BIRZEIT UNIVERSITY
FACULTY OF GRADUATE STUDY

SIMULATION OF GREENHOUSES WITH PHASE CHANGE MATERIAL S (PCMs)

By

Atyah M.Najjar

Submitted in Partial Fulfillment of the Requirement for the Master Degree in Sienntific Computing From the Graduate Faculty at Birzeit University

Supervisor

Dr.Afif Hasan
Mechanical Engineering Department

Birzeit, Palestine
Spring 2006

SIMULATION OF GREENHOUSES WITH PHASE CHANGE MATERIAL S (PCMs)

By

Atyah M.Najjar

This thesis was successfully defended on January 3, 2006 and approved

by:

Committee Members	Signature
1. Dr. Afif Hasan	…….
2. Dr. Hasan Shibli	……..
3. Dr. Jameel Harb	…….

(ABSTRACT)

The problem investigated in this thesis is how to influence the internal air temperature of the greenhouse.The internal air temperature depends on the outside weather conditions and the heat gains and losses from and to the greenhouse .

The main objective in this thesis is the simulation of greenhouse with PCMs.The PCM problem was solved as one dimensional problem in a rectangular container (rectangular coordinates) for two types of boundary conditions; first with constant end wall temperature, and second with constant temperature of fluid (air) at the end walls.

A gutter connected model of the greenhouse without PCM was simulated in 23 July for AL_Aroub Agricultural School, 12 km north of Hebron. Its outside temperature and its direct solar radiation was used in that day. It was noticed that by changing the cover of the greenhouse and the outside wind speed, the inside air temperature of this model changes.

After that, this greenhouse model was simulated with different PCMs inside it. Assuming that the rectangular containers that are filled with PCM and the total convectional surface area equals 400 m^2.

Three days all over the year were chosen for the simulation. These were the 23rd of July, 21st of January, and 21st of April; one was in Summer, the second was in Winter, and the third was in Spring. The model was simulated with different PCMs and different cover materials of the greenhouse model.

The main conclusion is that the PCMs can be used to control the air temperature inside the greenhouse. In July, the PCM will lower the air temperature at mid-day time. Whereas, in winter time the PCM will raise the air temperature of the greenhouse at the night time.

To raise the convectional heat transfer at the PCM surface fans must be used. Simulation using fans gave good results in controlling greenhouse temperature at summer and winter days .

The types of cover materials investigated in the simulation are double polyethylene, single polyethylene, fiberglass, and glass. Each of them has its own thermophysical properties; the values of thermal conductivity and the transmittance coefficient of double polyethylene are lower than the values of remaining two covers, while the fiberglass has a larger transmittance coefficient than the single polyethylene, and single polyethylene has a larger thermal conductivity larger than the fiberglass.

Using double polyethylene, will lower the heat loss from

the greenhouse, and this will prevent the lowering of the inside air temperature, especially in winter.

ملخص

المشكلة التي بحثت في هذه الرسالة هي التحكم في درجة حرارة البيت الزجاجي المستخدم لأغراض الزراعة ،حيث تَعتمدُ درجة حرارة الهواء داخل البيت الزجاجي بشكل مباشر على الأحوال الجويةِ خارجه، وعلى ضوء ذلك يحدث كسب وخسارة للحرارةَ مِنْ وإلى البيت الزجاجي.

إنّ الهدفَ الرئيسيَ في هذه الرسالة هو محاكاةُ البيت الزجاجي الذي بداخله مادة لها درجة حرارة إنصهار مناسبة .

في البداية قمت بحل وتمثيل انتقال الحرارة إلى مادة تغير حالتها ، حيث ان النموذج الذي مثلت انتقال الحرارة خلاله هو وعاء على شكل متوازي مستطيلات مصنوع من مادة معدنية وبداخله هذه المادة بحيث يفترض عزله من جميع الأوجه إلا من وجهين متقابلين حتى تنتقل الحرارة بداخله في اتجاه واحد.

هذه المشكلة حلت بشرطين حديين مختلفين ، الأول تم افتراض تثبيت حرارة الوجهين ، والثاني على اعتبار أن انتقال الحرارة بالحمل بين الهواء وسطحي المادة المخزنة للحرارة.

كذلك محاكاة نموذج افتراضي للبيت الزجاجي من نوع المزا ريب المُوَصَّلِةعلى فرض وجوده في منطقة العروب 12 كم شمال الخليل واستعمال درجة الحرارة و الإشعاع الشمسي المباشر لهذه المنطقة و كذلك ملاحظة تأثير نوعية غطاء البيت الزجاجي و سرعة الهواء خارجه في حرارة الهواء بداخله.

و في النهاية تم محاكاة هذا النموذج للبيت الزجاجي على افتراض و جود مادة تغير حالتها بداخل أوعية معدنية بحيث توضع فوق الأرض داخل البيت الزجاجي وبمساحة 400م^2، وعلى افتراض وجود البيت في منطقة العروب 12 كم شمال الخليل تم محاكاة هذا البيت وتمثيل درجة الحرارة لهوائه خلال ثلاثة أيام مختلفة 23 تموز و21 شباط و 21 كانون الثاني.

من خلال النتائج هذه المادة تلطف حرارة الهواء في البيت الزجاجي في الصيف وفي أيام الشتاء المشمسة نهارا ترفع درجة حرارة الهواء ليلا.

لزيادة معامل الحمل الحراري لسطح المادة التي تحول حالتها يجب استعمال مراوح، حيث أن نتائج المحاكاة عند استعمال المراوح كانت جيدة في أيام الصيف و الشتاء.

كذلك تم محاكاة البيت الزجاجي ، بتغيير الغطاء المستعمل حيث الأغطية التي افترضت هي الزجاج ،أو طبقة واحدة بولي ايثلين ،أو طبقتين بولي ايثلين ، أو الألياف الزجاجية، كل من هذه الأغطية له خصائص حرارية تميزه عن الآخر،حيث أن البولي إيثلين ذو الطبقتين له موصلية حرارية ومعامل نفاذية للضوء أقل من البولي ايثلين ذو الطبقة الواحدة و الالياف الزجاجية،بينما معامل نفاذية للضوء للالياف الزجاجية اكبر من تلك التي للبولي ايثلين ذو الطبقة الواحدة، و الموصلية الحرارية للبولي ايثلين ذو الطبقة الواحدة اكبر من تلك التي للالياف الزجاجية.

استعمال البولي ايثلين ذو الطبقتين يخفف تسرب الحرارة خارج البيت الزجاجي وبذلك يمنع انخفاض درجة حرارة الهواء داخل البيت الزجاجي خصوصاً في الشتاء.

Acknowledgment

I would like to offer my sincere thanks to Dr. Afif Hasan , both for his numerous efforts in helping me to complete this thesis, and for being my advisor. His insight encouragement and guidance over the past two years have been invaluable.

I would also like to thank Professor Hasan Shibli, the head of the Computational Science Department at Berziet university for his encouragement .

Moreover, my greatest thanking is to be delivered to my committee members for their help and guidance. I am not forgetting Dr. Aziz Shawabkeh for helping me to get this process started. I also would like to thank Mr. Mohammed Hamarsheh for his support, and for the opportunity of teaching me the Operating System course, an experience I have greatly enjoyed.

For her unwavering support and companionship, I want to offer special thanks to my wife, Suha Huneihin.

Finally, I would like to thank my brother Subhi Najjar and my dearest friend Salah Awad for their support in all of my endeavours.

Contents

Nomenclature:

A	Total cover surface area of the greenhouse [m^2].
ACH	Number of air exchanges .
A_i	Greenhouse cover surface area $_i$ (south ,north ,east ,west, or roof) [m^2].
A_{pcm}	Total surface area of phase change material [m^2].
c	Specific heat of phase change material[J kg^{-1} $^\circ C$].
c_{air}	Specific heat of air [J kg^{-1} $^\circ C^{-1}$].
c_{air}	Specific heat of air [J kg^{-1} $^\circ C^{-1}$].
f_l	Liquid fraction .
H	Total enthalpy [J kg^{-1}] .
h	Sensible enthalpy [J kg^{-1}].
h_{pcm}	Heat transfer coefficient from phase change material surface[W m^{-2} $^\circ C^{-1}$] .
I	solar radiation intensity beneath the covering material [W m^{-2}].
I_o	solar radiation intensity beneath the covering material [W m^{-2}].
k	Conductivity [W m^{-1} $^\circ C^{-1}$].
L	Latent heat of fusion [J kg^{-1}].
Q	heat gain or loss [W].
Q_{inf}	infiltration heat losses [W]
Q_{solar}	solar energy entering the greenhouse useful to increase internal temperature [W].
Q_t	Heat transmission losses through walls and roof [W].
S	Solar radiation [W m^{-2}].
T	Temperature of phase change material [$^\circ C$].
t	Time [s].
T_{air}	Temperature of inside air .

T_i	Temperature of node i in phase change material [°C].
T_m	Melting temperature of phase change material [°C].
T_o	Initial temperature [°C] .
T_{out}	Temperature of outside air [°C].
T_w	Wall temperature of phase change material [°C].
U	Over all Heat transfer coefficient [W m^{-2}°C^{-1}].
V	Outside air velocity [m s^{-1}].
v	Greenhouse volume[m^3]
ρ	Density of phase change material [kg m^{-3}].
ρ_{air}	Density of air [kg m^{-3}].
α	Thermal diffusivity of phase change material [m^2 s^{-1}].
Δx	Space increment in phase change material.
Δt	Time step [s] .
γ	Constant of the proportion of solar radiation entering the
τ	Transmittance of greenhouse cover to direct solar radiation.

Subscripts used :

air	Inside greenhouse air
ACH	Air change per hour.
cover	Greenhouse cover.
fluid	Hot or cold fluid .
inf	Infiltration.
l	Liquid.
LTHS	A latent heat storage system
m	Melting temperature.
old	Previous time step.
out	Outside.
pcm	Phase change material.
pcm17	Pcm has melting temperature 17 °C.

poly	**Polyethylene.**
solar	**Solar radiation.**
w	**Wall.**

List of figures :

List of Tables:

Chapter 1

Introduction

Dr. James Langer, Chairman of the DOE-NSF National Workshop on Advanced Scientific Computing, 30-31 July 1998, states: [1]

". . . scientific computation has reached the point where it is on a par with laboratory experiment and mathematical theory as a tool for research in science and engineering. The computer literally is providing a new window through which we can observe the natural world in exquisite detail."

Indeed, we now have the ability to solve many problems arising in the natural sciences that would have been deemed intractable only a few decades ago. It cannot be overstressed; however, that scientific computation is a research tool not different from any other experimental apparatus.

Langer continues:[1]

"It is essential to recognize that numerical simulation enhances, and does not substitute for, experimental and theoretical research. Meaningful simulations are based on reliable experimental and theoretical inputs, and their outputs are useful only if validated in the laboratory or the manufacturing plant .The best scientific simulations lead to new

theoretical understanding and to new experimental discoveries."

It is in this interdisciplinary frame of mind that this thesis covers the development and computational solution of a numerical intractable problem taken from mechanical engineering.

By definition a greenhouse is a structure covering ground with transparent material that utilizes solar radiation for growing a plant, glazing materials used in solar greenhouses should allow the greatest amount of solar energy to enter into the greenhouse while minimizing energy loss, using the greenhouse for growing plants will return a profit to the owner risking time and capital [2]. The greenhouse is used for overcoming climatic adversity, where suitable environment is maintained and created inside it for optimum plant growth, such that you can plant tomatoes in winter.

Although greenhouses provide a protected environment for plants, they themselves are in contact with the outside environment, which leads to energy transfer between the inside and the outside of the greenhouses.

Hence, in order to keep the inside conditions suitable against varying outside conditions, the greenhouses have to be heated, cooled, and ventilated depending on outside climatic conditions.

Since the primary aim of protected cultivation is to increase profits by improving product quality and increasing crop yield, the climate control system must be adequate to maintain the desired day or night temperature, and must be as cheap and as reliable as possible.

Controlling the greenhouse temperature, can be performed using different systems and energy sources, such as: fossil fuels, and phase change materials (pcms). Each of these systems has its advantages and disadvantages in the greenhouse heating system.

Heating systems utilizing fossil fuels might have low initial cost, but their running costs are high, as well as causing pollution to the environment. On the other hand pcms energy storage systems might have high initial costs but low running costs and are pollution free, safe to use, and storing high energy, but their disadvantage is their low conductivity which results in lowering the amount of energy stored or released from them .

In this thesis, a PCM system was designed for a typical Palestinian greenhouse, then a computer simulation of the heated greenhouse was made in order to further understand the response of the greenhouse to changes in outside conditions, and whether the heating system used can cope with these changes, and maintain the desired conditions. Pcms undergo

phase change in its state, and can store large amount of heat and release it later when needed. Scientists developed commercial kinds of pcms, that are used as heat storage devices, these devices with pcms are used in houses and greenhouses. The pcm problem was taken to be one dimensional, and solved by the familiar numerical solution which is the enthalpy formulation [3].

The model of the PCM is a rectangular stainless steel container that filled with pcm, and insulated from all sides except two opposite sides. The PCM problem was solved as one dimensional problem for two boundary conditions; constant temperature at the end walls, and constant fluid (air) temperature at the end walls; which is a convectional heat transfer boundary condition. Two different computer programs were written in Java language to simulate and solve the problem. Then a model of greenhouse was chosen as gutter connected model, the outside conditions were chosen at Al-Aroub 12 km north Hebron.

The heat balance equation of the greenhouse without PCM inside it, was solved numerically with a computer program in Java language. The outside conditions were taken in Al-Aroub, the cover of the greenhouse and the outside air velocity were changed in the greenhouse model to find their effect on the inside air temperature of the greenhouse.

Finally the mathematical model of the greenhouse with pcm was solved numerically and a computer program was written in Java language to solve the problem for three days in different months 23th July, 21th February, and 23th April. The greenhouse was modeled as gutter connected and a rectangular pcms was modeled to be distributed inside the greenhouse, the mathematical model of the greenhouse with pcm was solved for different pcms and different cover materials of the greenhouse.

Chapter 2

Phase Change Materials (PCMs)

2.1 Types of heat storage:

Energy storage devices are essential whenever the supply and consumption of energy varies independently with time. These devices have three methods of storing thermal energy: sensible, latent, and thermochemical heat, or cold storage.

Thermochemical storage offers an order of magnitude larger heat storage capacity compared to sensible heat storage. However, this technology is still in the development phase. A latent heat storage system (LHTS) is preferable to sensible heat storage in applications with a small temperature variation because of its nearly isothermal storing mechanism and high storage density.

2.1.1 Sensible heat storage

Storing the heat inside the material without changing its phase like concrete, rocks, or metals, these materials cannot be heated or cooled isothermally.

Rocks can be used to store the solar heat enters the greenhouse, also brick or concrete-filled cinder block walls at the back (north side) of the greenhouse can also provide heat storage.

2.1.2 *Latent heat storage:*

Storing the heat inside a material, that undergoes phase change, like paraffin, salt hydrates, which are called PCMs they can be heated or cooled isothermally and non isothermally.

2.1.3 Thermochemical heat

Thermochemical systems absorb and release the energy by breaking and reforming molecular bonds in a completely reversible chemical reactions. A thermochemical reaction requires large quantities of energy input, at elevated temperatures the energy storing reaction reverses, this reaction are typically unreactive at ambient temperatures, an example of a thermochemical reaction is water dissociation into hydrogen and oxygen. [4]

2.2 what is PCM:

It is a chemical material, solid or liquid or powder at the room temperature. When the pcm undergoes its phase change a large amount of energy is required or released, this energy is called the latent heat. The pcm can undergo its phase change thousands of times without affecting its properties.

2.3 Advantages *and disadvantages of PCMs*

The advantage of PCM storage compared to sensible heat storage systems, such as water storage, is its potential to store

large amounts of heat during its phase change with only a small temperature variation. The main disadvantage of pcm is that it has low heat-conductivity (k), the heat-conductivity of paraffins, varies between 0.1 and 0.2 W m^{-1} K^{-1} and that of salt hydrates between 0.4 and 0.6 W m^{-1} K^{-1}, depending on the material. This low conductivity will affect the charging and discharging processes; it lowers the amount of heat stored or discharged. [5]

2.4 Types of PCMs

Pcms may be inorganic or organic materials. The phase changes comprise predominantly solid-liquid transitions for thermal storage applications in buildings.

2.4.1 Inorganic pcms

Early efforts in the development of latent heat storage materials used inorganic PCMs. These materials were studied extensively in the early stages of research into PCMs in order to increase the amount of energy stored inside it through their phase transitions, these materials are salt hydrates, including Glauber's salt (sodium sulphate decahydrate). The thermal properties of these materials are shown in Table 2.1.

These pcms have some attractive properties including:

1. high latent heat values reach to 290 $kJ\ kg^{-1}$.
2. inflammability.

However, inorganic pcms have some unsuitable

characteristics. These include corrosiveness, instability, improper re-solidification, and a tendency to supercool. As they require containment, they have been deemed unsuitable for impregnation into porous building materials. Thus, Unsuitable characteristics have led to the investigation of organic pcms for this purpose.

Table 2.1 Salt hydrate PCMs [6].

PCM	Melting Point [°C]	Heat of Fusion (kJ kg^{-1})
$KF.4H_2O$ Potassium fluoride tetrahydrate	18.5	231
$CaCl_2.6H_2O$ Calcium chloride hexahydrate	29.7	171
$Na_2SO_4.10H_2O$ Sodium sulphate decahydrate	32.4	254
$Na_2HPO_4.12H_2O$ Sodium orthophosphate dodecahydrate	35.0	281
$Zn(NO_3)_2.6H_2O$ Zinc nitrate hexahydrate	36.4	147

2.4.2 Organic PCMs :

Organic PCMs have a number of preferable characteristics which make them useful for latent heat storage :

1. They are more chemically stable than inorganic substances.
2. They melt congruently, which occurs when the solid phase composition is the same as the liquid phase composition, and super cooling does not pose as a significant problem, when super cooling occurs the temperature of the PCM drops well below the melting point before freezing initiates,

which prevents the withdrawal of heat from the PCM.

3. They have been found to be compatible with and suitable for absorption into various building materials.

4. However, the initial cost of organic PCMs is higher than that of the inorganic type, but the installed cost is competitive.

5. Self-nucleation; melt congruently with low vapor pressure.

However, these organic materials have unsuitable properties. The most significant undesirable characteristics, are that: they are flammable, and they may generate harmful fumes on burning.

Other problems, which can arise in a minority of cases, are a reaction with the products of hydration in concrete, thermal oxidative ageing, smell and an appreciable volume change (expansion). Appropriate selection and modification have now eradicated many of these undesirable characteristics. "It has been found; that the thermal oxidative ageing of pcms concerned, can be inhibited by the use of a proper antioxidant" [6], to assess the flammability, and fume generation of some of the more effective pcms such that a fire rating may be established. Also The most promising selection of these organic pcms is shown in Table.2.2 .

Table.2.2 Organic pcms .[6]

PCM	Melting Point [°C]	Heat of Fusion (kJ/kg)
$CH_3(CH_2)_{16}COO(CH_2)_3CH_3$ Butyl stearate	19	140
$CH_3(CH_2)_{11}OH$ 1-dodecanol	26	200
$CH_3(CH_2)_{12}OH$ 1-tetradecanol	38	205
$CH_3(CH_2)_n\ CH_3$ paraffin	20- 60	200
(45% $CH_3(CH_2)_8COOH$ 55% $CH_3(CH_2)_{10}COOH$). 45/55 carpic-lauric acid	21	143
$CH_3(CH_2)_{12}COOC_3H_7$ Propyl palmitate	19	186

Though from tables 2.2 and, 1.2 the energy storage capacities of the organic pcms are generally lower than those of the inorganic pcms.

