements
      INTEGER MAX_GAUSS     ! Number of Gauss Points
      INTEGER MAX_NBC       ! Number of Natural Boundary Conditions (Flux type)
      INTEGER MAX_NODES     ! Number of Nodal Points
      INTEGER MAX_NODESx2   ! Twice the Number of Nodal Points
      INTEGER MAX_PNODES    ! Number of Nodes per Elements
      INTEGER MAX_POINTS    ! Number of Specified Points for a Spline
      INTEGER MAX_TYPES     ! Number of Soil Types
      REAL   PI             ! PI
      REAL   TINY           ! A very small number
```

```
C-----------------------------------------------------------C
C            Parameter Declarations (Constants)             C
C-----------------------------------------------------------C

   PARAMETER (MAX_DAYS   = 3   ) ! Only 2 days are required, since 1 day is read in advance
   PARAMETER (NEBC       = 2   ) ! This program is only currently able to use 2 EBC's
   PARAMETER (NNBC       = 2   ) ! This program is only currently able to use 2 NBC's
   PARAMETER (MAX_EBC    = 2   )
   PARAMETER (MAX_ELEM   = 100 ) ! = (MAXNODES - 1)/(PNODES-1)
   PARAMETER (MAX_GAUSS  = 7   ) ! Must be greater than or equal to MAX_PNODES
   PARAMETER (MAX_TYPES  = 10  )
   PARAMETER (MAX_NBC    = 3   )
   PARAMETER (MAX_NODES  = 105 )
   PARAMETER (MAX_NODESx2 = 210 ) ! = MAX_NODES * 2
   PARAMETER (MAX_PNODES = 3   )
   PARAMETER (MAX_POINTS = 40  )
   PARAMETER (PI = 3.14159265359 )
   PARAMETER (TINY = 1.0E-20   )

C-----------------------------------------------------------C
C          Logical (Constants)                              C
C-----------------------------------------------------------C
   LOGICAL  TRUE                  ! Used instead of .TRUE.
   LOGICAL  FALSE                 ! Used instead of .FALSE.
   PARAMETER (TRUE    = .TRUE. )       ! Define TRUE
   PARAMETER (FALSE   = .FALSE.)       ! Define FALSE

C-----------------------------------------------------------C
C        Graph Types (Constants)                            C
C-----------------------------------------------------------C

   integer linear,semi_log,logarithmic
   parameter (linear      = 1 )
   parameter (semi_log    = 2 )
   parameter (logarithmic = 3 )
```

C This is Declare.fi

```
C-----------------------------------------------------------C
C            Parameter Name Declarations                    C
C-----------------------------------------------------------C
        INCLUDE 'CONSTANT.FI'    ! Has all the constant declarations

C-----------------------------------------------------------C
C          Variable Declarations for the Common Blocks      C
C-----------------------------------------------------------C

        INTEGER AGAUSS                  ! Number of Gauss Points
        DOUBLE PRECISION AEsum              ! Counter for the actual evaporation
        DOUBLE PRECISION ARU (MAX_NODES)        ! (mm/sec) ACT ROOT UPTAKE FOR EACH TIME STEP
        DOUBLE PRECISION ATsum              ! (mm/day) ACT TRANSP FOR EACH TIME STEP
        REAL   AW    (MAX_GAUSS)        ! Temporary Storage for Gauss Points Wgts
        REAL   AX    (MAX_GAUSS)        ! Temporary Storage for Gauss Point Locations
        REAL   BCOEF  (MAX_DAYS)          ! B Coefficient for the mass transfer method
        REAL   BTBH   (MAX_PNODES,MAX_PNODES)  ! Element Stiffness Matrix associated with heat flow
        REAL   BTBHW  (MAX_PNODES,MAX_PNODES)  ! Element Stiffness Matrix associated with heat coupled to moisture
flow
        REAL   BTBW   (MAX_PNODES,MAX_PNODES)  ! Element Stiffness Matrix associated with moisture flow
        REAL   BTBWH  (MAX_PNODES,MAX_PNODES)  ! Element Stiffness Matrix associated with moisture coupled to heat
flow
        REAL   CHMASS  (MAX_GAUSS)          ! Element Property Coefficients
        REAL   CHSTIFF (MAX_GAUSS)          ! Element Property Coefficients
        REAL   CHWSTIFF(MAX_GAUSS)           ! Element Property Coefficients
        REAL   CWK    (MAX_GAUSS)         ! Element Property Coefficients @ t + dt
        REAL   CWMASS  (MAX_GAUSS)           ! Element Property Coefficients

        REAL   CWSTIFF (MAX_GAUSS)          ! Element Property Coefficients
```

```
      REAL  CWHSTIFF(MAX_GAUSS)          ! Element Property Coefficients
      INTEGER DAYS                       ! Number of Days to Run Simulation
      DOUBLE PRECISION DELTAT            ! Current time step in seconds
      REAL  EBH   (MAX_EBC,MAX_DAYS)     ! Essential Boundary Condition for temperature
      REAL  EBW   (MAX_EBC,MAX_DAYS)     ! Essential Boundary Condition for suction
      REAL  DELICE (max_nodes)           ! Nodal change in ice content over time step