Fig.2.1 Organic pcms (paraffin wax)

2.5 Uses of PCMs :

The first use of pcms was in British trains to prevent them from becoming too cold. Phase change material (pcm) storages - organic or non organic- are thermal storage materials. These

can be used to balance temporary temperature alternations and to store energy in several practical application areas for electronics, automobile industry and buildings. In current telecommunication electronics, both portable and larger scale thermal transients that occur due to temporarily varying power dissipation are customary.

During the change in state, large quantities of latent heat are required to be absorbed to allow the material to change from a solid to a liquid,

this large quantity of heat is released in the discharging process that will be used in heating systems like greenhouses .

The process of phase change can be repeated over an unlimited number of cycles with little change to their physical or chemical properties. [5]

2.6 PCM problem :

2.6.1 History of Stefan problem

"In the late 19th century J. Stefan formulated the problem of finding the temperature distribution and freezing front history of a solidifying slab of water" [7]. In the last 30 years, Stefan problem has been developed to include such complex phenomena as the solidification of alloy systems and melting due to laser irradiation. Melting and freezing are involving in a wide range of technologies, encompassing such diverse

applications as freeze drying of foods, growth of crystal, cryo-preservation of biological cells as well as latent heat thermal energy storage system .

phase change material problem is non-linear and its principal difficulty lies in the fact that one of its unknown is the region in which it is to be solved, for this reason it is called a "moving boundary problem (or Stefan problem)." For most reasonable phase change processes the heat equation and its differential expression hold within the liquid or solid and a jump condition expressing energy conservation prevails along the curves separating solid from liquid. Fuzzy or fat regions are sometimes seen in lieu of sharp interfaces separating solid from liquid.

2.6.2 Description of PCM problem :

During the phase change in a pcm storage system, the solid–liquid interface moves away from the heat transfer surface and the surface heat flux decreases due to the increasing thermal resistance of the molten or solidified medium.

Heat transfer in pcm storage is a transient, non-linear phenomenon with a moving solid-liquid interface, generally referred to as the "moving boundary" problem.

"Nonlinearity is the source of difficulties in moving

boundary problems" [5,8] and, therefore, phase change problems can be solved analytically for simple problems assumed for physical situations that have a simple geometry and simple boundary conditions. Stefan problem is a one-dimensional moving boundary problem, which has a well known precise analytical solution, and was originated by Neumann [9,10].

Some analytical approximations like the quasi-stationary approximation, perturbation methods, the Megerlin method and the Heat-balance integral method, have been produced for one-dimensional moving boundary problems with different boundary conditions. [5].

In these methods, it has been assumed that the melting or solidification temperature is constant. However, technical grade paraffin, for example, has a wide temperature range at the points where melting and solidification occur. Also, the aforementioned methods are only suitable for calculating semi-infinite or infinite storage. However in reality storages are finite and have to be handled three- or at least two-dimensionally so as to achieve a sufficiently accurate solution.

Therefore, numerical methods have to be used to achieve a sufficiently accurate solution for heat transfer in pcm storage. Phase change problems are usually solved with finite difference or finite element methods in accordance with the numerical

approach. The phase change phenomenon has to be modeled separately due to the non-linear nature of the problem. A wide range of different kinds of numerical methods for solving pcm problems exist.

The most common methods used are the enthalpy method and the effective heat capacity method, these methods are able to use pcms with a wide phase change temperature range. "The enthalpy method is introduced in many references as a numerical method for solving phase change problems" [5,9,11]. The method is based on the weak solution of partial differential equations.

[Voller V R.(1990)] have presented a simple development of the conventional enthalpy formulation which leads to very accurate solutions, the extension of this technique to two-dimensional problems is demonstrated by using an explicit method.

" A relative accuracy of 0.1 % has been obtained in the comparison between numerical results achieved by other authors and the results achieved with the developed method (enthalpy method)" [10].

To analyze numerically the thermal performance of latent heat storage

the enthalpy formulation with a fully implicit finite difference method is familiar method which can be used, in this method

the conduction is the sole heat transfer mode for a one dimensional solidification or melting processes.

PCM melts more quickly than it solidifies because natural convection speeds up the melting of pcm. In solidification the heat transfers by conduction in solid pcm out from the storage. If there is a temperature gradient in liquid pcm, natural convection exists in the liquid–solid interface. But even very strong natural convection has a negligible effect on the solid–liquid interface position compared to the effect of heat conduction in solid pcm [9,12]. PCM solidifies on the heat transfer surface and acts as a self-insulator because of low heat conductivity. In practice some kind of heat transfer enhancement technique has to be used in latent heat storage system(LTHS) because of the low heat conductivity of the pcm.

Heat transfer in LTHS can be enhanced by the following techniques:

(a) Active methods such as agitators ,vibrators, scrapers and slurries [7].

(b) Using microencapsulated PCM [10].

(c) Using PCM containing dispersed high conductivity particles or lessing rings [8].

(d) Using PCM graphite composite material [13].

(e) Using extended surfaces such as fins and honeycombs [7,5,8,12].

2.6.3 The numerical solution of phase change problem (moving boundary problem):

2.6.3.1 Mathematical Formulation

2.6.3.1.1 Overview of the Phenomena Involved in a Phase Change

Every pcm has certain melting and freezing temperature they may be the same or differs, phase transitions occurs when a solid melts (melting process) or a liquid solidifies (freezing process). Such a change of phase involves heat and often also mass transfer, possible supercooling, absorption or release of latent heat, changes in thermophysical properties, surface effects, etc.

The transition from one phase to the other occurs due to the absorption or release of the latent heat at some temperature, at which the stability of one phase breaks down in favor of the other according to the available energy, this phase change temperature, T_m, depends on pressure. Under a fixed pressure, T_m may be a particular fixed value characteristic of the material or a function of other thermodynamic variables.

Since most solids are crystalline, their particles (atoms, molecules or ions) are arranged in a repetitive lattice structure extending over significant distances in atomic terms. Since formation of a crystal may require the movement of atoms into the solid lattice structure, it may well happen that the

temperature of the material is reduced below T_m without formation of solid. Thus, supercooled liquid may appear; such a state is thermodynamically metastable. Note that no such structuring is required in the melting process.

When a pcm starts to melt or solidify, solid and liquid phases are coexist at the same time, these phases will be separated by a region which is called the interface.

The interface is that region where solid and liquid coexist, its thickness may vary from a few Angstroms to a few centimeters depending on the nature of the pcm, and its microstructure may be very complex. This interface appears (locally) planar and of negligible thickness when pure materials solidifying under ordinary freezing conditions at a fixed T_m , it may be thought of as a sharp interface. In other cases, typically resulting from supercooling, the phase transition region may have apparent thickness and is referred to as a mushy zone.

Most thermophysical properties of a material undergo sudden changes at T_m, these changes complicate the mathematical problems because they induce discontinuities in the coefficients of differential equations like the volume, density,…etc . The most fundamental and pronounced effects are due to changes in density; typically in the range of 5 to 10% but can be as high as 30%. The density variation with

temperature induces flow by natural convection in the presence of gravity, rapidly equalizing the temperature in the liquid and greatly affecting heat transfer. In microgravity, there is no natural convection but Marangoni convection, due to surface tension effects, may arise and dominate heat transfer.

2.6.3.1.2*The assumptions of the mathematical formulation*

Formulation of the Stefan problem for the mathematical description of a melting or freezing problem, the following assumptions are made

Table 2.3 Assumptions to simplify PCM's problem [7]

Physical factors involved in PCM's process	Simplifying assumptions for the problem
Heat and mass transfer by conduction,convection radiation,with possible gravational ,elastic ,chemical and electromagnetic effects	Heat transfer isotropically by conduction only
Release or absorption of latent heat	Latent heat is constant it is absorbed or released at the phase change temperature.
Variation of phase change temperature	Phase change temperature is affixed known temperature ,and is property of material
Nucleation difficulties ,supercooling	Assume not present
Interface thickness and structure	Locally planner and sharp
Surface tension and curvature effects at interface	Assume insignificant
Density changes	Assume constant for solid and liquid
Variation of thermophysical properties	Constant for liquid and solid (c,k)

The situation arises, very often in real engineering applications, with the result that the Stefan problem is by far the most frequently applied model of a phase change process.

Based on the aforementioned assumptions the governing conservation equations for a one-dimensional pcm problem are formulated [7]

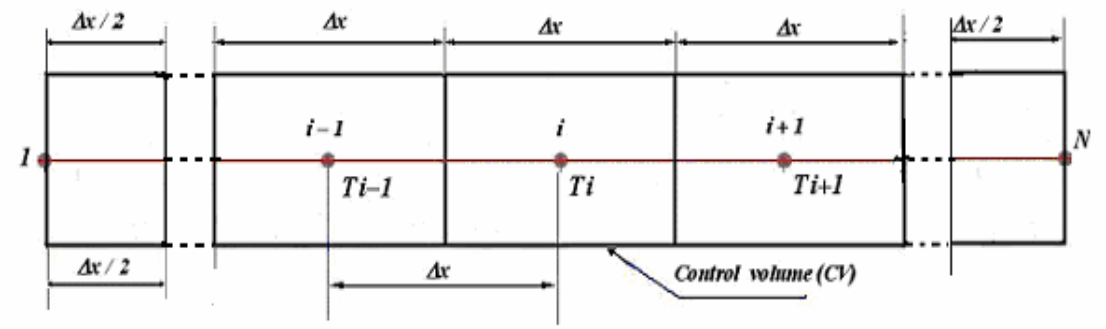


Fig.2.2 Discretization domain for one dimensional PCM problem [3].

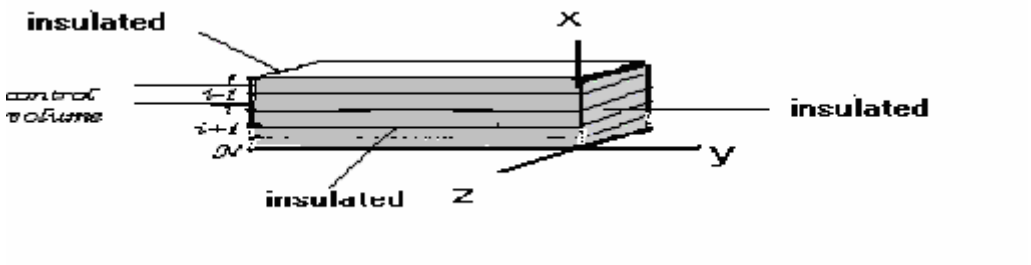


Fig.2.3 pcm's rectangular container

2.6.3.1.3 The enthalpy formulation:-

The enthalpy formulation for the conduction -controlled phase change material can be written as [3,14].

The general equation governing the conductive heat transfer in the PCM is therefore

$$\frac{\partial H}{\partial t} = div \left(\frac{k}{\rho} grad \left(T \right) \right) \quad (1)$$

By splitting the enthalpy H into sensible and latent heat components

$$H = h + L \cdot f_l \tag{2}$$

where:

$$h = \int_{Tm}^{T} c \; dT \tag{3}$$

for the problem of an isothermal phase change, the local liquid fraction f_l is defined as :

$$f_l = \begin{cases} 0 & \text{if } T < T_m \text{ , (Solid)} \\ 1 & \text{if } T > T_m \text{ , (Liquid)} \\ \text{through the melting or solidification processes of the CV} \\ f_l \text{ is time dependent and its value between (0) and (1)} \\ \text{then } T = T_m \text{ , (Interface)} \end{cases} \tag{4}$$

substituting Eq.2 into Eq.1 gives :

$$\frac{\partial h}{\partial t} = div\left(\frac{k}{\rho} grad\, T\right) - L\frac{\partial f_l}{\partial t} \tag{5}$$

Eq.5 represents together with Eq.3 and Eq.4 and the suitable boundary and initial conditions ,the mathematical model of conduction controlled isothermal phase change which assumes no convectional heat transfer inside the pcm.

The problem is one-dimensional isothermal phase change of the PCM encapsulated within a rectangular container the governing equation for the PCM follows from Eq.5:

$$\frac{\partial h}{\partial t} = \frac{\partial}{\partial x}\left(\frac{k}{\rho}\frac{\partial T}{\partial x}\right) - L\frac{\partial f_l}{\partial t} \quad (6)$$

For constant ρ and k eq.6 becomes :

$$\frac{\partial h}{\partial t} = \frac{k}{\rho}\frac{\partial^2 T}{\partial x^2} - L\frac{\partial f_l}{\partial t} \quad (7)$$

the fully implicit discritization equation (7) for an internal node (i) Fig.2.2, can be written as follows :

$$\frac{\partial h_i}{\partial t} = \frac{k}{\rho}\left(\frac{T_{i-1} - 2T_i + T_{i+1}}{\Delta x^2}\right) - L\frac{\partial f_l}{\partial t} \quad (8)$$

$$sensible\ \ enthalpy\ \ term \equiv \frac{k}{\rho\,\Delta x^2}\left(T_{i-1} - 2T_i + T_{i+1}\right) \quad (9)$$

$$liquid\ \ fraction\ \ term \equiv L\frac{\partial f_{li}}{\partial t} \quad (10)$$

the key feature of this method is the fact that for an isothermal phase change (the case for most salt hydrates), the temperature of the pcm within a given control volume remains constant and equal to its melting temperature until the PCM (control volume) has melted completely.

Consider first the case when the control volume "*i*" is fully solid ($f_{li} = 0$), or liquid ($f_{li} = 1.0$) in that case:

$$\frac{\partial f_{li}}{\partial t} \equiv 0 \quad (11)$$

From Eq.9 and Eq.10 it follows that :

$$\frac{\partial h_i}{\partial t} \equiv c\frac{\partial T_i}{\partial t} \text{ (only sensible (conduction))} \quad (12)$$

after introducing Eq.11 and Eq.12, Eq.8 reduces to ordinary

heat diffusion equation:

$$c\frac{\partial T_i}{\partial t}=\frac{k}{\rho}\frac{\partial^2 T}{\partial x^2} \tag{13}$$

the fully implicit discretization of the temperature is :

$$\frac{\partial^2 T_i}{\partial x^2}=\frac{k}{\rho\ \Delta x^2}\left(T_{i-1}-2T_i+T_{i+1}\right)+O(\Delta x^2) \tag{14}$$

where $O(\Delta x^2)$ *is the truncation error* *appendix A.6*

and

$$\frac{\partial T_i}{\partial t}=\frac{k}{\rho\, c\, \Delta x^2}\left(T_{i-1}-2T_i+T_{i+1}\right) \tag{15}$$

backward differencing of $\frac{\partial T_i}{\partial t}$ **gives :**

$$\frac{\partial T_i}{\partial t}=\left(\frac{T_i-T_i^{old}}{\Delta t}\right)+O(\Delta t) \tag{16}$$

where $O(\Delta t)$ *is the truncation error of the approximation* *appendix A.6*

from equation.15 then :

$$c\left(\frac{T_i-T_i^{old}}{\Delta t}\right)=\frac{k}{\rho\ \Delta x^2}\left(T_{i-1}-2T_i+T_{i+1}\right)+O(\Delta t)+O(\Delta x^2) \tag{17}$$

drop the truncation errors from equation (17), and rearranging its terms, the fully implicit discretization of the temperature is :

$$-\frac{k.\Delta t}{\rho\, c\, \Delta x^2}\cdot T_{i-1} + (1 + 2\,.\frac{k.\Delta t}{\rho\, c\, \Delta x^2}).T_i - \frac{k.\Delta t}{\rho\, c\, \Delta x^2}\cdot\ T_{i+1} = T_i^{old} \quad (18)$$

$$F_o = \frac{\alpha.\Delta t}{\Delta x^2} \quad ,where\ \alpha = \frac{k}{\rho\, c} \quad \text{(the thermal diffusivity of the PCM)} \quad (19)$$

$$-F_o.T_{i-1} + (1 + 2.F_o).T_i\ - F_o.T_{i+1} = T_i^{old} \quad (20)$$

$$a_{i-1}\cdot T_{i-1} + a_i\cdot T_i + a_{i+1}\cdot\ T_{i+1} = b_i \quad (21)$$

$$a_{i-1} = a_{i+1} = -F_o\ , a_i = 1 + 2\,.F_o\ \ , b_i = T_i^{old} \quad (22)$$

The stable approximations of this method are only obtained if ($F_o \le \frac{1}{2}$).**When the melting or solidification occurs around a certain node 'i' the liquid fraction** f_{li} **lies strictly in the interval [0,1] and recognizing that for an isothermal phase change :**

$$T_i \equiv T_m \quad (23)$$

from Eq.3 :

$$\frac{\partial h_i}{\partial t} \equiv 0 \quad (24)$$

Eq.8 becomes :

$$L\frac{\partial f_{li}}{\partial t} = \frac{k}{\rho\,\Delta x^2}\left(T_{i-1} - 2\,T_i + T_{i+1}\right) \quad (25)$$

backward differencing of the liquid fraction term gives :

$$f_{li} = f_{li}^{old} + \frac{k\ \Delta t}{\rho\ L\,\Delta x^2}\left(T_{i-1} - 2\,T_i + T_{i+1}\right) \quad (26)$$

which is the equation of updating the liquid fractions within

the control volume that undergoing a phase change.

2.6.3.2 Melting and solidification formulation:

The whole domain is partitioned in N equidistant nodes. The control volume (CV) associated to each node (Fig. 2.2) has a thickness Δx, and only nodes 1 and N have half-thickness ($\frac{\Delta x}{2}$).

2.6.3.2.1 The melting process

(A) The pcm is solid state so

$T < Tm$ so $f_{li} \equiv 0$ for all the control volumes the rectangular pcm

(B) The set of linear algebraic Eq(21) solved using the Gauss-Seidel iterative procedure

(C) start of melting if $if\ T_i \geq Tm$ while $T_i^{old} < T_m$ for node "i" the CV start of melting ,at this time the thermal diffusivity $(\alpha = \frac{k}{\rho c} \approx 0)$; The sensible heat of the liquid PCM is taken account in the enhanced latent heat ,so at this time $\frac{\partial T_i}{\partial t} = 0, and\ T_i = T_m$, coefficients of equation(21) must be updated $a_{i+1} = a_{i-1} = 0$, $a_i = 1$, $b_i = T_m$ the calculations of this time step must be recalculated to set the isothermal property of melting CV

and f_{li} **lies strictly in the interval [0,1] ,** $0 < f_{li} < 1$

$$\frac{\partial h_i}{\partial t} = \frac{k}{\rho \Delta x^2}\left(T_{i-1} - 2T_i + T_{i+1}\right) - L\frac{\partial f_l}{\partial t} \tag{27}$$

$$\frac{\partial h_i}{\partial t} \equiv c\frac{\partial T_i}{\partial t} = c\frac{T_m - T_i^{old}}{\Delta t} \tag{28}$$

$$\frac{\partial f_{li}}{\partial t} = \frac{f_{li} - f_{li}^{old}}{\Delta t} \tag{29}$$

$$c\left(\frac{T_m - T_i^{old}}{\Delta t}\right) = \frac{k}{\rho \Delta x^2}\left(T_{i-1} - 2T_i + T_{i+1}\right) - L\left(\frac{f_{li} - f_{li}^{old}}{\Delta t}\right) \tag{30}$$

by arranging equation (26) we get eq.31

$$f_{li} = f_{li}^{old} + \frac{k\,\Delta t}{\rho L \Delta x^2}\left(T_{i-1} - 2T_i + T_{i+1}\right) - \frac{c}{L}\left(T_m - T_i^{old}\right) \tag{31}$$

then f_{li} **recalculated using Eq.31**

(D) the liquid fractions are updated from the temperature field using Eq(26).

(e) end of melting of the CV if $f_{li} \geq 1$ **while** $f_{li}^{old} < 1$ **it indicates that within this time step the control volume has melted completely, the coefficients of Eq(21) reset to**

$a_{i+1} = a_{i-1} = -F_o$, $a_i = 1 + 2.F_o$.

$$b_i = T_m - \frac{L}{c}\left(1 - f_{li}^{old}\right) \tag{32}$$

the calculation within this time step is performed again .

here $\frac{L}{c}(1 - f_{li}^{old})$ **is the amount of heat needed to completely melt the control volume within the time step and which cant**

be used to raise the CV temperature .[3,15]

2.6.3.2.2 The solidification process :

(A) The pcm is in a liquid state so;

$T > Tm$ so $f_{li} \equiv 1$ for all the control volumes of the rectangular pcm and the set of linear algebraic Eq(21)solved using the Gauss-Seidel iterative procedure .

(B) start of solidification if $if\ T_i \le Tm$ while $T_i^{old} > T_m$ for node "i" the CV start of solidification, coefficients of equation 21 must be updated $a_{i+1} = a_{i-1} = 0$, $a_i = 1$, $b_i = T_m$ the calculations of this time step must be recalculated to set the isothermal property of solidification CV ,and f_{li} lies strictly in the interval [0,1] , $0 < f_{li} < 1$ then f_{li} recalculated using Eq(31).

(C) the liquid fractions are updated from the temperature field using Eq(26).

(D) end of solidification of the CV if $f_{li} \le 0 \ \ while \ \ f_{li}^{old} > 0$ it indicates that within this time step the control volume has solidified completely and the coefficients of Eq.21 are reset to $a_{i+1} = a_{i-1} = -F_o$, $a_i = 1 + 2.F_o$, and

$$b_i = T_m - \frac{L}{c}\left(0 - f_{li}^{old}\right) = T_m + \frac{L}{c}\left(f_{li}^{old}\right) \qquad (33)$$

the calculation within this time step is performed again .

here $\frac{L}{c}(f_{li}^{old})$ is the amount of heat needed to completely solidify the control volume within the time step and which cannot be used to low the CV temperature .

$$F_{ol} = \frac{k\,\Delta t}{\rho\, L\, \Delta x^2} \tag{34}$$

$$B_i = \frac{h_{fluid}\,\Delta x}{k} \tag{35}$$

[3,15]

2.6.3.3 Melting and solidification of PCM with constant end walls temperatures .

Rectangular slab of pcm with constant end walls (sides) temperature initially the pcm was in the solid state $T < Tm$ (melting temperature) the walls (two walls were subjected to $T > Tm$), the other walls are insuated($dq \equiv 0.0$), where q is the heat flux [J.m^{-2}].

The problem is a one-dimensional transient heat transfer, to simplify the problem many considerations were taken into account as follows:

1. thermo physical properties of the pcm are constant in solid and liquid (the same).

2.the heat transfer in solid and liquid phases is by conduction only .

3. the convectional heat transfer in the liquid phase was ignored .

4.the problem was taken as isothermal phase change .

5.the effect of gravitational force was ignored.

6. radiation heat transfer was ignored.

7.the melting temperature (*Tm*) is equal to the freezing temperature .

8.heat transfer in one dimension .

the problem was solved by the enthalpy formulation method .

A java program was written to simulate this problem .

The initial and boundary conditions are :

$$T(x,0) = T_0 \quad \textbf{(36)}$$

$$T(0,t) = T(L_x,t) = T_w(t) \quad \textbf{(37)}$$

2.6.3.4 Melting and solidification with convectional condition (constant end wall fluid temperature) .

The previous problem was solved but with convectional heat transfer boundary condition. The same considerations were taken as before, in addition to that the temperature of the fluid that gives or takes the heat from the pcm is assumed constant .

The initial and boundary conditions are :

$$T(x,0) = T_O \quad \textbf{(38)}$$

$$-k\frac{\partial T}{\partial x}\Big|_{x=0} = h(T_f(t) - T(0,t)) \quad (39)$$

$$-k\frac{\partial T}{\partial x}\Big|_{l_x} = h(T(l_x,t) - T_f(t)) \quad (40)$$

$$\frac{\partial T}{\partial z} = \frac{\partial T}{\partial y} = 0 \text{ (insulated)} \quad (41)$$

Table 2.4 Internal nodes coefficients for constant end walls temperature or convectional with constant fluid temperature:[15]

Case	a_{i-1}	a_i	a_{i+1}	b_i	f_{li}
A	$-F_o$	$1+2.F_o$	$-F_o$	T_i^{old}	Solid =0 ,liquid =1
B	0	1	0	Tm	$f_{li}^{old} + F_{ol}\left(T_{i-1} - 2T_m + T_{i+1}\right) - \frac{c}{L}\left(T_m - T_i^{old}\right)$
C	0	1	0	Tm	$f_{li}^{old} + F_{ol}\left(T_{i-1} - 2T_m + T_{i+1}\right)$
D	$-F_o$	$1+2.F_o$	$-F_o$	$b_i = T_m - \frac{L}{c}\left(f_l^* - f_{li}^{old}\right)$	End of melting=1 End of solidification=0

Table . 2.5 Node 1 coefficients for convectional boundary condition [15]

Case	a_0	a_1	a_2	b_1	f_{l1}
A	$-2F_oBi$	$1+2.F_o\ (1+B_i)$	$-2F_o$	T_1^{old}	Solid =0 ,liquid =1
B	0	1	0	Tm	$f_{l1}^{old}+2F_{ol}B_iT_0-2F_{ol}(1+B_i)T_m+2F_{ol}T_2-\frac{c}{L}(T_m-T_1^{old})$
C	0	1	0	Tm	$f_{l1}^{old}+2F_{ol}B_iT_0-2F_{ol}(1+B_i)T_m+2F_{ol}T_2$
D	$-2F_oBi$	$1+2.F_o\ (1+B_i)$	$-2F_o$	$Tm-\frac{L}{c}(f_{l1}^{*}-f_{l1}^{old})$	End of melting=1 End of solidification=0

Table 2.6 Node N coefficients for convectional boundary condition [15]

Case	a_{N-1}	a_N	a_N	b_N	f_{lN}
A	$-2F_o$	$1+2\times F_o\ (1+B_i)$	$-2F_oBi$	T_N^{old}	Solid =0 ,liquid =1
B	0	1	0	Tm	$f_{lN}^{old}+2F_{ol}B_iT_0-2F_{ol}(1+B_i)T_m+2F_{ol}T_{N-1}-\frac{c}{L}(T_m-T_N^{old})$
C	0	1	0	Tm	$f_{li}^{old}+2F_{ol}B_iT_{01}-2F_{ol}(1+B_i)T_m+2F_{ol}T_{N-1}$
D	$-2F_o$	$1+2\times F_o\ (1+B_i)$	$-2F_oBi$	$b_i=T_m-\frac{L}{c}(f_{lN}^{*}-f_{lN}^{old})$	End of melting=1 End of solidification=0

2.6.3.5 Flowchart of melting or solidification of the PCM.

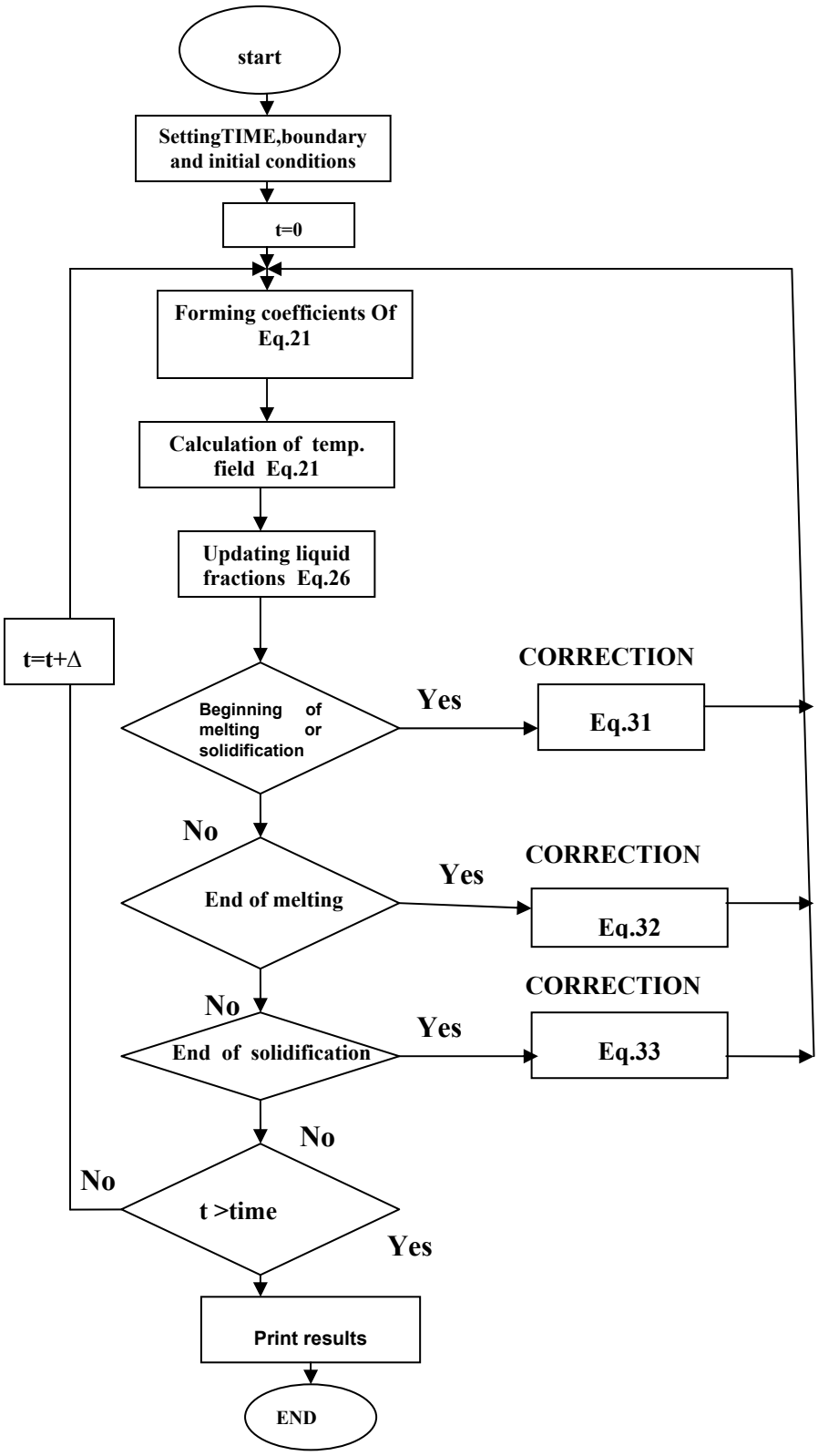


Fig.2.4 flow chart of numerical solution (enthalpy formulation) of PCM's one dimensional problem modified from [3]

2.6.3.5 Simulation of melting and solidification processes of rectangular container of PCM .

Consider one dimensional pcm problem; for a rectangular container filled of pcm as represented in fig2.3. The thermophysical properties of the pcm was taken from table2.7 which are constant for solid and liquid phases. The problem was solved with two different boundary conditions namely a constant end walls temperatures, and a constant temperature of fluid. The conducted walls of the container was taken to be made of stainless steel and the thermal resistance of the container was ignored.

Two computer programs were written in Java language to simulate the two different problems with the different boundary conditions(computer programs are given in appendix A).

Table.2.7 some PCM's thermophysical properties [16].

PCM	Tm(°C)	ρ(kg/m^3)	L(KJ/Kg)	c(kJ/kgK)	K [W/mK]
pcm32	32	1460	201	.7	.45
pcm30	30	1304	201	.69	.48
pcm28	28	789	245	2.22	.21
pcm21	21	1480	150	.68	.43
pcm19	19	1484	146	.68	.43
pcm17	17	1487	143	.67	.43
pcm13	13	1489	140	.67	.43
pcm10	10	1519	140	.66	.43
pcm8 b	8	1469	140	.67	.44
pcm7	7	1542	120	.62	.4
pcm4	4	766	227	2.18	.21
TH29	29	1500	190.0	1.76	1.0

The materials in table.2.7 were manufactured and tested by scientists, they are different in the composition .

The pcm problem was solved using different thermophysical properties of the solid liquid phases of the pcm. A comparison between some experimental data from [3], and the results of the numerical solution by taking the same conditions of this experiment was done. (appendix A.5 gives Java code for this case).

Finally a comparison is made between the results of two cases; the first solves the pcm melting process with different thermophysical properties for solid and liquid phases of the pcm A.5, and the second solves the melting process with the same thermophysical properties of the liqid and solid phases of the pcm A.3.

2.6.3.6.1 Constant temperature end walls of rectangular container of PCM:

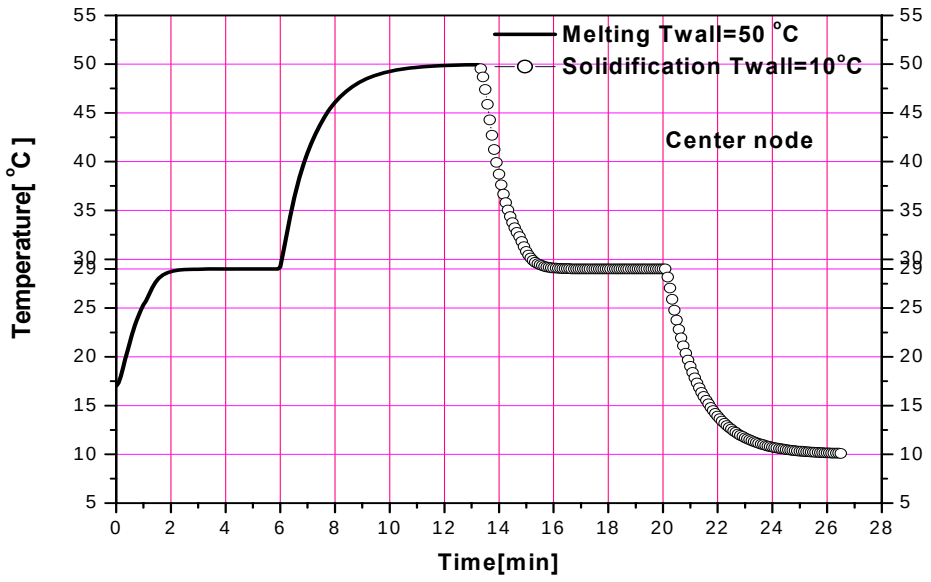


Fig.2.5 variation of time of PCM's temperature at the center with constant end walls , Δx=2mm, Δt=5s.

In fig.2.5 results are given for one dimensional problem rectangular slab of PCM TH29 with thermodynamical proprieties from Table.2.7 The PCM has a constant temperature at two ends (1,N) was set to 50 ^{o}C which is larger than T_m of the PCM the initial temperature was set to 17.0 ^{o}C at each node, after melting of the central node then suddenly the temperature of the two end walls was set to be 10 ^{o}C which is lower than the melting temperature of TH29 ,it starts to solidify and Fig.5 shows the variation of the temperature and time of melting and solidification of the PCM at the center. The central node was transfer the heat as sensible during (0-2.2) min , while during (2.2-6.0)min as latent; at this time the CV will store the heat during this phase change. Then the control volume (CV) melted completely, the heat returns to transfer as sensible. Suddenly the temperature at the two end walls was set to 10 ^{o}C then for the time (13-14.6)min heat is sensible. It begins to solidify at (14.6)min and ends at (20.2) min then sensible for (20.2 - 26.4)min.

2.6.3.6.2 Constant fluid temperature at end walls of rectangular container of PCM:

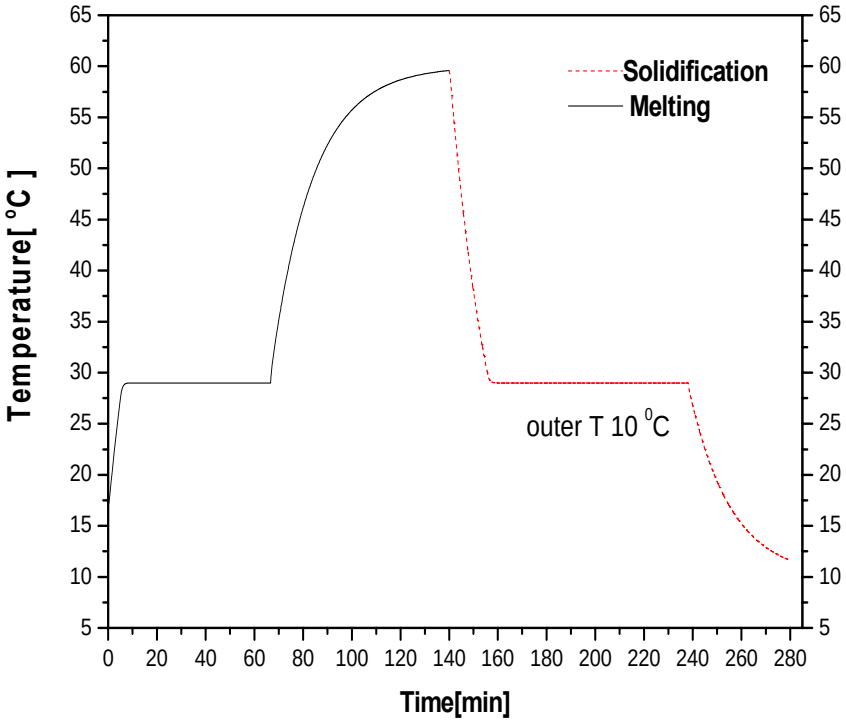


Fig.2.6 variation of temperature and time of PCM's center ,the ends were set to constant fluid temperature.

Fig.2.6 show that one dimensional convectional heat transfer of PCM at the end walls by hot air, here the PCM was TH29 $T_{initial}$ of it was 17.0 $^{\circ}C$ at t=0. The walls of the container of TH29 are of stainless steel, the resistance of the walls that conduct the heat to the PCM was ignored

The end walls are subjected to a certain fluid (hot air) at 60.0 $^{\circ}$C, the PCM thickness is 10mm and 9 nodes were taken Δx=2mm, Δt = 5s. T_{fluid} is set to 60.0 $^{\circ}$C for a time of

140(min).

Suddenly at the end of 140(min) , the end walls nodes (1,N) were subjected to fluid at 10.0 ̊C for a another time 140(min) using cold air.