      REAL  DETJ  (MAX_GAUSS)            ! Determinant of Jacobian
      REAL  DINVJ (MAX_GAUSS)            ! Inverse of Jacobian
      REAL  DSTK  (MAX_PNODES)           ! Element load vector related to gravity
      REAL  DURATION(MAX_DAYS)           ! Flag which implements precip. ramp function
      REAL  FIRST_DELTAT                 ! Initial Daily Time Step in seconds
      DOUBLE PRECISION FLUX_L(MAX_NODES)     ! (mm/s) Liquid flux at the element boundary
      DOUBLE PRECISION FLUX_V(MAX_NODES)     ! (mm/s) Vapour flux at the element boundary
      CHARACTER*10 GaussLcFile           ! File of Gauss Pt Locations
      CHARACTER*10 GaussWtFile           ! File of Gauss Pt Weights
      REAL  GRAV                         ! (m/s^2) Accelaration due to gravity
      REAL  GS    (MAX_TYPES)            ! Specific Gravity of Each Soil Layer
      DOUBLE PRECISION INCRUNOFF         ! Instantaneous Runoff rate (mm/s)
      REAL  LAI                          ! Leaf Area Index for each day
      REAL  LAT                          ! Degrees latitude of the site
      REAL  LimitingPt                   !  Limiting point on Plant Limiting Factor function
      REAL  MAXD_OUT_TODAY               ! Counter for TTIME that system didn't converge
      REAL  MAX_DELTAT                   ! Maximum Time Step Allowed in seconds
      REAL  MIN_DELTAT                   ! Minimum Time Step Allowed in seconds
      INTEGER MOISCODE                   ! Flag, 2 = specified initial nodal watercontents
      REAL  MULCH                        ! Daily mulch LAI value
      INTEGER MXITER                     ! Maximum Number of Iterations
      INTEGER NDAY                       ! Current Simulation Day
c     INTEGER NEBC                       ! Number of essential boundary conditions
      INTEGER NELCON  (MAX_PNODES, MAX_ELEM)  ! Element Connectivity Matrix
      INTEGER NELEM                      ! Number of elements
      INTEGER NNODES                     ! Number of Nodes
c     INTEGER NNBC                       ! Number of Natural Boundary Conditions
      INTEGER NSTART                     ! Number of days from January 1st
      INTEGER NODEB  (MAX_EBC  ,MAX_DAYS)    ! Node to which EBC is to be applied
      REAL  NodeContrib (MAX_NODES)      ! Contributing thickness of each node
      INTEGER NODEN  (MAX_NBC+1 ,MAX_DAYS)   ! Node to which NBC is to be applied
      REAL  NODVOLICE (MAX_NODES)        ! Ice storage at each node

      REAL  OLDNODVOLICE (MAX_nodes)
      CHARACTER*11 OUTPUT                ! Output File Name
      DOUBLE PRECISION  PENMAN           ! (mm/day) Evaporation rate calculated by the modified penman method
      DOUBLE PRECISION  PEsum            ! counter for the PE flux
      REAL  PHIA  (MAX_NODESx2)          ! Suctions and Temps for Solver
      DOUBLE PRECISION  PLANTSUM         ! Counter for PLANT ROOT FLUX
      INTEGER PNODES                     ! Number of Nodes per Element
      INTEGER POINTS1 (MAX_TYPES)        ! Number of Data Pts. in VolWc vs Suction
      INTEGER POINTS2 (MAX_TYPES)        ! Number of Data Pts. in Suction vs Permeability
      INTEGER POINTS3 (MAX_TYPES)        ! Number of Data Pts. in ThermCond vs Wc by Wgt
      INTEGER POINTS4 (MAX_TYPES)        ! Number of Data Pts. in SpecHeat vs Wc by Wgt
      INTEGER POINTS5 (MAX_TYPES)        ! Number of Data Pts. in GREEN LAI vs Day
      INTEGER POINTS6 (MAX_TYPES)        ! Number of Data Pts. in MULCH LAI vs Day
      INTEGER POINTS7 (MAX_TYPES)        ! Number of Data Pts. in grav. water vs temp
      INTEGER POINTS8 (MAX_TYPES)        ! Number of Data Pts. in neg. temp vs grav water content
      integer points9 (max_types)
      REAL  PORS  (MAX_TYPES)            ! Porosity of Each Layer
      REAL  PRESNOD (MAX_NODES )         ! Nodal Suctions from Previous Iteration
      REAL  PRETNOD (MAX_NODES )         ! Nodal Temperatures from Prev Iteration
      INTEGER PrintTime                  ! Flag which specifies noon or midnight output
      DOUBLE PRECISION  PRU (MAX_NODES )     ! (mm/sec) POT ROOT UPTAKE FOR EACH TIME STEP
      DOUBLE PRECISION  PTsum            ! (mm/day) COUNTER FOR THE DAILY PT FLUX
      REAL  PUSNORM                      ! Specified Maximum Allowable Change in Suction Normal
      REAL  PUTNORM                      ! Specified Maximum Allowable Change in Temp Normal
      REAL  PVAIR1                       ! (kPa)  Partial Vapour Pressure
      REAL  QSTAR                        ! (mm/day) Net solar radiation
      REAL  QW    (MAX_NBC,MAX_DAYS)     ! Evap flux  One is set aside for plant root flux
      REAL  RAIN                         ! Rainfall rate minus runoff (used in display sub.)
      REAL  RH_LAG                       ! (HRS)  Difference in time between daily peak rh and daily Qnet peak
      REAL  RH_MAX  (MAX_DAYS )          ! Maximum Daily Relative Humidity of the air
      REAL  RH_MIN  (MAX_DAYS )          ! Minimum Daily Relative Humidity of the air
```