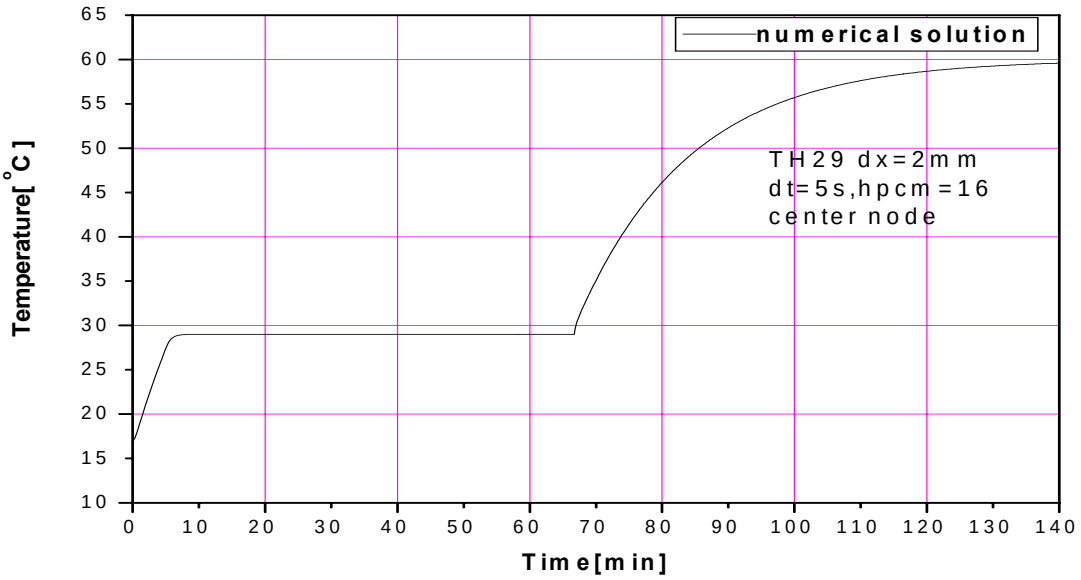


Fig.2.7. Temperature or the node at the center of the pcm

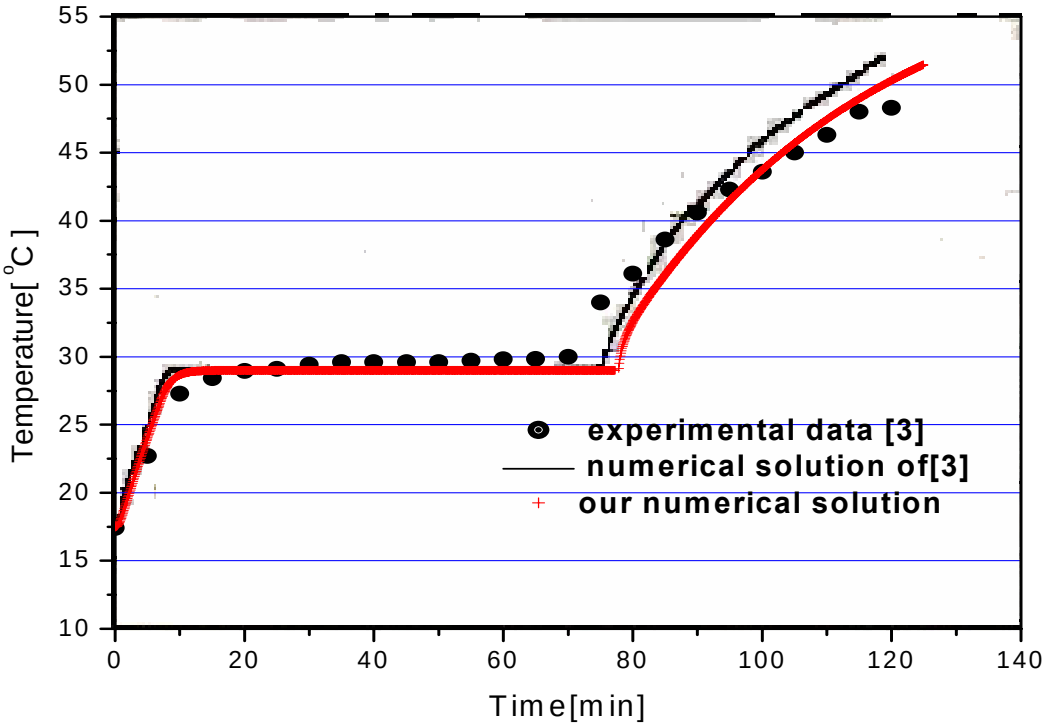


Fig.2.8 variation with time of pcm's temperature at the center of a rectangular container .

Fig 2.8 represents a comparison between, the results of experimental data, and our numerical solution. In the experimental model [3]; a rectangular container, made of stainless steel, with dimensions of *length* = 100mm, and, thickness = 20 mm. The container is filled with the calcium chloride hexahydrate (TH29) (appendix B.3 shows its thermal properties), and well insulated on the lateral sides.

" A thermocouple was placed in the centre of the container shown in Fig. 2.3 The container with the solid PCM was then placed vertically in the constant temperature bath, where the temperature was set to T = 60 °C."[3]

The computational model was set up to reproduce experimental conditions within the constant temperature bath [3].

"The convection heat transfer coefficient, between the air, and the container wall, was set to be h_{pcm} = 16 [$Wm^{-2}K^{-1}$].

Furthermore, a time step Δt = 5 s and space increment Δx =2mm were used in the calculation"[3]. In Fig.2.8 the variation with time of my numerical solution and experimental values of the temperature at the centre of the test container is shown.

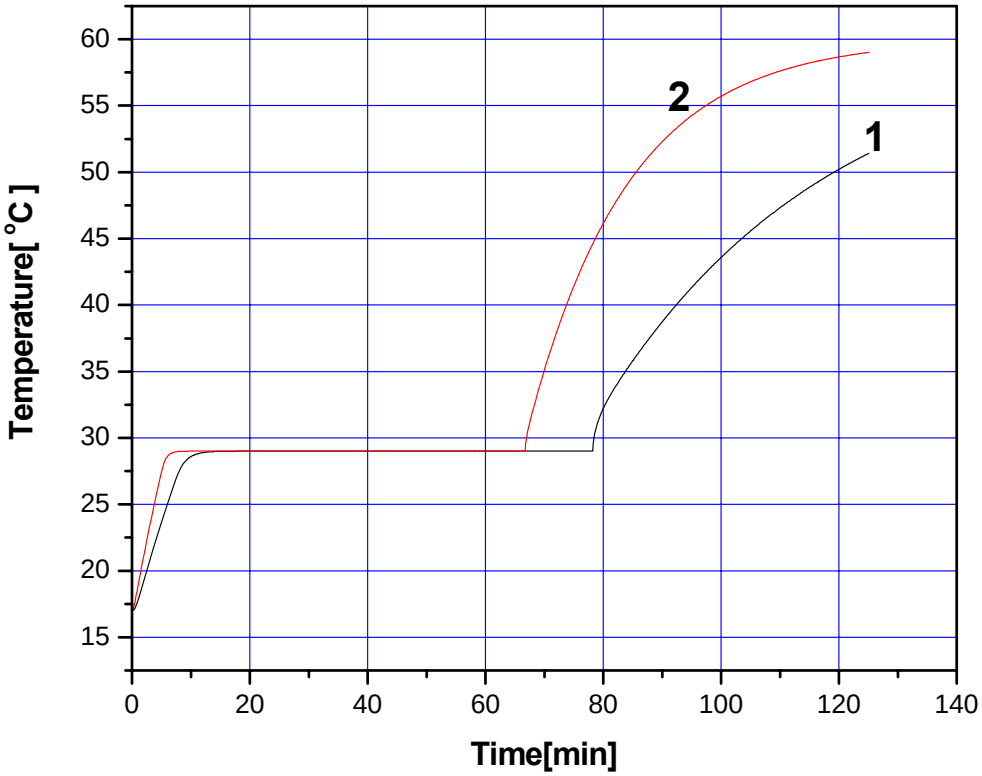
The numerical solution doesn't represent the experiment exactly this because of the mathematical model assumptions in our solution.

Consequently, the convective heat transfer coefficient between the air and the container wall used in the calculation is higher than the real one. Furthermore, it can be observed from Fig2. 8, that in numerical solution PCM reaches its melting temperature faster than in the experiment.

This is assumed to be due to the basic assumption made in the mathematical model that the conduction resistance of the container wall is neglected which is made of stainless steel and this assumption increases the heat entering at the boundaries .

Moreover, from the same figure it can be observed that the calculated PCM's melting time is slightly longer than the experimental one, which may be due to the fact that the natural convection within the liquid PCM is ignored. However, it can be seen that neglecting both the natural convection within the liquid PCM and the conduction within the PCM in the direction of the HTF flow do not introduce significant error in the prediction of the PCM's temperature variation during melting. [8].

The t-test, statistical analysis of fig.2.8 was doen for the experimental data and our numerical solution, for the time intervals (0-80) min, (80-120) min, and (0-120)min, for these three intervals the p-values were larger than 0.05 table.B.6; so they are not significantly different.



1 :different liqiud and solid thermal properties
2:the same liqiud and solid thermal properties

Fig.2.9. Variation with time of TH29's temperature at the center of a rectangular container .

Fig 2.9 represents two numerical solution of the problem solved in fig2.8, that in line 1 the difference between the thermal properties of the liquid and solid phases was taken into a account appendix B table.B.3, but in line 2 the thermal properties of the PCM TH29 was taken the same at the liquid and solid phases.

From the results shown in fig2.9; it can be concluded in line 2 the central node reaches to the melting point and ends the melting process faster than line 1 because of that the liquid thermal conductivity is 0.53 [W $m^{-1}K^{-1}$], and for solid 1.09

[W $m^{-1}K^{-1}$], the surrounding CVs of the center melts completely and their thermal conductivity will be set to that of the liquid phase, while in line 2 the thermal conductivity of the liquid and solid was taken to be 1.0[$Wm^{-1}K^{-1}$], the higher slope of line 2 after end of melting also due to the difference in the thermal conductivity, and the difference in the time of melting of the two models is small.

Chapter 3

The greenhouse

3.1 The aim of using greenhouses :

The greenhouse is a structure covering ground for growing a plant, that will return a profit to the owner, risking time and capital. It depends primarily on the solar radiation entering it through its cover. This cover is manufactured to allow direct and diffuse radiation with small wave lengths to transmit through it from outside, and prevent the transmission of these radiations from inside. Then they change into long waves due to the absorption of part of their energy by the greenhouse ingredients.

The greenhouses provide a protected environment for crops; they are used for overcoming climatic adversity, where preferable conditions for optimum plant growth are created inside them. Using greenhouses for plantation will decrease the bad influences of the changes in outside climate conditions especially in winter days. The Greenhouses are in contact with the outside environment. This leads to energy transfer between the inside and the outside of the greenhouses. Therefore, they have to be heated, cooled and ventilated depending on outside climatic conditions, so as to keep the inside conditions suitable

for plant growth and prevent the damage of the crop.

Since the primary aim of protected cultivation, is to increase profits by preserving product quality, and increasing crop yield, the climate control system must be as cheap and as reliable as possible. Heating a greenhouse can be performed using different systems and energy sources, such as fossil fuels and pcm systems. Each of these energy sources has its advantages and disadvantages in the greenhouse heating system. Heating systems that utilize fossil fuels might have low initial cost, but their running costs are high, as well as they cause pollution to the environment. On the other hand, pcm systems might have high initial costs but low running costs and are pollution free. This makes these systems highly attractive to farmers.

In principle four, physical phenomena are responsible for the differences between greenhouse and external climatic conditions:

1. Solar radiation, in particular the light whose wave length short carry large energy, it can penetrate the glass or plastic covering the greenhouse. They also penetrate, absorb and diffuse depending on the kind of the cover. So the cover is chosen depending on the outside climate conditions of the area.

On reaching the soil surface, plant canopy, heating installation, etc., the radiation changes to long-wave (infrared) which carry small energy, and can no longer pass through the covering, or it will pass with difficulty and change to thermal energy which raises the inside air temperature of the greenhouse.

2. The enclosed air within the greenhouse is nearly inactive because the inside air velocity is much smaller than it is outside. Accordingly, the effects of temperature transfer are entirely different.

3. The concentration of plant mass in the greenhouse space is much higher than outside. Artificial control of humidity and condensation clearly creates a different mass transfer from outside the greenhouse.

4. The presence of heating and other installations change some of the energy characteristics of greenhouse climate and affect the inside air temperature.

3.2 Static Design Procedure:

3.2.1 Heating load requirements

In order to select a heating system for a greenhouse, the peak-heating requirement for the structure must be determined at first. Each greenhouse has its own peak-heat requirement

that depends primarily on the greenhouse structure and the outside climate conditions.

Heat loss for a greenhouse is composed of two components:

(a) Heat will be lost through wall and roof by conduction and convection, and

(b) Infiltration and ventilation loss is caused by the entering of cold outside air through the walls, roof and the openings of the cover.

3.2.1.1 Transmission heat loss calculation

To evaluate transmission heat loss, the surface area of the structure of the greenhouse and the conductional and convectional heat-transfer coefficients between the outside air and the cover should be calculated. This surface area should be subdivided into the various materials employed, i.e. square meters of polyethylene, square meters of fiberglass, etc.. Each material has its own thermal properties. After that the transmission heat loss is calculated using the following equation:

$$Q_t = U \times A \times (T_i - T_{out}) \qquad (42)$$

where

Q_t **= Heat transmission losses through walls and roof [W].**

U **= Heat transfer coefficient [W m^{-2} °C^{-1}].**

A **= Surface area [m^2].**

T_i = Indoor design temperature [°C].

T_{out} = Outdoor design temperature [°C].

The value of the heat transfer coefficient (U), depends on wind speed. Hence, using table 3.1, a correlation between the U factor, and wind speed was found in order to use it in the program.

Table 3.1 (U) Heat transfer coefficient values (W m^{-2} $°C^{-1}$), at various wind speeds, V. [17]

Material	V = 0 (m/s)	V = 2.24 (m/s)	V = 4.47 (m/s)	V = 8.94 (m/s)	V = 11.18 (m/s)	V = 13.41 (m/s)
Glass	4.34	5.40	5.91	6.47	6.59	6.70
Fiberglass	3.95	4.91	5.39	5.87	6.01	6.12
Single poly	4.60	5.68	6.19	6.76	6.87	6.98
Double poly	3.04	3.58	3.83	4.07	4.13	4.18

$$U=-0.017\times V^2+0.39\times V+4.7 \quad (43)$$

Equation 43 was derived from Table 3.1 for single polyethylene using a simple Matlab program for fitting the data.

The glass is preferred usually for greenhouses due to its attractive proprieties; light transmission 85-90%, and lifespan indefinite if not broken.

3.2.1.2 Infiltration heat loss calculation

The air inside the greenhouse is replaced by cold air penetrating from outside via the natural air change that is called infiltration.

Infiltration heat loss is calculated by using the number of times of changing the inside air per hour (ACH). ACH primarily

depends on the kind of the cover and the openings made in that cover. Also it is a function of wind speed, greenhouse construction, and inside and outside temperatures. Table 3.2 outlines general values for different types of greenhouse constructions, which can be used by the designers.

Table.3.2 .Air change data for different glazing materials [21].

Greenhouse cover material	ACH
Single glass	2.5 - 3.5
Double glass	1.0 - 1.5
Fiberglass	2.0 - 3.0
Single polyethylene	0.5 - 1.0
Double polyethylene	0.0 - 1.0
Single polyethylene with low fiberglass sides	1.0 - 1.5
double polyethylene with low fiberglass sides	0.5 - 1.0
Single polyethylene with high fiberglass sides double	1.5 - 2.0
polyethylene with high fiberglass sides	1.0 - 1.5

$$Q_{inf} = \mathrm{V} \times ACH \times C_{air} \times \rho_{air} \times (T_{air} - T_{out}) \quad (44)$$

where

Q_{inf} **= Infiltration heat losses [W];**

V **= Volume of greenhouse [m^3];**

ACH **= Air change per hour (from Table 3. 2);**

C_{air} **= Specific heat capacity of air [J/kg°C];**

ρ_{air} **= Density of air [kg/m^3];**

T_{air} **= Indoor design temperature [°C];**

T_{out} **= Outdoor design temperature [°C].**

By addition of transmission and infiltration heat losses we get the total heat loss of the greenhouse as shown in Equation (45):

$$Q_{TOTAL} = Q_t + Q_{inf} \quad (45)$$

After calculating the maximum required heating load for the greenhouse by eq.40. The second step is to choose the appropriate system to heat the greenhouse. If unit heaters or fan coil units are chosen, only the devices with the required capacities are to be selected from the manufacturers' catalogues and installed in the greenhouse.

As for soil and bare tube heating systems, further calculations are needed in order to complete the design of these heating systems.

3.2.1.3 Solar Radiation Heat Gain

The key feature of the greenhouse phenomena is that glass and other greenhouse cover materials have the property of allowing short wave radiation coming from the sun to penetrate through them. Meanwhile, they prevent long wave radiation (infrared) which comes from the plants and soil from leaving the greenhouse, this infrared has peak at about 9.6 μm.

The solar radiation is the most important factor in greenhouse crop growing, this is because light is needed by the plants to produce organic compounds and oxygen by way of photosynthesis, light is also absorbed by the soil. The greenhouses cover trapping the thermal energy inside the greenhouse, which raises the air temperature inside the greenhouse. The respiration of the plants inside the

greenhouse, and their biological activities depends primarily on the inside air and soil temperatures.

"The fraction of the energy used in photosynthesis is negligible compared to the solar energy used in heating" [2], so it will be neglected in the calculations. Solar energy entering the greenhouse which is used to raise the inside temperature is calculated according to the following relation:

$$Q_{solar} = I \times \gamma \times \tau \times A_{cover} \quad (46)$$

where:

Q_{solar} = Useful solar energy entering the greenhouse [W].

I = Total outside solar energy falling on a surface (obtained from meteorological data), [W m^{-2}];

A_{cover} = Surface area of greenhouse cover , [m^2];

τ = Light transmission of greenhouse cover for solar radiation;

τ = 0.8 [18];

γ = Constant of the proportion of solar radiation entering the greenhouse, that is useful in increasing internal temperature; (γ is in the range 0.3-0.7).[18]

The greenhouse is a thermo mechanical system depends primarily on light, also light is the most significant parameter for the plant development and life. Most the active life process in the greenhouses can be achieved only in the presence and active influence of light.

Natural light (solar radiation) with specific wave lengths range has specific influence to the life processes of the plants. Solar radiation out of this range has energy related influence to the plants, directly or indirectly through the influence of the environment. This depends on the energy of the solar radiation which strikes the plant canopy, and depends directly on the wave length of the solar radiation.

It is found by the use of different scientific methodologies and investigations of changes in photosynthetical, phototropical, photomorphogenical and other plant activities, that only the part of total solar spectrum between 400 and 700 *nm* influences significantly plants life processes. This part of the solar spectrum must be maximally transparent by the greenhouse cover, and directly determines the quality of transparent cover materials.

3.2.1.3.1 Solar radiation transmission :

The greenhouse effect is the welcome result of radiation transmission into a closed space, although subsequent cooling of the air is required to maintain desirable plant growth temperatures some of them are in table B.5. The heat transfer capability of the covering system therefore becomes a very important secondary factor to consider when designing the greenhouse.

Each cover has its own transmittance, reflectance, and absorbance coefficients, these coefficients depend on the amount of solar energy which can be transmitted, reflected or absorbed by the greenhouse cover, some glazing characteristics are listed in table.B.4.

The transmitted portion is needed for plant growth, but only a fraction (1 - 5%) is actually utilized by the plant activities the remainder is absorbed and re-emitted as thermal radiation (heat), there by warming the greenhouse air.

The greenhouse cover transmits the direct radiation or beam radiation, and the diffuse radiation results from the scattering of direct radiation within the atmosphere (or by the cover) and can be received by the plant from all directions. The transmitted proportions of both direct and diffuse radiation, depends on the physical properties of the cover. The diffuse component will increase while the direct component is reduced. This is particularly true for many of the plastic films.

Transmittance (τ) is a physical property of the covering material. It is defined as the ratio of the measured radiation intensity beneath the covering material (I) to that measured above the cover simultaneously

Transmittance: $$\tau = \frac{I}{I_o} \qquad (47)$$

Transmittance is generally measured as the total radiation,

which is the sum of the direct and diffuse components. Transmittance values for a glazing are typically obtained from laboratory studies. The larger the value, the more desirable, but it must always be less than 100%. In most cases, this value will be representative of new, clean material, receiving beam radiation that is directed perpendicular at its surface. This procedure cannot determine the transmittance of similar cover material once installed within a greenhouse, as the circumstances are significantly different. Unfortunately, these are the values regularly provided for the comparison of prospective cover materials.

Diffuse radiation is much more predominant on the light-limiting, cloudy periods of the year. In order to maximize solar radiation for the plant at a specific time period (day, season, or year), covering materials with the best direct visible and diffuse radiation transmittance properties must be used. This could be calculated from the product of the hourly transmittance (τ) of direct and diffuse radiations, measured at the plant canopy, and the solar radiation above the cover (I_o) for each hour.

Total Hourly Energy = $I_o \times \tau$ **(48)**

[19].

3.3 Orientation of the greenhouse:

The orientation of the greenhouse is an important so that to posses maximum amount of the solar radiation at the day time. In Al-Aroub the orientation is that the long line of the greenhouse is aligned north south, and the wind direction must be taken into account. The prevailing wind direction is western in Hebron, so the direction is chosen to reduce and prevent damaging the greenhouse .

3.4 Mathematical modeling of the greenhouse:

3.4.1 Introduction

For a successful plantation inside the greenhouses; the inside temperature must be kept near the optimum level for plant growth, this level will depend on the crop being grown inside the greenhouse, and the stage of growth of this plant. In order to obtain the optimum level many climatic factors including relative humidity, air movement, and carbon dioxide must be controlled. Temperature and relative humidity are normally controlled by the accurate heating, and ventilation equipment. Continuous air circulation, especially in the winter time, is important for distribution of heat and uniformity of inside conditions by preventing air stagnation and stratification. Carbon dioxide enrichment of the air for greater plant growth and production is sometimes profitable; it is an active

participant of the chlorophyll assimilation. Generally, designs are for 18.3 ^{o}C inside capability with thermostatic adjustment for suitable conditions per horticultural recommendations.

Factors primarily affecting the heat requirements of a greenhouse are: [18]

1. The external environmental condition
2. The size of the greenhouse.
3. The structural nature of the greenhouse.
4. The number of layers of glazing material used to cover the house.

3.4.2 The heat balance equation of the greenhouse without PCM

To construct any control system, an appropriate building thermal model is needed. (Chao, K. and R.S Gates, 1999) reviewed some recent models for greenhouses and livestock houses, respectively.

The heat is gained by : [18]

1. conduction through the cover of the greenhouse
2. solar radiation transmission through the cover of thegreenhouse .
3. heaters inside the greenhouse .
4. convection through the cover

and the heat losses are by : [18]

1. radiation from the greenhouse to out of it.

2. conduction through the cover of the greenhouse

3. convection through the cover

4. ventilation and infiltration

Assumptions to simplify the greenhouse problem:

1. the air temperature inside is uniform .

2. the greenhouse roof is horizontal.

3. the effect of the humidity ,CO_2 concentration ,and latent heat of the water vapor was ignored.

4. the greenhouse assumed to be empty from crops.

5. the orientation is that long axis is aligned north- south.

The greenhouse dynamical equation:

$$\times c_{air} \times \rho \times \frac{d\,T_{air}}{dt} = \sum_{i=1}^{5}\left(A_i \times \tau \times \gamma \times S_i\right) - A \times U \times \left(T_{air} - T_{out}\right) - \rho_{air} \times c_{air} \times ACH \times \left(T_{air} - T_{out}\right) \quad \textbf{(49)}$$

≡ .

A ≡

Ai ≡ i .

τ ≡

U ≡

T_{out} ≡ .

T_{air} ≡ .

ACH ≡

S_i ≡

$i(1 \rightarrow 5)$ ≡

$\sum_{i=1}^{5}\left(A_i \times \tau \times \gamma \times S_i\right)$: **solar radiation that enters the greenhouse**

$A \times U \times \left(T_{air} - T_{out}\right)$: **convection and conduction heat losses or gain through the cover.**

$\rho_{air} \times c_{air} \times ACH \times \left(T_{air} - T_{out}\right)$: **infiltration and ventilation heat losses or gain.**

The implicitly formulation of the greenhouse dynamical equation:

$$\times c_{air} \times \rho_{air} \times \frac{\left(T_{air} - T_{air}^{old}\right)}{\Delta t} = \sum_{i=1}^{5} \left(A_i \times \tau \times \gamma \times S_i\right) - A \times U \times \left(T_{air} - T_{out}\right) - \rho_{air} \times c_{air} \times ACH \times \left(T_{air} - T_{out}\right) \quad \textbf{(50)}$$

$T_{air}^{old} \equiv$ *air temperature inside the greenhouse of the previous time step.*
$\Delta t \equiv$ *time step .*

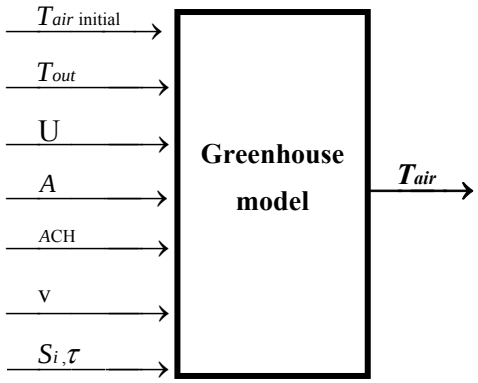
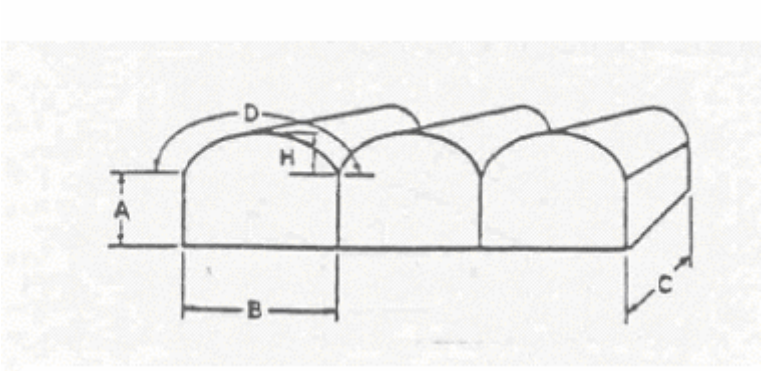


Fig.3.1 greenhouse model



A=3.0m
B=8.0m
C=50.0m
H=1.5m
D=10.0m

Fig.3.2 Gutter connected greenhouse

Fig.3.2 shows the dimensions and the shape of the greenhouse used in the simulation, it is a gutter connected type ,the orientation such that the long axis (C) directed toward south -

north ,the outside climate data; temperature, and solar radiation Al-Aroub 12 km north Hebron, was taken from [20] .

Table 3.3. Greenhouse input data used in the simulation.

Greenhouse data	
1. Greenhouse type	**Gutter connected**
2. greenhouse length (m)	**50.0 0**
3. greenhouse width (m)	**24.00**
4. side wall area (m^2)	**150.00**
5. Front wall area (m^2)	**96.25**
6. Back wall area (m^2)	**96.25**
7. roof cover area (m^2)	**1500.0**
8. greenhouse volume (m^3)	**4800.0**
9. greenhouse total surface area (m^2)	**1992.0**
10. greenhouse cover material	**Poly single**
11. greenhouse floor area (m^2)	**1200.0**
12. air density (kg/m^3)	**1.0**
13. air specific heat (c) (J/kg k)	**1044.0**

3.4.3. Simulation of greenhouse inside air temperature without PCM With various covers in 23th July at Hebron

A computer program in Java language was written to simulate the greenhouse mathematical model appendix A.1, the main input and output data are shown in Fig.3.1, at the night time there is no solar radiation S=0.0.

The solar radiation and outside temperature were taken from table.B.1 and table B.2 respectively, for Al-Aroub (north Hebron). The simulation was carried out for 23th July for 24 hours.

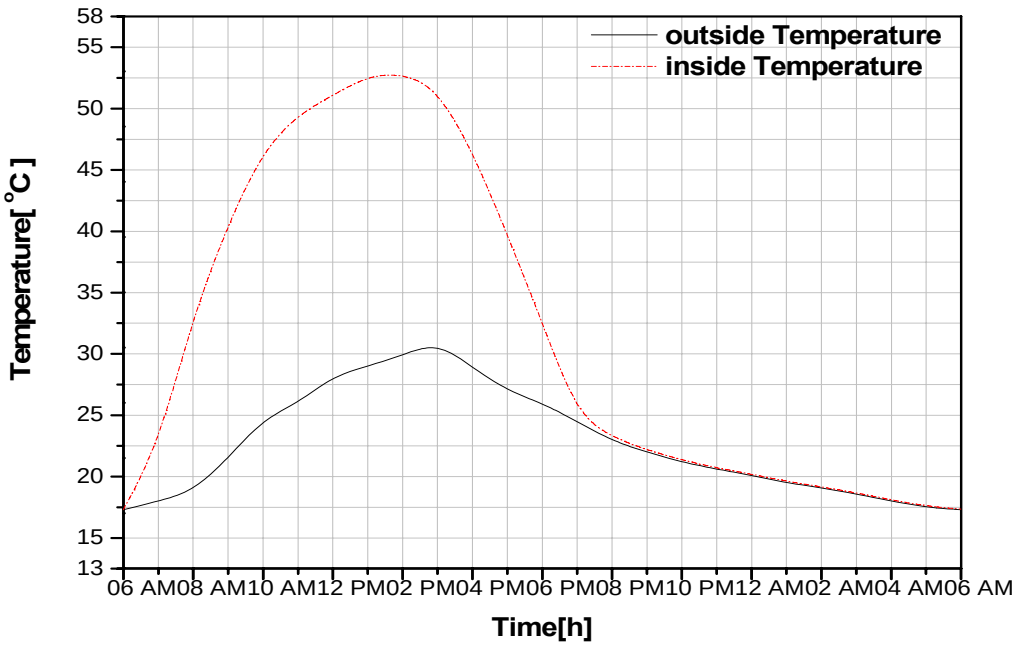


Fig.3.3 The outside air temperature and inside greenhouse air temperature single poly cover in 23 July at Al-Aroub .

Fig.3.3. shows the calculated air temperature inside the greenhouse and outside temperature in Al-Aroub in 23 July for 24 hours without pcm. The maximum difference in the inside and outside

temperature was 20 °C the initial temperature inside the greenhouse was taken 17.3°C, U=6.0 w/m^2 °C, v=5.0m/s, ACH[day=2.0, night=1.2].

The statistical analysis of fig 3.3, see table.b.6, shows that for the time interval (6am-6pm) the p-value is less than 0.05 so it is significantly different, the heat gains by the solar radiation is larger than the heat losses by the conduction and infiltration.

The p-value (7pm-6am) table.b.6 is larger than 0.05 so it is not significantly different, at this interval which is at night, so there is no heat gains, the heat inside the greenhouse will be lost, and the temperature inside the greenhouse will be the same as outside .

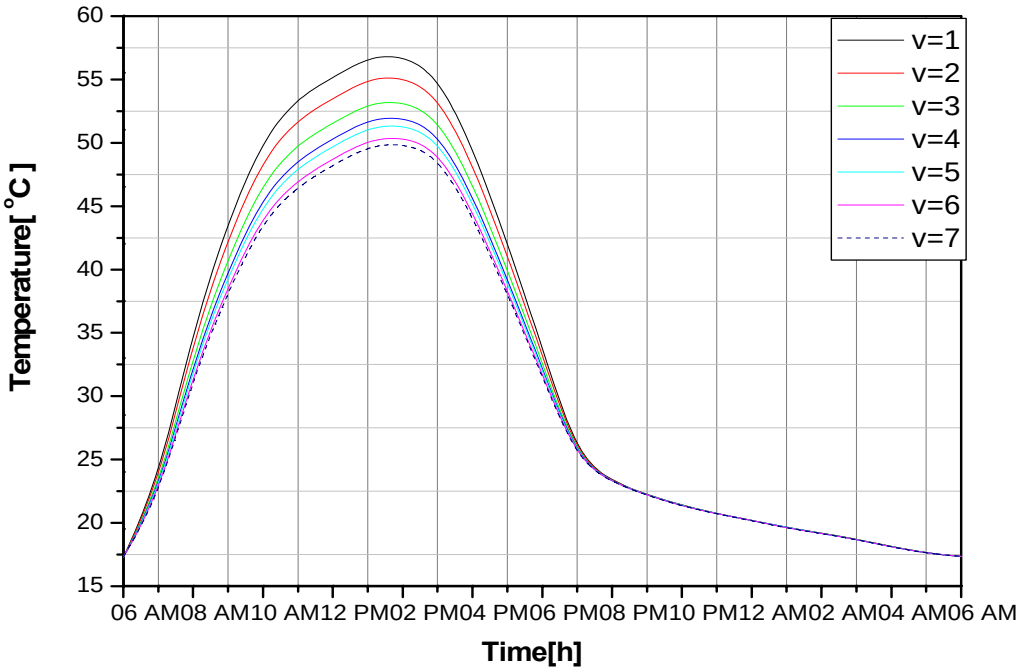


Fig 3.4 the temperature inside the greenhouse covered with single poly at 23 July with different outside air speeds [V] [m/s] .

Fig 3.4 shows the calculated inside air temperature for various outside air speeds, from this figure we can see that the maximum temperature is inversely proportional to the outside

air speed this is due to the increasing of the overall heat transfer coefficient (U) which increases with increasing the air velocity [18], hence increasing heat losses from the greenhouse, the initial temperature of the air inside the greenhouse was chosen 17.3 °C, τ =0.9, and the cover of the greenhouse is single polyethylene .

The statistical analysis of fig.3.4 was done by the t-test for (v=1m/s) and (v=6m/s), at three time intervals; first for (6am-10am) the p-value is larger than 0.05 statistically it is not significantly different, the solar heat gain at this interval is not large ,so the temperature inside the greenhouse at these two wind speeds nearly equal to the outside temperature.

The second time interval is (11am-4pm) the p-value table.b.6 is less than 0.05 so it is significantly different , the maximum solar heat gain occurs at this interval, the heat losses at (v=6 m/s) is larger than those at (v=1m/s) .

The third time interval is (6pm-6am), which is at night, no heat gains so the temperature inside the greenhouse at the two speeds nearly the same as the outside temperature.

Table3.4 U and ACH for various outside air velocities for single polyethylene [18]

V [m/s]	1	2	3	4	5	6	7
U [J/m^{2}k]	4.93	5.33	5.62	5.85	6.00	6.30	6.60
Day ACH	1.50	1.50	1.80	2.00	2.00	2.20	2.20
Night ACH	1.00	1.00	1.20	1.20	1.40	1.40	1.40

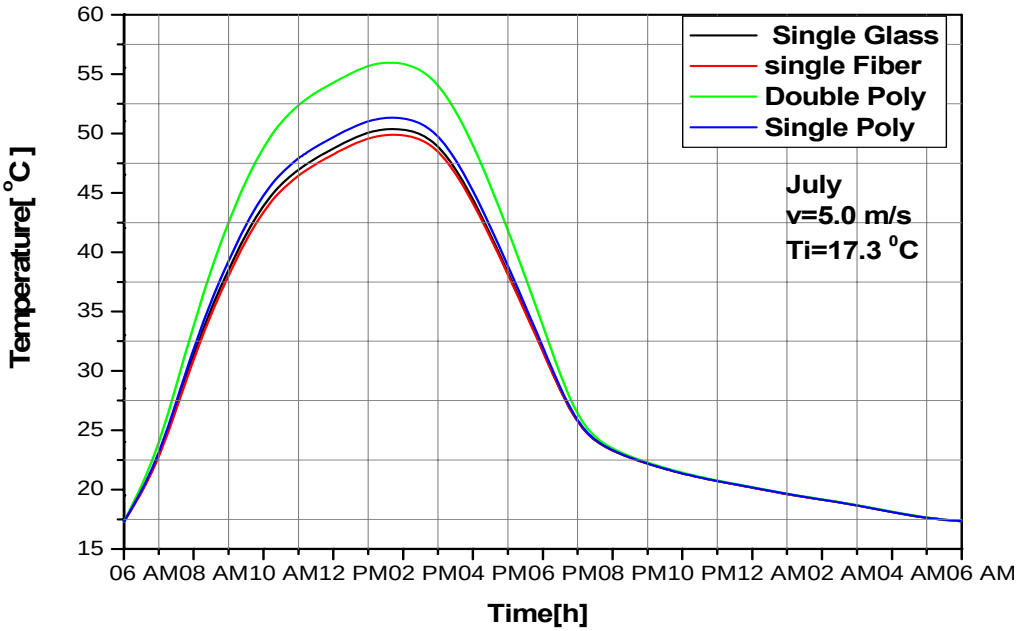


Fig.3.5 Variation with time of the greenhouse's inside air with various cover materials.

Fig.3.5 shows the inside air temperature of the greenhouse with various cover materials the figure tells us that double polyethylene cover traps the energy inside it more than single polyethylene, fiber, and glass because of high transmission of solar radiation and low infiltration heat losses.

The p-value of fig.3.5 for single poly and double poly curves at the time interval (12am-4pm), table.b.6 is less than 0.05, so it is significantly different, the heat losses of the greenhouse covered with double poly where U=3.9 Wm^{-2} k^{-1} is less than for single poly where U=6.06 $Wm^{-2}k^{-1}$, while p-values for the two time intervals (6am-11am), and (5pm-6am) of the same curves where larger than 0.05 so they are not significantly different, for the interval (6am-11am) the

solar heat gain is small and the transmittance of single poly is larger than the transmittance of double poly table 3.5, so the inside temperatures of the two different covered greenhouses is not significantly different .

For the time interval (5pm-6am), the solar radiation is minimum, and equal to zero at night time, so the inside greenhouses temperature will be the same as the outside temperature.

The t-test of the temperatures of the curves single glass and single poly at the time interval (11am-4pm); gives p-value larger than 0.05 so it is not significantly different, the transmittance of the two covers are the same, and also U is nearly the same, small difference only on the infiltration heat losses of the two covers which doesn't give significantly difference.

Table 3.5 input data for various greenhouse covers. V=5m/s.

Greenhouse cover	Input data				
	$\tau * \gamma$ (J/m^2k)	ACH(day)	ACH(night)	(τ)	U [w/m^2k]
Single polyethylene	.280	1.5	1.0	0.90	6.06
Single glass	.280	2.5	2.5	0.90	6.01
Single fiber	.255	2.5	2.5	0.85	5.46
Double polyethylene	.225	1.5	1.5	0.75	3.9

Chapter 4

Simulating the greenhouse with pcm

4.1The heat balance equation of the greenhouse with PCM :

$$\times c_{air}\times\rho_{air}\times\frac{d\,T_{air}}{dt}=\sum_{i=1}^{5}\left(A_i\times\tau\times\gamma\times S_i\right)-A\times U\times\left(T_{air}-T_{out}\right)-\rho_{air}\times c_{air}\times ACH\times\left(T_{air}-T_{out}\right)$$
$$A_{pcm}\times h_{pcm}\times(T_{air}-T_{wpcm}). \qquad \textbf{(51)}$$

A_{pcm} : *the surface area of the pcm.*
h_{pcm} : *the convectional heat transfer coeffecient of the pcm.*
Tw_{pcm} : *the temperature of the pcm surface.*

Eq.51 formulated implicitly as :

$$v\times c_{air}\times\rho_{air}\times\left(\frac{T_{air}-T_{air}^{old}}{\Delta t}\right)=\sum_{i=1}^{5}\left(A_i\times\tau\times\gamma\times S_i\right)-A\times U\times\left(T_{air}-T_{out}\right)-$$
$$\rho_{air}\times c_{air}\times ACH\times\left(T_{air}-T_{out}\right)-A_{pcm}\times h_{pcm}\times\left(T_{air}-Tw_{pcm}\right) \qquad \textbf{(52)}$$

rearanging Eq.52

$$T_{air}\times\left(1+\frac{(U\times A\times\Delta t)}{(v\times c_{air}\times\rho_{air})}+\Delta t\times ACH+\frac{\left(A_{pcm}\times h_{pcm}\times\Delta t\right)}{\left(v\times c_{air}\times\rho_{air}\right)}\right)-$$

$$Tw_{pcm}\times\frac{\left(A_{pcm}\times h_{pcm}\times\Delta t\right)}{\left(v\times c_{air}\times\rho_{air}\right)}=T_{air}^{old}+T_{out}\times\left(\frac{U\times A\times\Delta t}{v\times c_{air}\times\rho_{air}}+\Delta t\times ACH\right)$$

$$+\left(\frac{\Delta t\times\sum_{i=1}^{5}\left(A_i\times\gamma\times\tau\times S_i\right)}{v\times c_{air}\times\rho_{air}}\right). \qquad \textbf{(53)}$$

and from the boundary conditions Eqs.(40,41) at nodes 1 and N ,

and Eq.20 the fully implicit finite difference equation of node 1 is :

$$-2\times F_o\times B_i\times T_{air}+2\times F_o\left(1+B_i\right)\times T_{wall\ pcm}-2\times F_o\times T_2=T^{old}_{wall\ pcm} \qquad \textbf{(54)}$$

while the equation of node N is :

$$-2\times F_o\times B_i\times T_{N-1}+2\times F_o\left(1+B_i\right)\times T_N-2\times F_o\times T_{air}=T^{old}_N \qquad \textbf{(55)}$$

the other equations of the inner nodes of the PCM formed by Eq.20 .