```
REAL   RHAIR1                 ! Current  Relative Humidity of the air
REAL   RHOWAT                 ! Density of Liquid Water        (Mg/m^3)
REAL   RHOICE                 ! Density of frozen water
REAL   RLATENT                ! Latent Heat of Vapourization of Water (J/Mg)
REAL   FLATENT                ! Latent Heat of Fusion of water
REAL   RM2WA   (MAX_TYPES)          ! Value of RM2W at 1st Data Point of a Layer
INTEGER RootDepth (MAX_DAYS)        ! Node at maximum root depth for each day
INTEGER RootTop  (MAX_DAYS)         ! Node at top root depth
DOUBLE PRECISION RUNOFF             ! Counter for Daily Runoff
REAL   SATK    (MAX_TYPES)          ! Saturated Permeability of Each Soil Layer
REAL   IMPFACT (max_types)          ! ice impedence factor
DOUBLE PRECISION  SFLUX   (MAX_NODES)   ! (mm/day) Counter for Flux at each element boundary
DOUBLE PRECISION  SFLUXARU(MAX_NODES)   ! (mm/day/cm) counter for act rt upt for each node
DOUBLE PRECISION  SFLUXL  (MAX_NODES)   ! (mm/day) Counter for Flux at each element boundary
DOUBLE PRECISION  SFLUXPRU(MAX_NODES)   ! (mm/day/cm) counter for pot rt upt for each node
DOUBLE PRECISION  SFLUXV  (MAX_NODES)   ! (mm/day) Counter for Flux at each element boundary
REAL   SLPOT1                 ! (kPa/C)  Slope of Sat VP vs Temp Curve at surface.
REAL   SOLAR  (MAX_DAYS )           ! (MJ/m^2/day) Net Solar Radiation
real   nexttoptemp   ! next days user defined temp at surface
INTEGER SOILTYPE(MAX_NODES )        ! The soil type corresponding to the node
REAL   SPLINSL1(MAX_POINTS,MAX_TYPES)   ! Calculated Spline Wgts.
REAL   SPLINSL2(MAX_POINTS,MAX_TYPES)   ! Calculated Spline Wgts.
REAL   SPLINSL3(MAX_POINTS,MAX_TYPES)   ! Calculated Spline Wgts.
REAL   SPLINSL4(MAX_POINTS,MAX_TYPES)   ! Calculated Spline Wgts.
REAL   SPLINSL5(MAX_POINTS,MAX_TYPES)   ! Calculated Spline Wgts.
REAL   SPLINSL6(MAX_POINTS,MAX_TYPES)   ! Calculated Spline Wgts.
REAL   SPLINSL7(MAX_POINTS,MAX_TYPES)   ! Calculated Spline Wgts.
REAL   SPLINSL8(MAX_POINTS,MAX_TYPES)   ! Calculated Spline Wgts.
real   splinsl9(max_points,max_types)
REAL   STSW   (MAX_PNODES,MAX_PNODES)  ! Element Mass Moisture Storage Matrix
real   stswh  (max_pnodes,max_pnodes)  ! mass storage related to temp. below freezing
REAL   STSH   (MAX_PNODES,MAX_PNODES)  ! Element Mass Heat Storage Matrix
REAL   SUC_DAMP               ! Suction Dampening coefficients
REAL   SUCNOD  (MAX_NODES)          ! (kPa) Nodal Suctions
REAL   SUCT_INT(MAX_TYPES)          ! (kPa) The suction intercept of the Moist. Retent. Curve
REAL   SYSF   (MAX_NODESx2)         ! The global load vector
REAL   SYSMAS  (MAX_NODESx2,MAX_NODESx2) ! The global mass storage matrix
REAL   SYSTIF  (MAX_NODESx2,MAX_NODESx2) ! The global Stiffness matrix
REAL   TEM_DAMP               ! Temperature Dampening coefficient
REAL   TEMPAIR                ! (K) The current air temperature
REAL   TEM    (MAX_NODES)           ! (C) Nodal Temperatures
REAL   TEMPAMAX (MAX_DAYS )         ! (C) Max air temp
REAL   TEMPAMIN (MAX_DAYS )         ! (C) Min air temp
REAL   TEMPERATURE_LAG              ! (HRS)  Difference in time between daily peak temp and daily Qnet peak
REAL   TIME0                  ! (sec) Initial time at start of run.
REAL   TOLS                   ! Time Step Tolerence for Nodal Suctions
REAL   TOLT                   ! Time Step Tolerence for Nodal Temperatures
DOUBLE PRECISION TTIME              ! Total elapsed time in seconds of current day
DOUBLE PRECISION VFLUX  (MAX_NODES)    ! (m/s) Vertical Flux at each node
DOUBLE PRECISION VFLUXAE            ! ACTUAL EVAPORATION FOR EACH TIME STEP
DOUBLE PRECISION VFLUXAT            ! ACTUAL TRANSPIRATION FOR EACH TIME STEP
DOUBLE PRECISION VFLUXPAN(MAX_DAYS)    ! (m/s) pan evaporation
DOUBLE PRECISION VFLUXPE            ! (m/s) potential evaporation
DOUBLE PRECISION VFLUXPT            ! (m/s) potential transpiration
REAL   WiltingPt              ! Wilting Point on Plant Limiting Factor function
REAL   WIND   (MAX_DAYS )           ! (km/hr) Average Daily Wind Speed
REAL   WTWC   (MAX_NODES )          ! Gravimetric Nodal Water Contents
REAL   XK     (MAX_POINTS,MAX_TYPES)   ! Permeability Data Points for Suction vs Permeability
REAL   XKSUC  (MAX_POINTS,MAX_TYPES)   ! Suction Data for Suction vs Permeability
REAL   XLAI (MAX_POINTS,MAX_TYPES)     ! GREEN Leaf Area Index for GREEN LAI vs Day
REAL   XLAIDAY (MAX_POINTS,MAX_TYPES)  ! Day input for GREEN LAI vs Day
REAL   XLAMD  (MAX_POINTS,MAX_TYPES)   ! Thermal Conductivity Data for TC vs WC
REAL   XLAMDWC (MAX_POINTS,MAX_TYPES)  ! WaterContent by Wgt Data for Therm. Cond. vs WC
REAL   XVOLUWC (MAX_POINTS,MAX_TYPES)  ! Unfrozen Vol. Water Content for UWc vs. Temp.
REAL   XTEM (MAX_POINTS,MAX_TYPES)     ! Temp. data for UWc vs. Temp.
REAL   YVOLUWC (MAX_POINTS,MAX_TYPES)  ! Unfrozen vol. water content  for TEMP vs UWC curve
REAL   YTEM (MAX_POINTS,MAX_TYPES)     ! Temp. data for TEMP vs UWc curve
REAL   XMULCH  (MAX_POINTS,MAX_TYPES)  ! MULCH LAI FOR MULCH LAI vs DAY CURVE
REAL   XMULCHDAY (MAX_POINTS,MAX_TYPES) ! DAY INPUT FOR MULCH LAI vs DAY CURVE
REAL   XSH    (MAX_POINTS,MAX_TYPES)   ! Spec. Heat Data for Sp.Heat vs WC
REAL   XSHWC  (MAX_POINTS,MAX_TYPES)   ! WC by Wgt. Data for Sp.Heat vs WC
```

```
      REAL  XSUC  (MAX_POINTS,MAX_TYPES)  ! Suction Data for Suction vs. Perm.
      REAL  XSUC1  (MAX_TYPES)            ! Initial Suction Point in Suction vs. Perm.
      REAL  XVOLWC (MAX_POINTS,MAX_TYPES) ! Volumetric Water Content for Suct. vs WC.
      REAL  YCORD  (MAX_NODES)            ! Nodal Coordinates
      real  xgg (max_points,max_types)
      real  gvoluwc (max_points,max_types)
******************************************************************************
* Logical variables and definitions
*
******************************************************************************
      LOGICAL  BOT_MOIS_BNDRY(MAX_DAYS)  ! Bottom Boundry is specified as Volumetric Water Content
      LOGICAL  DETAILED                  ! Logical switch set when detailed output is required.
      LOGICAL  DFLUX                     ! A flag indicating the Darcy Flux analysis is required.
      LOGICAL  GRAPHICS                  ! Logical switch to allow GRAPHICS information to be output.
      LOGICAL  ICE                       ! Flag indicating freeze/thaw is to be modelled
      LOGICAL  SHUTDOWN                  ! Flag indicating runoff
      LOGICAL  STEADYSTATE               ! Flag indicating a steady state analysis is required
      LOGICAL  CALCULATE_TEMPS           ! Flag indicating to intrisically calc surface temperatures
      LOGICAL  TOP_MOIS_BNDRY(MAX_DAYS)  ! Top Boundry is specified as Volumetric Water Content
      LOGICAL  TRANSIENT                 ! Flag indicating a transient solution is required
      LOGICAL  VEGETATION                ! Flag indicating vegetation is to be modelled
C     ==========================================================================

      INCLUDE 'CBLOCKS.FI'   ! Contains all the common block declarations
```

C FUNCTION.FI

```
C     ==========================================================================
C           Function Name Declarations
C     ==========================================================================
   real  Calc_Airtemp           ! Sin distribution for air temp
   real  Calc_AirRH             ! Sin distribution for relative humididty
   real  Calc_DayLeng           ! Calculates the length of the day
   real  Calc_Dflux             ! Calculates AE Using Mass Transfer Method
   real  Calc_DfluxPE           ! Calculates PE
   real  Calc_dicedTEM          ! Calculates the change in vol.ice per dTEM
   real  Calc_dicedsuc          ! Calculates the change in ice per change in suction for CWMASS
   real  Calc_dicedt            ! Calculates the change in vol. per dt
   real  Calc_iceflux           ! Calc ice flux in m/s
   real  Calc_guess_newtem      ! Calculates a guessed new node temp to use to get ice content
   real  Calc_K                 ! Calculates the hydraulic conductivity
   real  Calc_Netrad            ! Sin distribution for net radiation
   real  Calc_RH                ! Calculates the relative humidity
   real  Calc_PlantLimitFactor  ! Calculates the plant limiting factor
   real  Calc_PRU               ! Calculates the Pot Root Uptake for each node
   real  Calc_RM2W              ! Calculates RM2W
   real  Calc_SatVp             ! Calculates the saturated vapour pressure (mbar)
   real  Calc_Soiltemp          ! Calculates the surface temperature of the soil
   real  Calc_Specific_Heat     ! Calculates the specific heat
   real  Calc_Thermal_Cond      ! Calculates the thermal conductivity
   real  Calc_Vapour_Diff       ! Calculates the vapour diffusion
   real  Calc_Vflux             ! Calculates AE Using The Modified Penman Method
   real  Calc_VfluxPE           ! Calculates PE
   real  Calc_VFLUXPT           ! CALCULATES PT
   real  Calc_VolWc             ! Calculates the volumetric water content
   logical Convergence          ! Determines if system has converged
   real  Fn_Point               ! Calculates the Y value of the Spline at a specified point
   real  Fn_Slope               ! Calculates the Slope of the Spline at a specified point
   character inkey
   real  Secnds                 ! Calculates time in seconds for total run time.
   double precision Lump        ! Returns the lumped storage terms on the current row.
```

```
c===========================================================================
c     C O M M O N L Y   U S E D   S P P   P A R A M E T E R S
c===========================================================================
c     Define printer buffer size
```

```
c-----------------------------------
      integer maxbuf
      parameter (maxbuf=100000)
c-----------------------------------
c Define logical unit numbers:
c     iunitp...Move-draw file/printer
c     iunitv...Move-draw file/video
c     iunitz...equip.dat/rough.dat
c     iunitf...Font definition file
c     iunitm...Move-draw spill file
c     iunith...'Hardcopy' disk file
c-----------------------------------
      integer iunitp,iunitv,iunitz,iunitf,iunitm,iunith
      parameter (iunitp=10, iunitv=20, iunitz=30)
      parameter (iunitf=40, iunitm=50, iunith=60)
c-----------------------------------
c Declare character strings:
c     bitmap...Printer bitmap buffer
c     ans......Response from GETC
c     getc.....Function GETC
c     devid....Device ID for DEVICE
c     parity...Parity for DEVICE
c     path.....Path for font files
c-----------------------------------
      character bitmap*1,ans*2,getc*2,devid*4,parity*4,path*40
c-----------------------------------
c Declare bitmap array...Using blank
c common  reduces executable program
c  size for some Fortran compilers
c-----------------------------------
      common bitmap(maxbuf)
c==========================================================================
```

File "Boards..dat"