The mathematical model of the greenhouse with the PCM was formed by Eq.52 and the enthalpy formulation of the PCM problem with the initial and boundary conditions Eqs.[38,39,40,41].

The system of equations was solved numerically with the Gauss-Seidel iterative procedure .

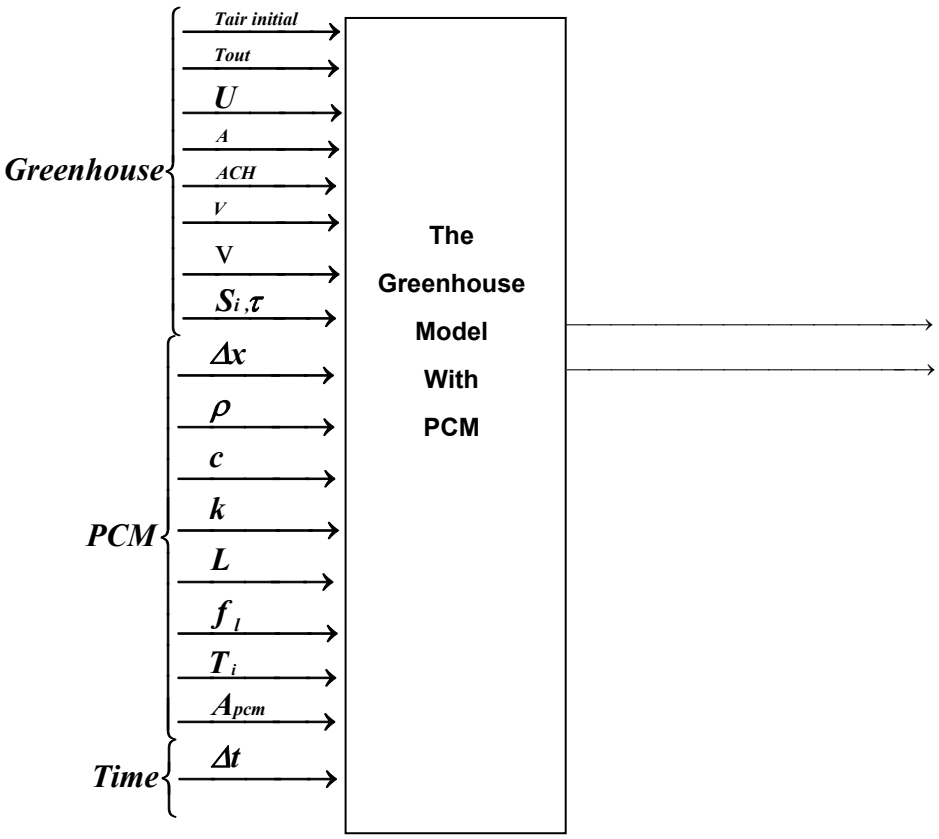


Fig.4.1 the greenhouse model with PCM.

Assumptions to simplify the greenhouse with PCM model:

1. **the problem is one dimensional heat problem.**
2. **the heat transfer from and to the PCM via two surfaces by convection only, and the other sides of the rectangular containers of the pcm are insulated .**
3. **the heat transferred inside the PCM is by conduction only.**
4. **the assumptions mentioned later in the greenhouse model without PCM.**
5. **the density, specific heat, latent heat , and thermal conductivity for the PCM are constants and the same for**

liquid and solid phases the initial temperature of the PCM assumed the same at each node.

6. the resistance of the container wall is ignored.

4.2 Results of the simulation of the greenhouse with PCM:

Pcm rectangular containers of stainless steel was distributed inside the greenhouse thus storage has h_{pcm} =5.0 $Wm^{-2}\,°C^{-1}$, total surface area is 200 m^2 for one side and because we have two symmetric surfaces of the pcm area must be multiplied by 2, the thickness of each container is 10 cm, Δx =0.02 m, Δt=300.0 s.

The simulation was done for three different days(23th July, 21th January, 23th February), the solar radiation and the outside temperature was taken at Al-Aroub, here the pcm thermal properties are taken the same for liquid and solid states .

A computer program in Java language appendix A.2, was written to solve this problem of greenhouse with pcm storage numerically (Eq.20, Eq.52).

4.2.1 the greenhouse with PCM in 23th July in Al-Aroub :

ACH, and U, at air velocity 5m/s are taken from table 3.4 for single polyethylene and from table 3.5 for other cover material.

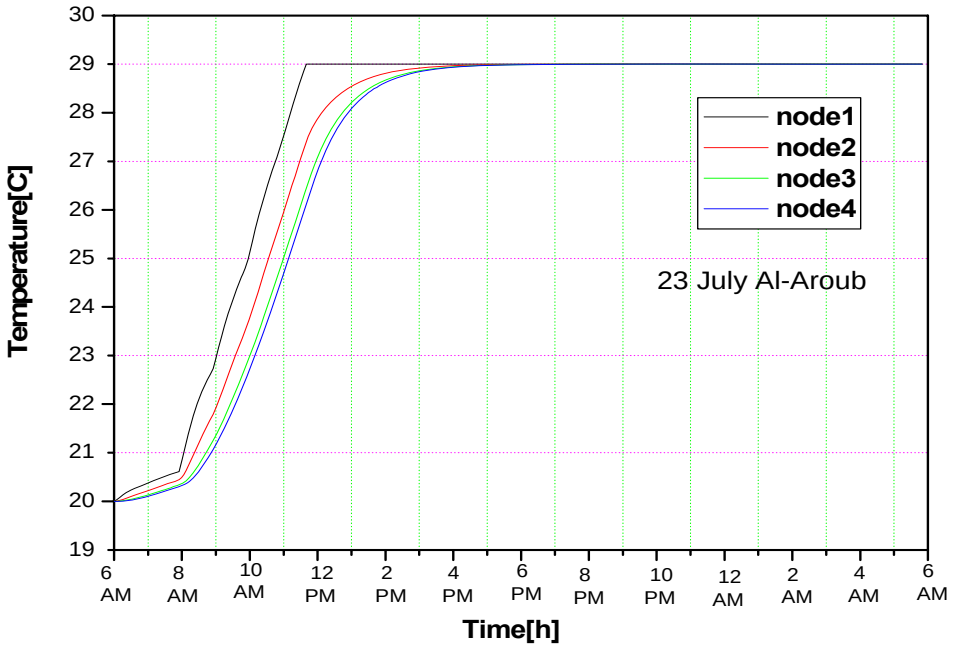


Fig.4.2The temperature of the nodes of the pcm inside the greenhouse through out day and night in 23 July in Al-Aroub .

Fig.4.2 shows the temperature nodes in pcm TH29 simulated inside the greenhouse during 24 hours

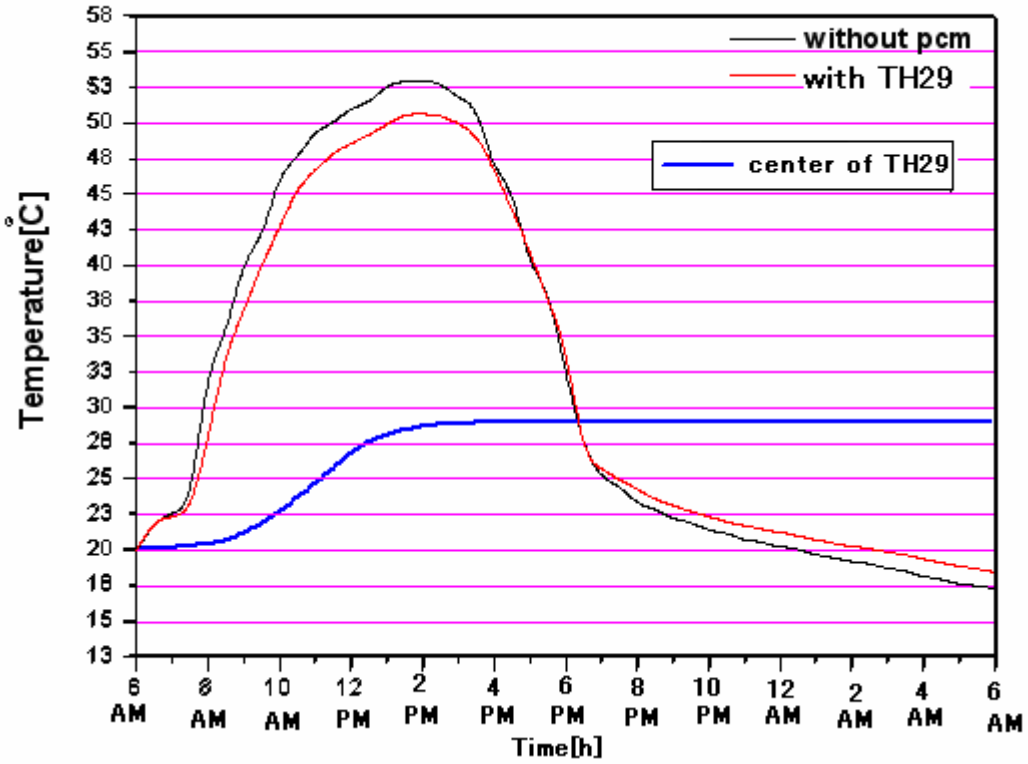


Fig.4.3 The air temperature inside the greenhouse covered with single poly with and without pcm.

The pcm worked as conditioner it takes heat from the air and store it at the day time and at night it discharge some of its stored energy to the air inside the greenhouse to raise the air temperature inside the greenhouse.

Using TH29 reduces the peak temperature of air inside the greenhouse by 3 ^{o}C and raises the minimum (lowest) temperature by 1 ^{o}C (at night). This means that the temperature range of the greenhouse is decreases by 4 ^{o}C during the whole day (24 hours).

The main problem of the pcm solidification and melting is it's low thermal conductivity and the low convectional heat

transfer coefficient from its surface to the air which is taken to be (5 W m^{-2} $^{\circ}C^{-1}$), both factors slow down the heat exchange between the pcm and the air, this increases the phase change time .

Hence the h_{pcm} must be increased to increase the discharging of heat and this will increase the inside greenhouse air temperature.

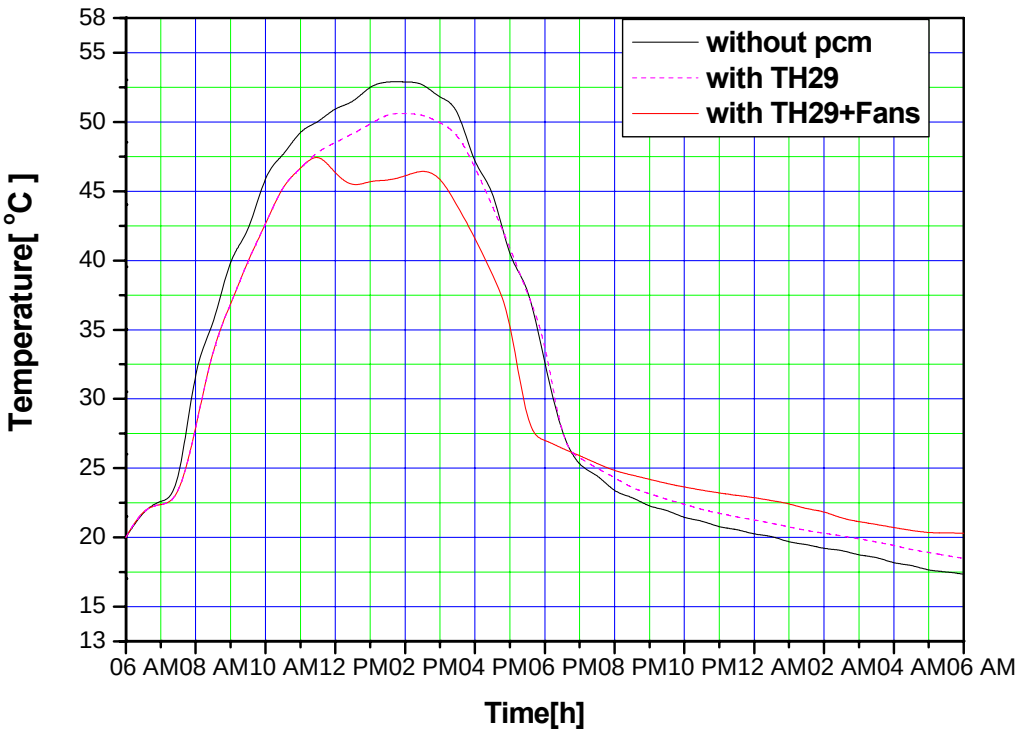


fig.4.4 the temperature inside the greenhouse single poly in July at Hebron with pcm TH29.

In order to get most of the stored or released energy and to increase the convectional heat transfer coefficient of the pcm (h_{pcm}) the fans must be used ,so from fig.4.4 the fans are used from time 12PM- 4PM O'clock at day to increase charging and from time 11PM-5.5AM O'clock at night time to

increase discharging of the heat, hpcm was taken to be (20 W $m^{-2}°C^{-1}$).

From this figure the temperature inside the greenhouse was decreases by $9^{o}C$ at day time and increases by $3^{o}C$ at night time.

Hence the temperature swing inside the greenhouse was 26 ^{o}C, in comparison with 35^{o}C without pcm.

The t-test of the two curves in fig.4.4; without pcm and with TH29 +fans was done for three time intervals :

1. (6am-11am) the p-value table.b.6 is larger than 0.05 so they are not significantly different, the mean of the air temperature inside the greenhouse with TH29+fans is 32.757 C at this interval is, and hpcm=5 ,the pcm stores small amounts of sensible heat which doesn't affect the temperatures of the two curve significantly.

2. (12am-4pm) the p-value table.b.6 is smaller than 0.05 so they are significantly different, the mean of the air temperature inside the greenhouse with TH29+fans is 45.26683 C at this interval, and the fans is on in this interval so hpcm is set to (20 W $m^{-2}°C^{-1}$), TH29 stores large amount of latent heat through the melting process .

3. (11pm-5.5am) the p-value table.b.6 is smaller than 0.05 so they are significantly different, the mean of the air temperature inside the greenhouse with TH29+fans is 21.59 °C at this

interval, and the fans is on in this interval so hpcm is set to (20 w $m^{-2}°C^{-1}$) , TH29 releases large amounts of latent heat through the solidification process .

Also the t-test of the two curves in fig.4.4; without pcm and with TH29 was done for two time intervals :

1. (12pm-4pm) the p-value table.b.6 is smaller than 0.05 so they are significantly different, the mean of the air temperature inside the greenhouse with TH29 is 49.47686 C at this interval, TH29 stores large amounts of latent heat through the solidification process .

But hpcm is set to (5 W $m^{-2}°C^{-1}$) , the p-value for this value is smaller than the p-value when hpcm is set to (20 W $m^{-2}°C^{-1}$) .

2. (12am-6am) the p-value table.b.6 is larger than 0.05 so they are not significantly different, the mean of the air temperature inside the greenhouse with TH29 is 19.85 °C, at this interval, and hpcm=5 w $m^{-2}°C^{-1}$, the pcm release small amounts of latent heat which doesn't affect the temperatures of the two curve significantly.

So in order to use the pcm efficiently you must increase the charging heat at the day time and increase the discharging at night.

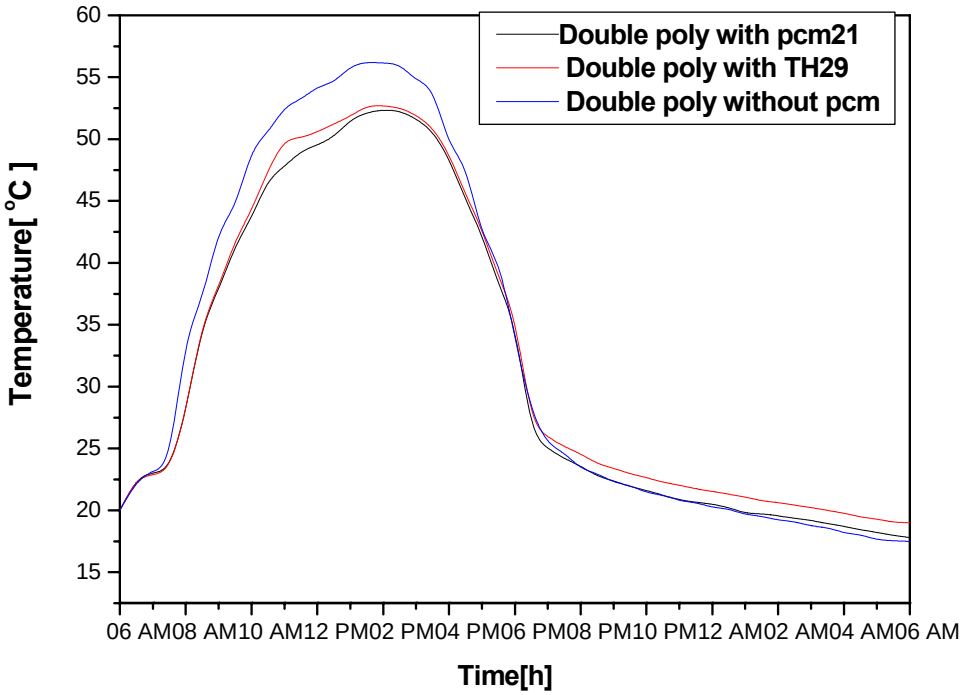


Fig.4.5 Variation with time of greenhouse's (-covered with single, and double poly-) inside air temperature with and without pcm.

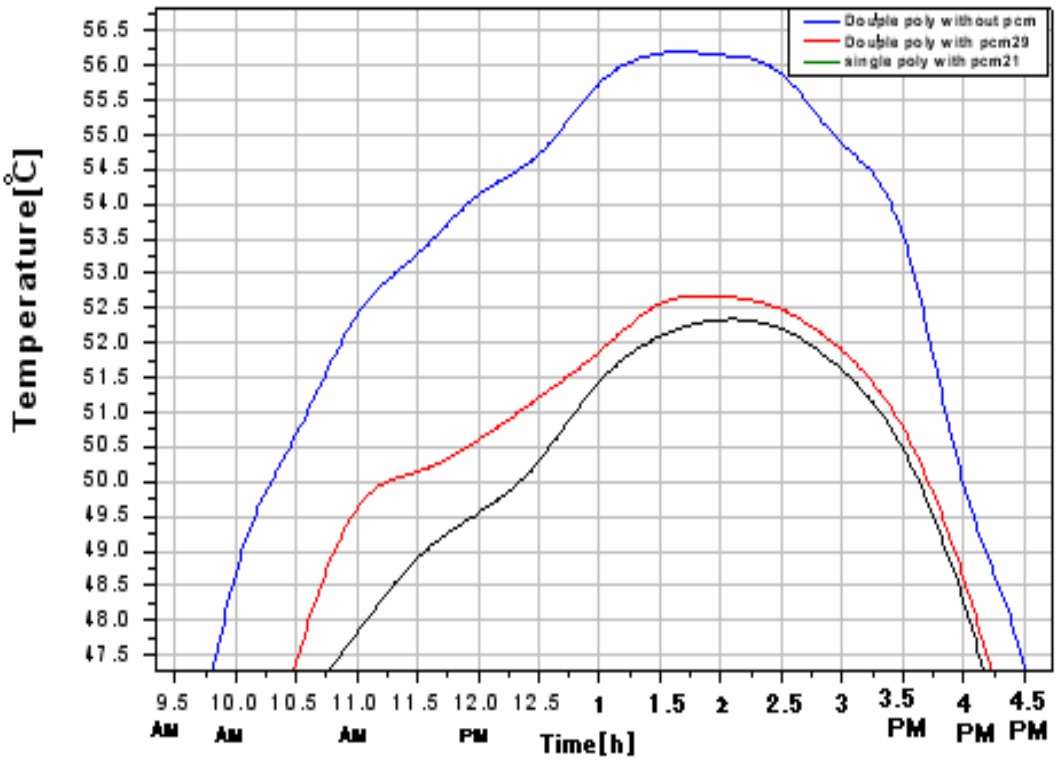


Fig4.6 Magnification of night time of fig 4.5

In Fig.4.5 and Fig.4.6 the cover of the greenhouse is doublepoly and, pcm21 and pcm29 were used in the simulation. Using pcm21 is better than pcm29 since it lowers the temperature of the air inside the greenhouse at day time. Pcm21 stores more a mount of heat at day time during it's phase change, it will melts faster than pcm29,now at the night time the temperature of the greenhouse is in the range of 20-25 °C.

The t-test of the two curves in fig.4.5; double poly without pcm and double poly with pcm21 was significantly different for the time interval (10am-4pm) with p-value 5.98367E-7, pcm21 melts completely and store large amounts of latent and sensible heat .

While it is not significantly different for the time intervals;(6am-12am) and (6pm-6am), this is due; the air temperature inside the greenhouse and the melting point , the small value of hpcm, and the small value of the conductivity pcm21 table.2.7.

Also the t-test of the two curves in fig.4.5; double poly without pcm and double poly with TH29 was significantly different for the time interval (10am-4pm) with p-value 0.00263, and (1am-6am) ,this is due the difference of the air temperature inside the greenhouse and the melting point of TH29 ,and its conductivity .

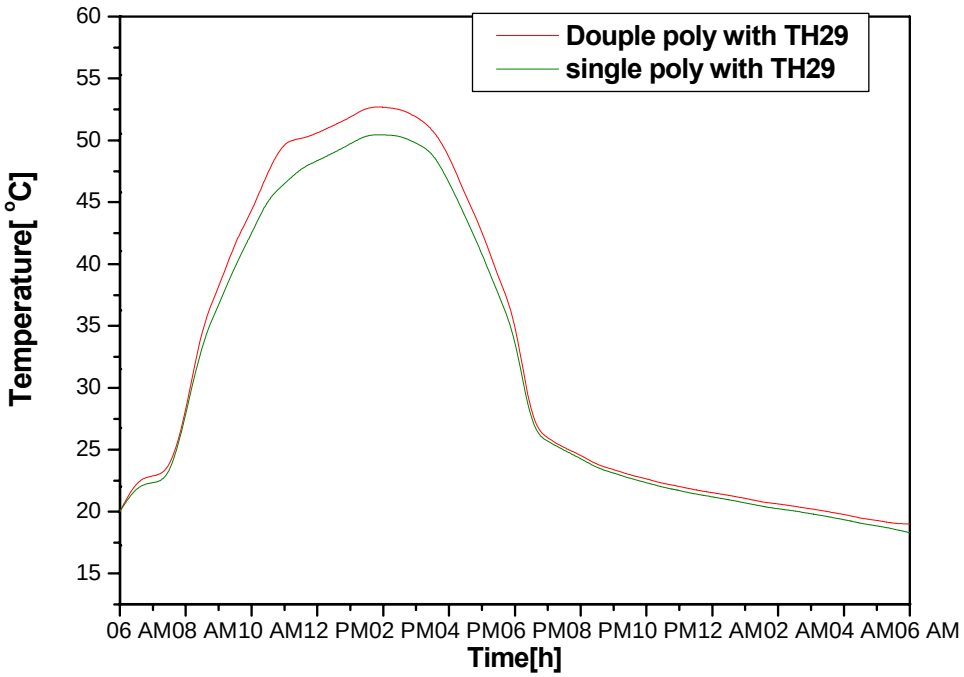


Fig .4.7 Variation with time of greenhouse's inside air temperature 23 July Hebron.

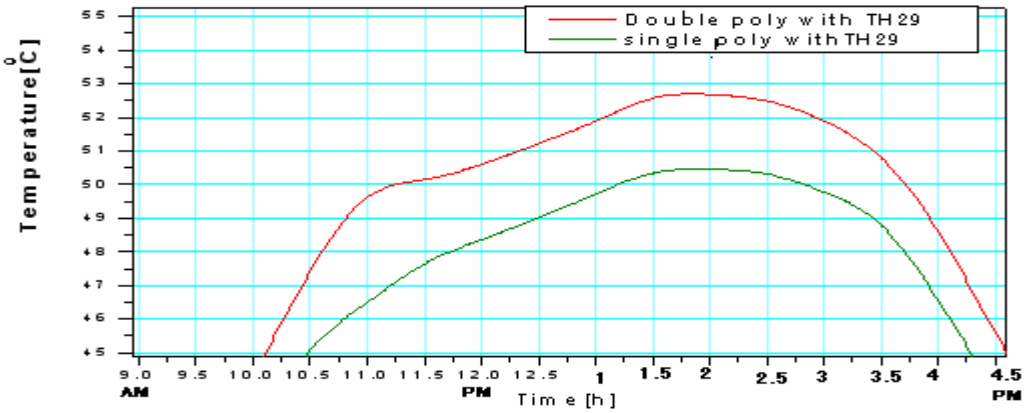


Fig 4.8 Magnification of part of fig 4.7.

The temperature of the air inside the greenhouse covered with doublepoly at the day time is higher than that covered with siglepoly

as shown in figs 4.7 and 4.8 this is because U of doublepoly is smaller that U of singlepoly, so the heat losses will be more for singlepoly .

The t-test of the two curves in fig.4.7; double poly with TH29 and single poly with TH29 was significantly different for the time interval (11am-4pm) with p-value 0.00124, the air temperature of the greenhouse covered with double poly is larger than the one covered with single poly, this is due to the difference in U table3.5, also the t-test is not significantly different when the solar heat gain is small at the interval (6am-11am), and at night time (6pm-6am).

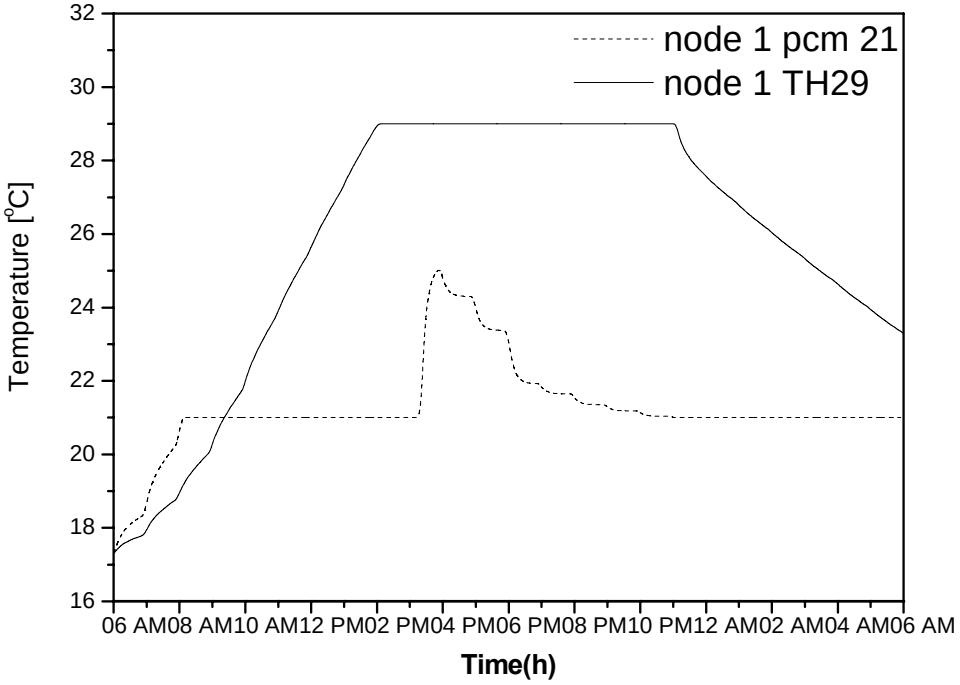


Fig.4.9 Variation with time of PCM's temperature at node 1 the rectangular containers inside the greenhouse covered with single poly at 23 July in Al-Aroub.

Pcm21 solidifification during the day time starts at time 3 PM O'clock, hence it is not useful. However TH29 solidifies at time 11 PM at night which is more useful, in summer days.

4.2.2 the greenhouse with PCM in 21th January in Al-Aroub :

Input data for January at outside velocity 7.0 m s^{-1} for single polyethylene greenhouse cover are taken from table.3.4 and with pcm21.

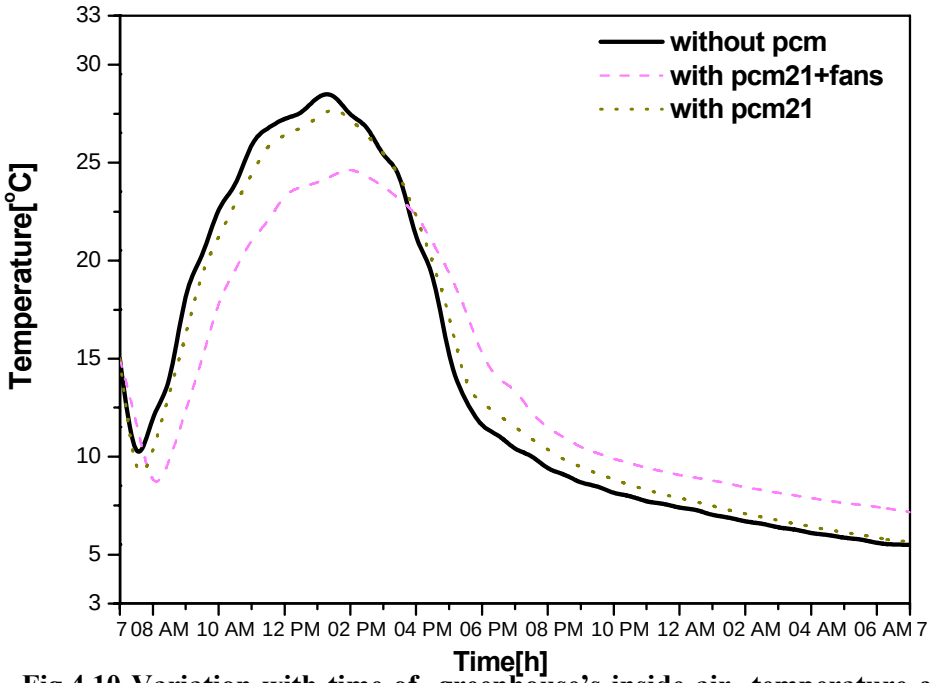


Fig.4.10 Variation with time of greenhouse's inside air temperature covered with single poly with PCM21 ,without PCM ,with PCM21+Fans at 21 January .

Fig.4.10 shows the air temperature inside the greenhouse versus time for three cases without pcm, with pcm21, and with pcm21 and fans.

Using the fans to increase the convectional heat transfer coefficient of the pcm (hpcm=20 $Wm^{-2}k^{-1}$) makes it more efficient.

The Fans will be used inside the greenhouse to increase the air speed of the air over the PCM surface containers this will increase the heat stored inside the PCM at day time .

The fan was on from (9,30AM-2,00PM) at day time and from (8PM-7AM) at night time.

So h_{pcm} is a primary variable that affects the discharging and charging operations of the PCM inside the greenhouse.

The t-test table.b.6 of the two curves in fig.4.10; without pcm and with pcm21+fans was significantly different for the time interval (11am-4pm) with p-value 7.31887E-4, this is caused by setting hpcm=20 $Wm^{-2}k^{-1}$ through (9,30AM-2,00PM), the pcm melts and store large amount of latent heat, also the t-test is significantly different for the time interval (8pm-7am) with p-value 9.78737E-6, when the temperature inside the greenhouse drops to low values int the range (7-5) C, then pcm21 release the stored latent heat, using the fans the fan through this interval speeds up the discharging process of the pcm .

The t-test of the two curves in fig.4.10; without pcm and with pcm21 was not significantly different for the time intervals, (11am-4pm) with p-value 0.5788, and (8pm-7am) with p-value 0.23869, for these two intervals hpcm=5, the mean of the temperature inside the greenhouse at the interval (11am-4pm) is 25.87864 C, this mean

doesn't allow the pcm to melt since hpcm =5, when the inside air temperature drops to low values at night time (8pm-7am), the released heat from pcm21 is not enough to affect the inside air temperature significantly.

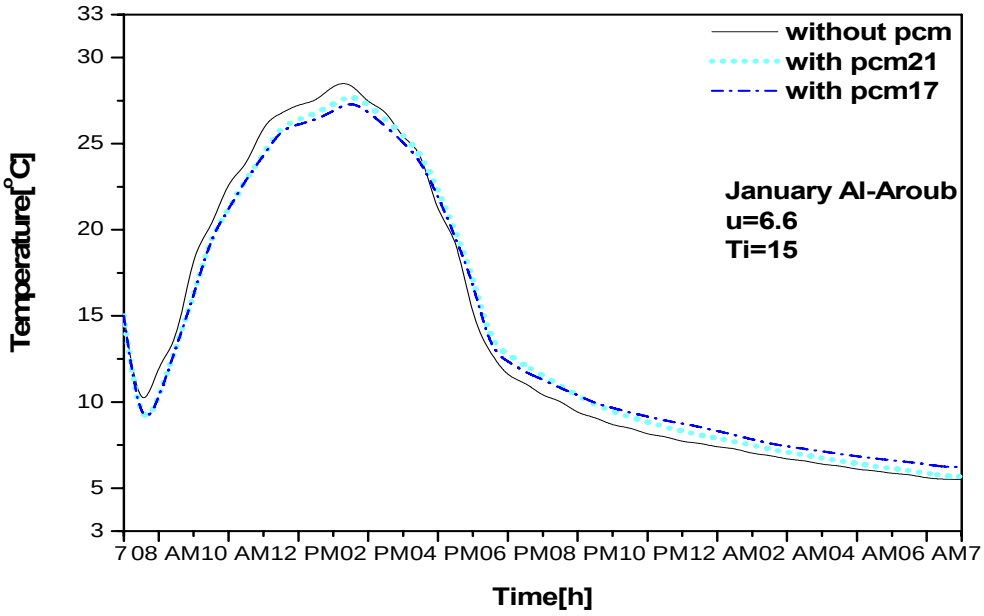


Fig.4.11 Variation with time of greenhouse's inside air temperature covered with single poly with PCM21 ,without PCM ,with PCM17s at 21 January .

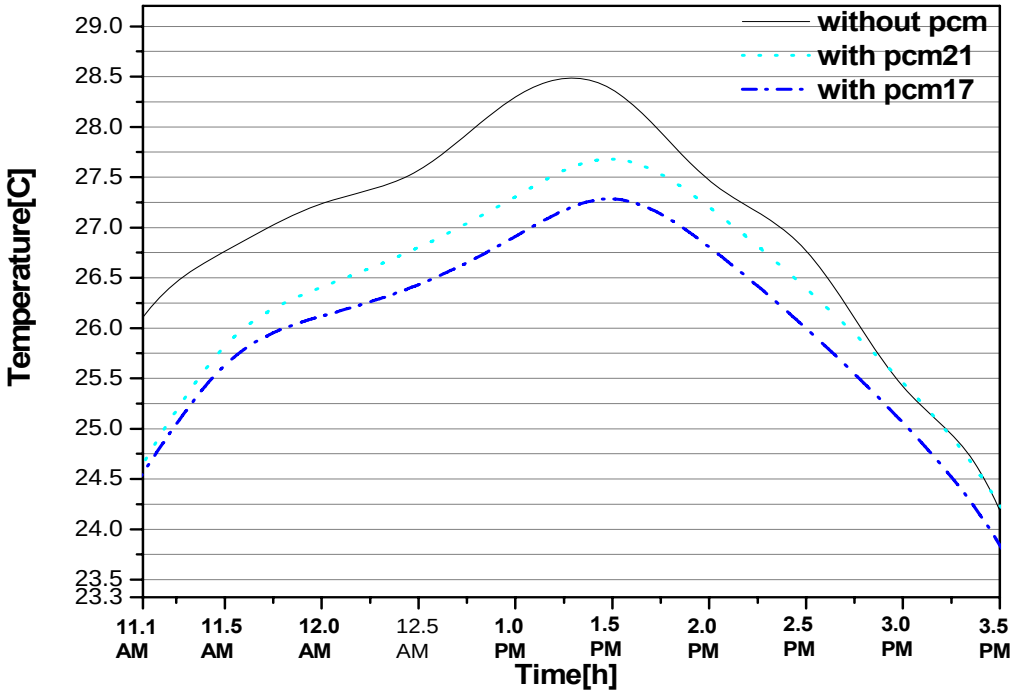


Fig.4.12 Magnification of Fig.4.11. form 11.1pm to 15.5 pm ,at day

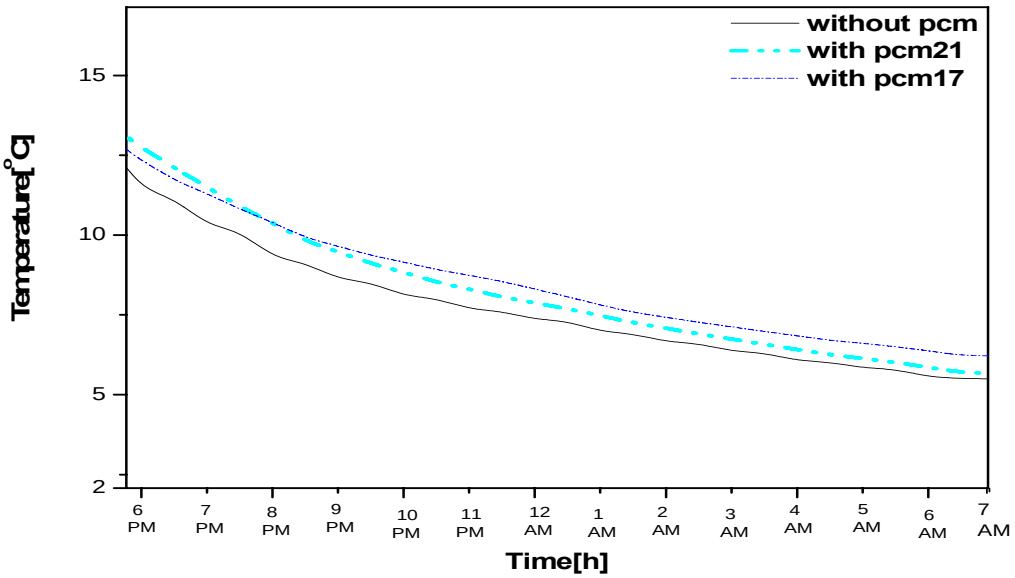


Fig. 4.13 Magnification of Fig.4.11 form 17.00 Pm to 7.00 Am ,at

The PCM initial temperature was set to 15 °C which is the same as the initial temperature of the greenhouse inside air .