```
' 4-CGA Color.....320x200 '
' 5-CGA B&W.......320x200 '
' 6-CGA B&W.......640x200 '
' 13-EGA Color.....320x200 '
' 14-EGA Color.....640x200 '
' 15-EGA Mono......640x350 '
' 16-EGA Color.....640x350 '
' 17-MCGA & VGA....640x480 '
' 18-VGA Color.....640x480 '
' 19-MCGA & VGA....320x200 '
' 37-Genoa VGA.....640x480 '
' 39-Genoa VGA.....720x512 '
' 40-Hercules......720x348 '
' 41-Genoa/Orchid..800x600 '
' 45-Genoa EGA.....640x350 '
' 46-Genoa/Orchid..640x480 '
' 47-Genoa VGA.....720x512 '
' 55-Genoa/Orchid.1024x768 '
' 72-AT&T 6300.....640x400 '
' 88-Paradise VGA..800x600 '
' 89-Paradise VGA..800x600 '
' 91-Genoa EGA.....640x350 '
' 92-Genoa VGA.....640x480 '
' 93-Genoa VGA.....720x512 '
' 94-Paradise VGA..640x400 '
' 95-Paradise VGA..640x480 '
' 99-Tatung VGA....720x540 '
'100-Tatung VGA....800x600 '
'115-Genoa VGA.....640x480 '
'121-Genoa VGA.....800x600 '
'124-Genoa VGA.....512x512 '
'125-Genoa VGA.....512x512 '
'200-Everex VGA....640x480 '
```

'201-Everex VGA....752x410 '
'202-Everex VGA....800x600 '
'217-Everex EGA...1280x350 '
'219-Everex EGA....640x350 '
'220-Everex VGA....640x400 '
'221-Everex VGA....512x480 '
'255-VESA SVGA**...800x600 '
'256-VESA SVGA.....640x400 '
'257-VESA SVGA.....640x400 '
'258-VESA SVGA.....800x600 '
'259-VESA SVGA.....800x600 '
'260-VESA SVGA....1024x768 '
'261-VESA SGVA....1024x768 '
'262-VESA SVGA...1280x1024 '
'263-VESA SVGA...1280x1024 '
'296-Video-7 EVGA..752X410 '
'297-Video-7 EVGA..720x540 '
'298-Video-7 EVGA..800x600 '
'299-Video-7 EVGA.1024x768 '
'300-Video-7 EVGA.1024x768 '
'301-Video-7 EVGA.1024x768 '
'302-Video-7 EVGA..640x400 '
'303-Video-7 EVGA..640x480 '
'304-Video-7 EVGA..720x540 '
'305-Video-7 EVGA..800x600 '
'391-Trident EVGA..800x600 '
'392-Trident EVGA..640x400 '
'393-Trident EVGA..640x480 '
'394-Trident EVGA..800x600 '
'395-Trident EVGA.1024x768 '
'396-Trident EVGA.1024x768 '
'397-Trident EVGA.768x1024 '
'398-Trident EVGA.1024x768 '
'700-Wyse 700.....1280x800 '

File “equip.dat”

mon = 16
ifore = 15
iback = 1
nprin = 8
mode = 5
isave = -1
'lpt1', 9600, 'none', 1, 8, "

File “gauslc.dat”

0
-0.57735
0.57735
-0.77459
0
0.77459
-0.86113
-0.33998
0.33998
0.86113
-0.90617
-0.53846
0
0.53846
0.90617
-0.93246
-0.6612

-0.23861
0.23861
0.6612
0.93246

File "gauswt.dat"

2.0
1.00
1.00
0.55555
0.88888
0.55555
0.34785
0.65214
0.65214
0.34785
0.23692
0.47862
0.56888
0.47862
0.23692
0.17132
0.36076
0.46791
0.46791
0.36076
0.17132

Spline Smoothing Settings for soil type 1

Suction vs Volumetric Water Content
Order #Times
1 2
Suction vs Hydraulic Conductivity
Order #Times
1 2
Gravimetric Water Content vs Thermal Conductivity
Order #Times
1 2
Grav. W. C. vs Specific Heat
Order #Times
1 2
Unfrozen w/c vs. Temperature
Order #Times
1 2

Spline Smoothing Settings for soil type 2

Suction vs Volumetric Water Content
Order #Times
1 2
Suction vs Hydraulic Conductivity
Order #Times
1 2
Gravimetric Water Content vs Thermal Conductivity
Order #Times
1 2
Grav. W. C. vs Specific Heat
Order #Times
1 2
Unfrozen w/c vs. Temperature
Order #Times
1 2

Spline Smoothing Settings for soil type 3

Suction vs Volumetric Water Content
Order #Times
1 2
Suction vs Hydraulic Conductivity
Order #Times
1 2
Gravimetric Water Content vs Thermal Conductivity
Order #Times
1 2
Grav. W. C. vs Specific Heat
Order #Times
1 2
Unfrozen w/c vs. Temperature
Order #Times
1 2

Spline Smoothing Settings for soil type 4

Suction vs Volumetric Water Content
Order #Times
1 2
Suction vs Hydraulic Conductivity
Order #Times
1 2
Gravimetric Water Content vs Thermal Conductivity
Order #Times
1 2
Grav. W. C. vs Specific Heat
Order #Times
1 2
Unfrozen w/c vs. Temperature
Order #Times
1 2

Spline Smoothing Settings for soil type 5

Suction vs Volumetric Water Content
Order #Times
1 2
Suction vs Hydraulic Conductivity
Order #Times
1 2
Gravimetric Water Content vs Thermal Conductivity
Order #Times
1 2
Grav. W. C. vs Specific Heat
Order #Times
1 2
Unfrozen w/c vs. Temperature
Order #Times
1 2

Spline Smoothing Settings for soil type 6

Suction vs Volumetric Water Content
Order #Times
1 2
Suction vs Hydraulic Conductivity
Order #Times
1 2
Gravimetric Water Content vs Thermal Conductivity
Order #Times
1 2
Grav. W. C. vs Specific Heat
Order #Times
1 2
Unfrozen w/c vs. Temperature
Order #Times
1 2

Spline Smoothing Settings for soil type 7

Suction vs Volumetric Water Content
Order #Times
1 2
Suction vs Hydraulic Conductivity
Order #Times
1 2
Gravimetric Water Content vs Thermal Conductivity
Order #Times
1 2
Grav. W. C. vs Specific Heat
Order #Times
1 2
Unfrozen w/c vs. Temperature
Order #Times
1 2

Spline Smoothing Settings for soil type 8

Suction vs Volumetric Water Content
Order #Times
1 2
Suction vs Hydraulic Conductivity
Order #Times
1 2
Gravimetric Water Content vs Thermal Conductivity
Order #Times
1 2
Grav. W. C. vs Specific Heat
Order #Times
1 2
Unfrozen w/c vs. Temperature
Order #Times
1 2

Spline Smoothing Settings for soil type 9

Suction vs Volumetric Water Content
Order #Times
1 2
Suction vs Hydraulic Conductivity
Order #Times
1 2
Gravimetric Water Content vs Thermal Conductivity
Order #Times
1 2
Grav. W. C. vs Specific Heat
Order #Times
1 2
Unfrozen w/c vs. Temperature
Order #Times
1 2

Spline Smoothing Settings for soil type 10

Suction vs Volumetric Water Content
Order #Times
1 2
Suction vs Hydraulic Conductivity
Order #Times
1 2
Gravimetric Water Content vs Thermal Conductivity
Order #Times
1 2
Grav. W. C. vs Specific Heat
Order #Times
1 2
Unfrozen w/c vs. Temperature
Order #Times
1 2

Spline Smoothing Settings for vegetation (GREEN)

GREEN LAI vs Day
Order #Times
1 0

Spline Smoothing Settings for vegetation (MULCH)

MULCH LAI vs Day
Order #Times
1 0

Raw Spline Data Output File

Suction vs Volumetric Water Content
Soil #Points First Point Last Point
1,0,5,500
Suction vs Hydraulic Conductivity
Soil #Points First Point Last Point
1,0,1,400
Gravimetric Water Content vs Thermal Conductivity
Soil #Points First Point Last Point
1,0,0.02,0.25
Grav. W. C. vs Specific Heat
Soil #Points First Point Last Point
1,0,0.02,0.25
Temperature vs unfrozen water content
Layer #Points First Point Last Point
1,0,0.02,0.25
Unfrozen water content vs temperature
Layer #Points First Point Last Point
1,0,0.01,10
dSUC/dTEM vs unfrozen water content
Layer #Points First Point Last Point
1,0,0.01,.25
GREEN LAI vs Day
Soil #Points First Point Last Point
1,0,0.01,5
MULCH LAI vs Day
Soil #Points First Point Last Point
1,0,0.01,5

APPENDIX G
SAMPLE INPUT FILES

Main Input File for SoilCover

****

"Analysis Type (0=SteadyState,1=DarcyFlux,2=SoilCover)"
2

Output File Name?
equity.out

"Output Data Corresponding to Conditions at? (1-Noon,2-Midnight)"
2

Constants DataFile Name?
equity.cst

Soil Property DataFile Name?
equity.prp

Mesh DataFile Name?
equity.msh

Daily Input DataFile Name?
equity.day

"Will Vegetation be Modelled? (1=Yes,0=No)"
0

Vegetation Input DataFile Name?
warda.vgt

"Will Freeze/thaw be Modelled? (1=yes, 0=N0)"
1

Freeze/Thaw DataFile Name?
equity.ice

Soil_Mesh_Data_File_For_SoilCover

Convergence_Criteria

Max. Max.Change Max.Change Suction Temperature
Iterations Suction Temperature Dampening Dampening
(%) (%) (%) (%)
50 1 1 20 20

Time_Step_Control

Max.Change Max.Change Minimum First Maximum
Suction Temperature TimeStep TimeStep TimeStep
(%) (%) (secnds) (secnds) (secnds)
5 5 120 2 21600

Soil_Profile_Data

Number Of Element NumberOf
"Nodes ""Type_->(1=Linear,2=Quadratic)"" GaussPts "
33 1 2

Initial_Moisture_Conditions

"Specified_by_->_1=Grav.W/C_,2=Suction,_3=Vol.W/C "
1

Mesh_Data

Node	Soil Type	Elevation (cm)	MoistureCondition (dec._or_kPa)	Temperature (C)
1	1	180	0.17	0
2	1	179	0.174	0
3	1	177	0.1756	0
4	1	175	0.1772	0
5	1	173	0.1788	0
6	1	171	0.1804	0
7	1	168	0.19	0
8	1	164	0.19	0
9	1	160	0.19	0.1
10	1	156	0.19	0.2
11	1	154	0.19	1.2
12	1	150.8	0.19	1.72
13	2	147.6	0.19	2.53
14	2	145.2	0.19	2.74
15	2	145	0.19	2.74
16	2	142	0.19	3.2
17	2	138	0.19	3.5
18	2	135	0.19	3.6
19	2	130	0.19	4.3
20	2	125	0.19	5.2
21	2	116	0.18	5.81
22	2	110	0.18	6.7
23	2	100	0.18	7.7
24	3	90	0.06	8.7
25	3	80	0.06	9.7
26	3	70	0.06	10.7
27	3	60	0.06	11.7
28	3	50	0.06	12.7
29	3	40	0.06	13.7
30	3	30	0.06	14.7
31	3	20	0.06	15.2
32	3	10	0.06	15.2
33	3	0	0.06	15.2

Constants Input File For SoilCover
****

Acceleration Due Gravity (m/s^2)	Density To Water (g/cm^3)	Latent of Heat Of Vaporization (J/Mg)
9.807	1	2.46E+09

Gauss Point Location and Weights Data Files
Gauslc.dat
Gauswt.dat

Freeze/Thaw In put File
*********** *** *** ***** ****

DENSITY OF ICE (g/cm^3)=
0.9
LATENT HEAT OF FU SION OF WATER(J/Mg)=
3.34E+08
ENTER THE I NIT IAL GRAV ICE CONTENT (IN TERMS OF WATER)

1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0

10 0
11 0
12 0
13 0
14 0
15 0
16 0
17 0
18 0
19 0
20 0
21 0
22 0
23 0
24 0
25 0
26 0
27 0
28 0
29 0
30 0
31 0
32 0
33 0

Soil Type # 1
NUMBER OF D ATA PO INTS IN THE UNFROZEN W/C VS. TEMPERATURE CURVE FOR LAYER #1
15
GRAV WAT Co nte nt NE G. TEMP
0.001 840.7696
0.0141 211.1918
0.027663 74.42342
0.0546 17.97534
0.0811 3.711712
0.1 1.363568
0.1192 0.4728
0.14 0.18018
0.1474 0.118762
0.159 0.067568
0.1697 0.045046
0.1888 0.01926
0.207 0.009654
0.2145 0.004504
0.22 0.0009

number of data points in dsuc/dtem vs. volumetric water content function
18
Vol Wat dsuc/dtem
0.02 1110
0.021 1110
0.023 1110
0.024 1110
0.028 1110
0.034 1110
0.041 1110
0.057 1110
0.081 1110
0.114 1110
0.159 1110
0.216 1110
0.268 1110
0.327 1110
0.379 1110
0.434 1110
0.464 1110
0.492 1110

Soil Type # 2
NUMBER OF D ATA PO INTS IN THE UNFROZEN W/C VS. TEMPERATURE CURVE FOR LAYER #2

15
GRAV WAT Co nte nt NE G. TEMP
0.001 840.7696
0.0141 211.1918
0.027663 74.42342
0.0546 17.97534
0.0987 4.311982
0.1293 1.363568
0.158 0.4728
0.1669 0.265874
0.1771 0.13022
0.1854 0.067568
0.1911 0.034252
0.1943 0.01926
0.196 0.009654
0.1975 0.004504
0.1982 0.0009

number of data points in dsuc/dtem vs. volumetric water content function
18
Vol Wat dsuc/dtem
0.02 1110
0.021 1110
0.023 1110
0.024 1110
0.028 1110
0.034 1110
0.041 1110
0.057 1110
0.081 1110
0.114 1110
0.159 1110
0.216 1110
0.268 1110
0.327 1110
0.379 1110
0.434 1110
0.464 1110
0.492 1110

Soil Type # 3
NUMBER OF D ATA PO INTS IN THE UNFROZEN W/C VS. TEMPERATURE CURVE FOR LAYER #2
15
GRAV WAT Co nte nt NE G. TEMP
0.001 420.3848
0.008 25.92072
0.0182 1.635487
0.0305 0.060764
0.04 0.015619
0.06 0.005671
0.08 0.003747
0.104 0.003189
0.1474 0.002714
0.159 0.002653
0.1697 0.002533
0.1888 0.002206
0.2229 0.001878
0.2344 0.000986
0.2394 0.000451

number of data points in dsuc/dtem vs. volumetric water content function
18
Vol Wat dsuc/dtem
0.02 2442
0.021 2442
0.023 2442
0.024 2442
0.028 2442
0.034 2442
0.041 2442

0.057 2442
0.081 2442
0.114 2442
0.159 2442
0.216 2442
0.268 2442
0.327 2442
0.379 2442
0.434 2442
0.464 2442
0.492 2442

Soil_Property_InputFile_for_SoilCover
****

Soil_Type_#1

Porosity Specfic
Gravity
0.36 2.77

Moisture_Characteristic_Curve_For_Soil_Type_#1

NumberOf Mv WaterContent
"DataPoints (1/kPa) Type(1=Gravimetric,2=Volumetric) "
15 0.009061 1
Suction WaterContent
(kPa) (dec)
1 0.22
5 0.2145
10.71519 0.207
21.37962 0.1888
50 0.1697
75 0.159
131.8257 0.1474
200 0.14
524.8075 0.1192
1513.561 0.1
4120 0.0811
19952.62 0.0546
82610 0.027663
234422.9 0.0141
933254.3 0.001

Hydraulic_Conductivity_Function_For_Soil_Type_#1

NumberOf a SatHydCond Imp. Factor
DataPoints (cm/s)
10 2.00E-06 0
Suction HydraulicConductivity
(kPa) (cm/s)
1 1.00E+00
10 8.67E-01
15.84893 2.75E-01
25.70396 7.45E-02
50.11872 8.71E-03
104.7129 1.12E-03
251.1886 1.58E-04
1584.893 2.00E-06
21877.62 1.29E-08
467735.1 7.59E-11

Thermal_Conductivity_Function_For_Soil_Type_#1

NumberOf WaterContent
"DataPoints Type(1=Gravimetric,2=Volumetric) "
10 1
Water Thermal
Content Conductivity

(dec) (W/m C)
0.005 0.41
0.02 0.8
0.04 1.35
0.06 1.61
0.08 1.72
0.1 1.79
0.12 1.86
0.14 1.92
0.16 1.99
0.16872 2.02

Specific_Heat_Function_For_Soil_Type_#1

NumberOf WaterContent
"DataPoints Type(1=Gravimetric,_2=Volumetric) "
10 1
Water Specific
Content Heat
(dec.) (J/m^3-C)
0.005 1469520
0.02 1601400
0.04 1714440
0.06 1827480
0.08 1940520
0.1 2053560
0.12 2147760
0.14 2260800
0.16 2355000
0.16872 2392680

Soil_Type_#2

===

Porosity Specfic
Gravity
0.336 2.7

Moisture_Characteristic_Curve_For_Soil_Type_#2

NumberOf Mv WaterContent
"DataPoints (1/kPa) Type(1=Gravimetric,2=Volumetric) "
15 0.009061 1
Suction WaterContent
(kPa) (dec)
1 0.1982
5 0.1975
10.71519 0.196
21.37962 0.1943
38.01894 0.1911
75 0.1854
144.544 0.1771
295.1209 0.1669
524.8075 0.158
1513.561 0.1293
4786.301 0.0987
19952.62 0.0546
82610 0.027663
234422.9 0.0141
933254.3 0.001

Hydraulic_Conductivity_Function_For_Soil_Type_#2

NumberOf a SatHydCond **Imp. Factor**
DataPoints (cm/s)
10 2.00E-08 0
Suction HydraulicConductivity
(kPa) (cm/s)
1 1.00E+00
10 8.67E-01
44.66836 5.37E-01
102.3293 7.45E-02

281.8383	8.71E-03
794.3282	1.12E-03
2187.762	1.58E-04
16982.44	3.80E-06
102329.3	2.19E-07
436515.8	3.98E-08

Thermal_Conductivity_Function_For_Soil_Type_#2

NumberOf WaterContent
"DataPoints Type(1=Gravimetric,2=Volumetric) "
10 1
Water Thermal C)
Content Conductivity
(dec) (W/m
0.005 0.41
0.02 0.8
0.04 1.35
0.06 1.61
0.08 1.72
0.1 1.79
0.12 1.86
0.14 1.92
0.16 1.99
0.16872 2.02

Specific_Heat_Function_For_Soil_Type_#2

NumberOf WaterContent
"DataPoints Type(1=Gravimetric,_2=Volumetric) "
10 1
Water Specific
Content Heat
(dec.) (J/m^3-C)
0.005 1469520
0.02 1601400
0.04 1714440
0.06 1827480
0.08 1940520
0.1 2053560
0.12 2147760
0.14 2260800
0.16 2355000
0.16872 2392680

Soil_Type_#3

Porosity Specfic
Gravity
0.4 2.65

Moisture_Characteristic_Curve_For_Soil_Type_#3 WaterContent

NumberOf Mv
"DataPoints g (1/kPa) Type(1=Gravimetric,2=Volumetric) "
15 0.009061 1
Suction WaterContent
(kPa) (dec)
1 0.2394
2.187762	0.2344
4.168694	0.2229
4.897788	0.1888
5.495409	0.1697
5.888437	0.159
6.309573	0.1474
7.079458	0.104
8.317638	0.08
12.58925	0.06
34.67369	0.04
134.8963	0.0305
3630.781	0.0182

57543.99 0.008
933254.3 0.001

Hydraulic_Conductivity_Function_For_Soil_Type_#3

NumberOf SatHydCond
DataPoints (cm/s)
10 3.00E-03 0
Suction HydraulicConductivity
(kPa) (cm/s)
1 1.00E+00
3.235937 8.67E-01
5.058247 5.75E-01
8.222426 2.09E-02
11.61449 2.51E-03
14.79108 5.50E-04
19.6336 6.17E-05
28.31392 3.80E-06
38.90451 2.19E-07
120.2264 1.17E-11

Thermal_Conductivity_Function_For_Soil_Type_#3

NumberOf WaterContent
"DataPoints h Type(1=Gravimetric,2=Volumetric) "
10 1
Water Thermal
Content Conductivity
(dec) (W/m^2)
0.005 0.41
0.0481 0.963
0.0662 1.119
0.0786 1.224
0.0883 1.313
0.1 1.396
0.12 1.537
0.14 1.657
0.16 1.761
0.25 2.172

Specific_Heat_Function_For_Soil_Type_#3

NumberOf WaterContent
"DataPoints i Type(1=Gravimetric,_2=Volumetric) "
10 1
Water Specific
Content Heat
(dec.) (J/m^3-C)
0.0066 1469520
0.02 1530000
0.04 1625000
0.06 1719000
0.08 1806000
0.1 1909000
0.12 1988000
0.14 2059000
0.1874 2262000
0.25 2475000

Daily_Data_Input_File_For_SoilCover

****

"Should_SoilCover_Use_Specified_Surface_Temperatures,_or_Calculate_it's_Own?_(0=Specified,1=Calculate)"
0

Total Temperature Rel.Humidity Latitude NumberOfDays
DaysOfData Lag Lag Past_January_1st

181 0 0 56 320

Daily_Data

Day Max Min Net Max Min Wind TopBoundryCondition BotBoundryCondition Top Bottom Run
AirTemp AirTemp Radiation RH RH Speed Type Value Duration Type Value Temperature Temp NextDayTemp
(C) (C) (Mg/m^2-day)(dec) (dec)(km/hr)
"[(0=SUC,1=VWC,2=GWC,3=Flux,4=PE)(0=kPa,1=dec.,2=dec,3=mm/day,4=mm/day)]" (hrs.)

1	-0.2	-0.2	0	0	0	0	3	0.00E+00	24	3	0.00E+00	-0.2	15	**1.5**
2	1.4	1.4	0	0	0	0	3	0.00E+00	24	3	0.00E+00	1.5	15	**-0.7**
3	-0.9	-0.9	0	0	0	0	3	0.00E+00	24	3	0.00E+00	-0.7	15	**-3.7**
4	-4	-4	0	0	0	0	3	0.00E+00	24	3	0.00E+00	-3.7	15	**-3**
5	-3.4	-3.4	0	0	0	0	3	0.00E+00	24	3	0.00E+00	-3	15	**0.1**
6	-0.4	-0.4	0	0	0	0	3	0.00E+00	24	3	0.00E+00	0.1	15	**-1.1**
7	-1.7	-1.7	0	0	0	0	3	0.00E+00	24	3	0.00E+00	-1.1	15	**-0.9**