The simulation was done with pcm17 and pcm21 at the day time

pcm17 will absorb more heat than pcm21 because of it low melting temperature so the pcm absorb large amount of heat during its phase change .

And at the night time when the temperature of the inside air of the greenhouse become lower than the PCM's surface temperature the operation of the discharging will start since PCM17 absorbed large amount of heat from as shown in Fig4.12, it will release more heat to the greenhouse than PCM21 as shown in fig 4.13 .

Using PCM17 gives better results than pcm21 as it reduce air temperature during the day and increase the inside temperature of the greenhouse by nearly 1.8 °C, while using PCM21 will increase it by 1.0 °C at night time.

From Fig.4.11 the t-test table.b.6 for the curves; without pcm and pcm21 is not significantly different, this is due the low mean of the temperature at the time interval (10am-5pm), pcm21 doesn't store large amount of heat, so it will release most of its stored heat at the at the few hours at night time, so the p-value for the time interval (8pm-7am) is not significantly different.

However the t-test table.b.6 for the curves of without pcm and pcm17 is significantly different at the time interval (8pm-7am), it stored larger amounts of heat than pcm21, this is due to the larger difference of the inside air temperature and the pcm melting temperature(17°C), this will affect the greenhouse inside air temperature significantly at the time interval (8pm-7am).

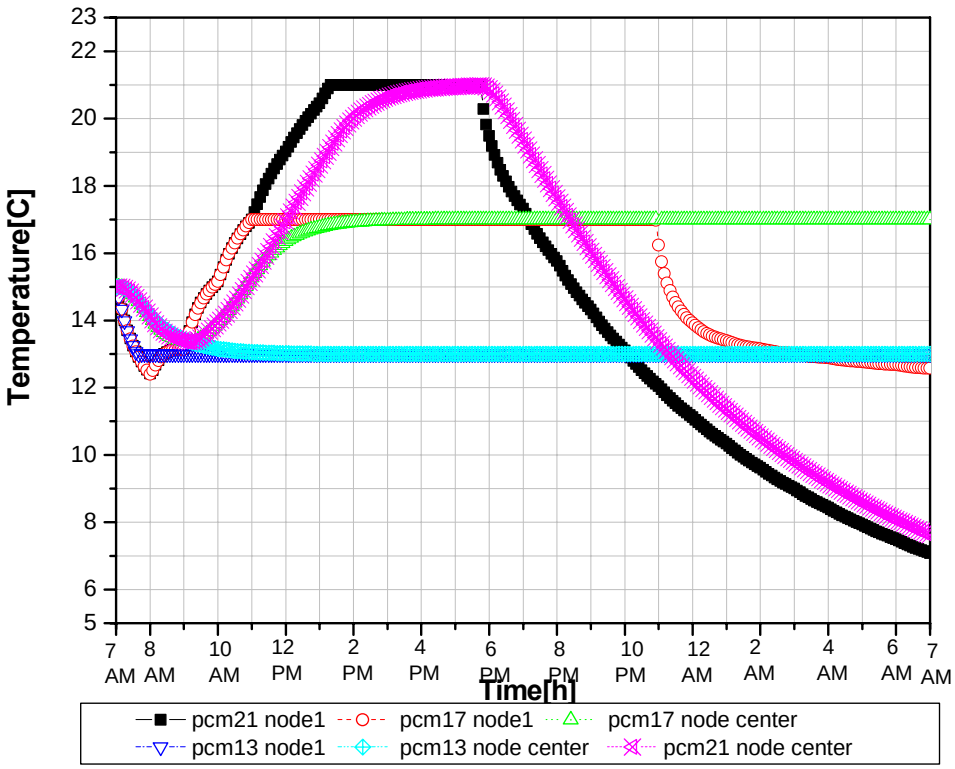


Fig.4.14.Variation with time of PCM's temperature at center of the rectangular containers inside the greenhouse covered with single poly at 23 January in Hebron

Fig4.14 shows comparison between the temperature of three different pcms at the surface node and the center node, for all the pcms node one starts of melting and starts of solidification faster than the node at the center, pcm13 starts of melting first and then pcm17 finally pcm21, but pcm21 ends solidification faster than pcm17 and pcm13, this is due the large difference of the temperatures of the air and the melting temperature (21-10) °C and smaller for pcm17 and pcm13, so the larger difference the faster in solidification and the slower in melting, so pcm13 is the best pcm to be used in winter days

4.2.3 the greenhouse with PCM in 23th February in Al-Aroub :

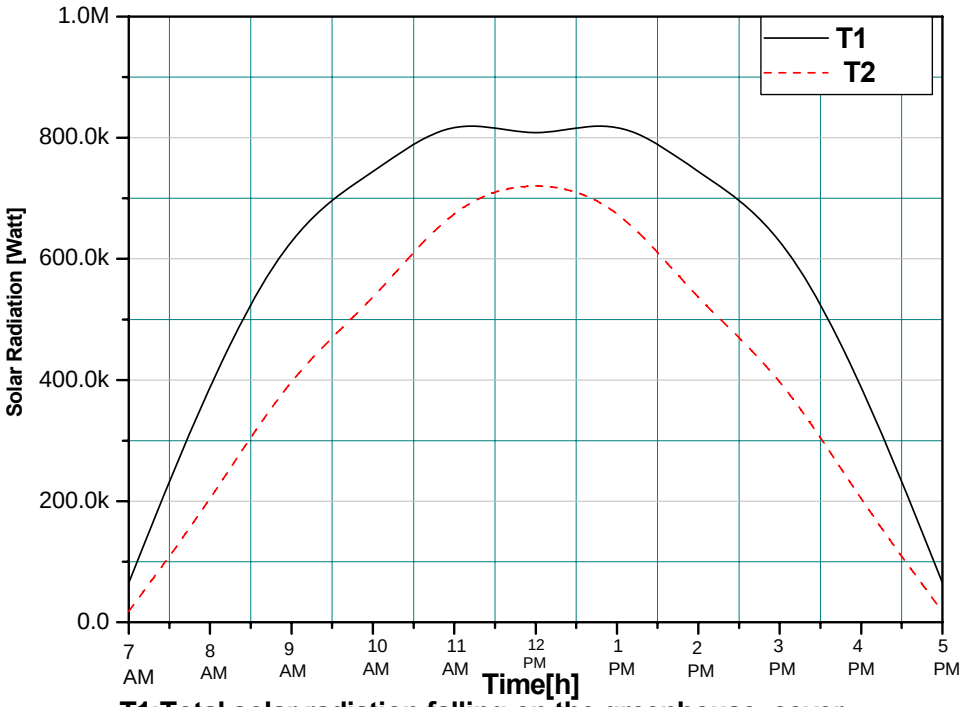


Fig.4.15 Total solar radiation falls at the sides of the greenhouse at day of 23 February

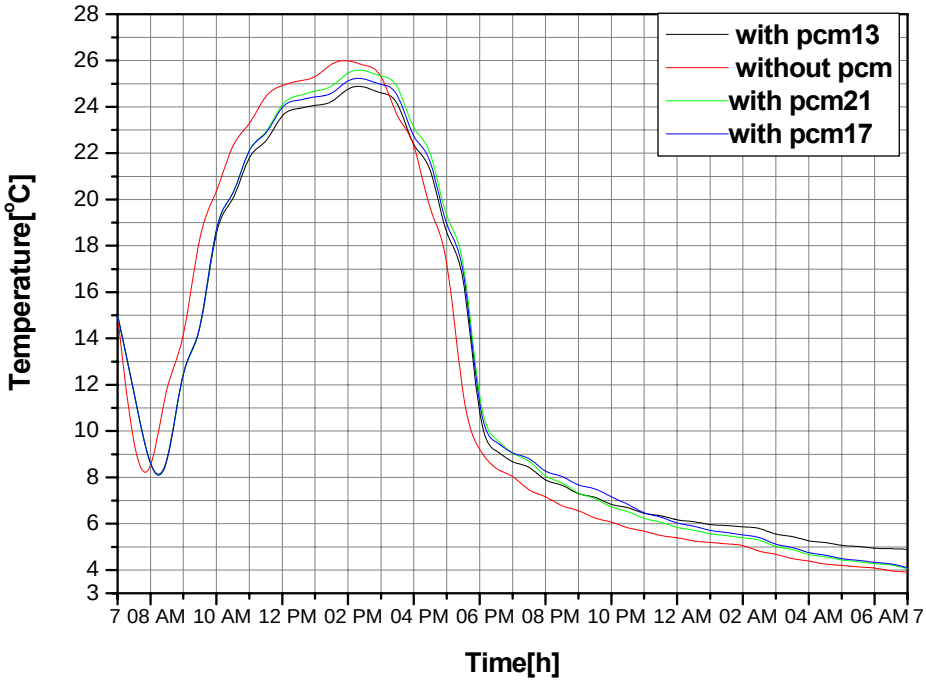


Fig4.16.Inside air temperature of the greenhouse with pcm13 ,21,17and without pcm in February 23 .

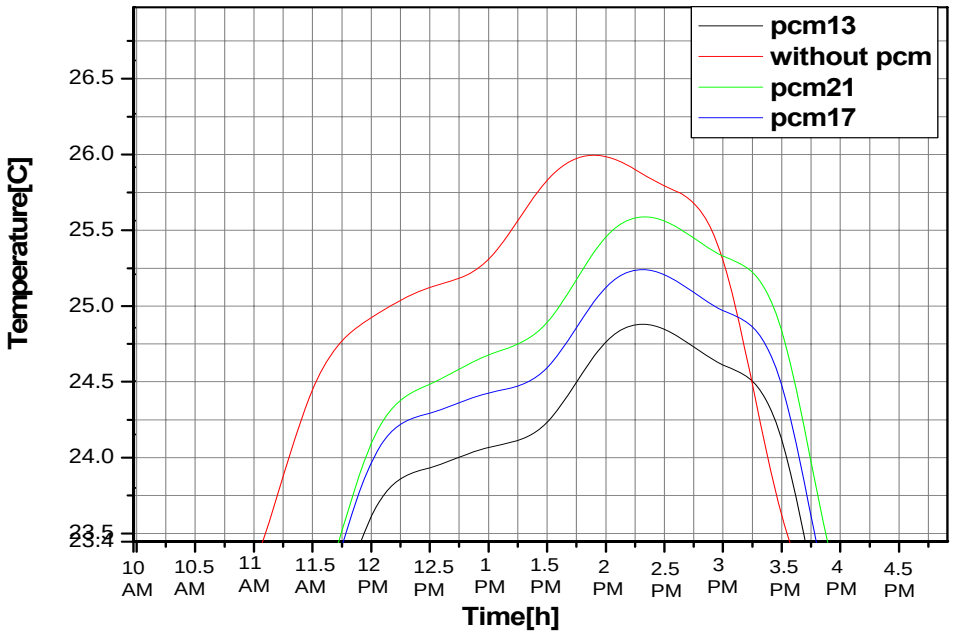


Fig4. 17. Magnification of Fig.4.16 of the greenhouse with pcm13 ,21,17and without pcm in 23 February

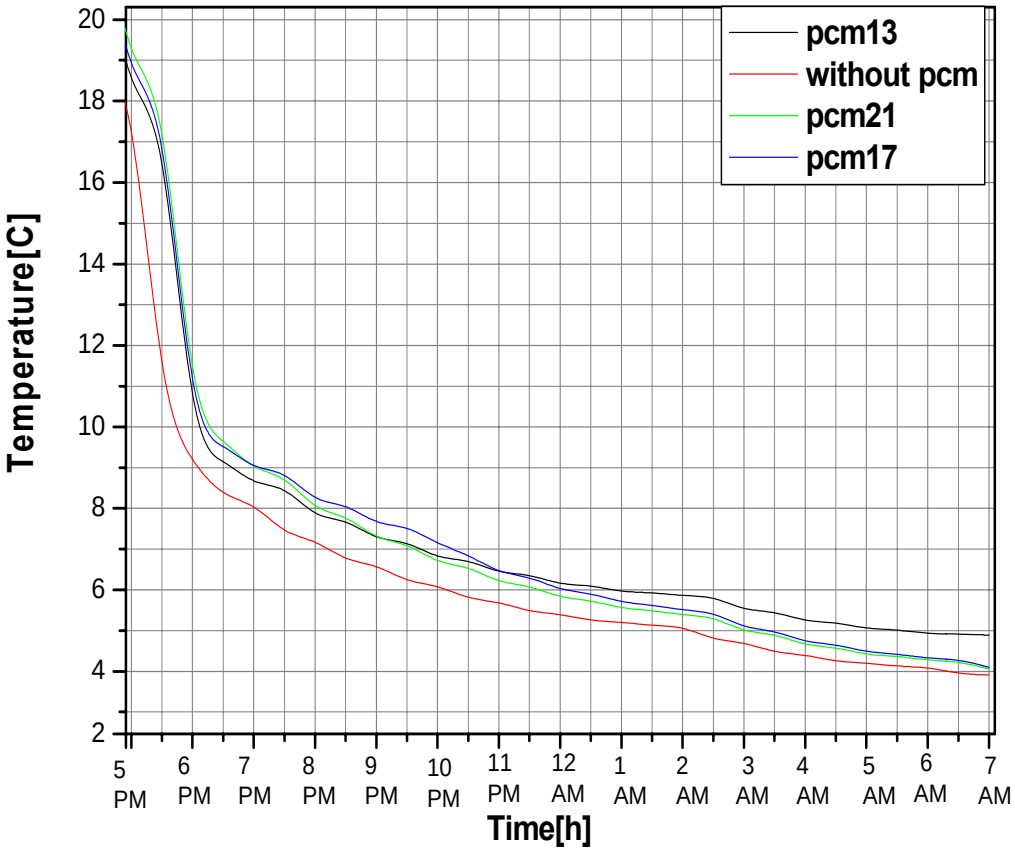


Fig4.18. magnification of Fig.4.16 of the greenhouse with pcm13 ,21,17and without pcm in 23 February 23 At night time.

From fig4.16 through fig.4.18 the simulation shows that using pcms 13 and 17 is more efficiently than using pcm21 this due to their low melting points which will allow it to make phase transitions and absorb the heat that will be released at night .

since the maximum temperature inside the greenhouse is 26.0 °C so we must use PCM's with low melting temperature to make them reach the melting temperature, so they can store large amounts of heat, that will be released later at night when the temperature of the greenhouse air drop down to low temperatures .

The t-tests of the curves in fig.416; without pcm and (pcm21, pcm17), are not significantly different ,this is due to the low day time temperature its maximum value is 26 C, pcm21 and pcm17 don't store large amounts of heat at the day time, so theire stored heat will not affect the inside air temperature at the night time when the inside air temprature drops in the range (4-7) C, but pcm13 will store larger amounts of latent heat this shown in fig.4.19 ,so the t-est for pcm13 is significantly different at the time interval (12am-7am).

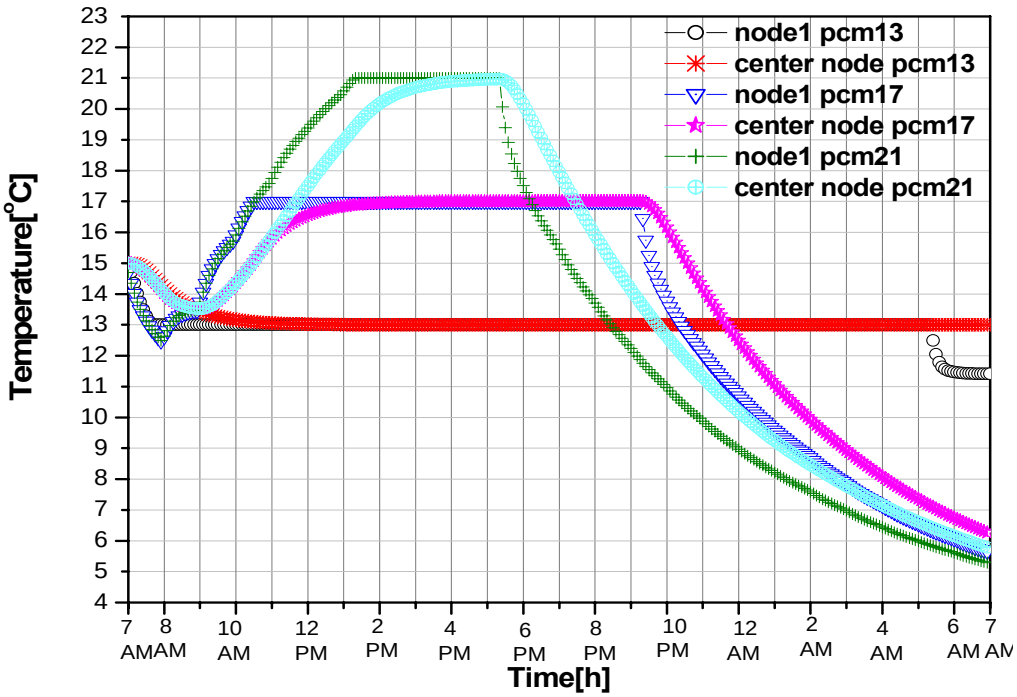


Fig4.19. variation with time of PCM's(17,13,21) temperature at the center and first node at February

Fig 4.19 shows that pcm13 the slower in releasing the stored energy so it is better to use in winter days than pcm21 and pcm17, pcm13 will release its stored energy when the temperature of the greenhouse drops to lower temperature nearly 6 °C or lower.

Chapter 5

Conclusions and recommendations

The main problem of using the pcm is its low thermal conductivity, and the low convectional heat transfer coefficient of air inside the greenhouse; this makes the rate of heat transfer from the air to the pcm storage and then through the pcm low.

The temperature of the greenhouse drop to low temperatures especially during the winter and this may destroy the plants inside it, so heating the greenhouse is needed to prevent such damage of the plants.

Using the phase change material inside the greenhouse in summer days lower the inside air temperature of the greenhouse at the day time by about (3- 4)^{o}C, so it will absorb large amounts of heat during the phase change process, and this stored heat will be released to the greenhouse when the air temperature drops at the night time .

Hence in sunny winter days the pcm raise the air temperature inside the greenhouse ,and reduce the swing in air temperature by about 5 ^{o}C .

The pcm chosen must be suitable, so it must undergo the phase transitions, and because the air temperature of the greenhouse is in the range (20-50) °C in summer days thus pcm29 or pcm21 can be

used .

In sunny winter days the day temperature inside the greenhouse is in the range (20-26) °C, so pcm13 or 17 is used in the simulations, in order to store the heat during their phase transitions.

The discharging and charging process of the pcm is slow because of their low thermal conductivities and convectional heat transfer coefficient between the air inside the greenhouse and pcm containers so, using fans raise convectional heat transfer coefficient between air and the pcm containers, so at day time larger amounts of heat may be stored inside the pcm, and using fans at night release this stored energy .

The cover material of the greenhouse is important, using singlepoly for the greenhouse at Al-Aroub gives good results of the inside temperature, but high results at noon time from the simulations using doublepoly will raise the temperature in winter days .

The outside air velocity and conditions directly affecting the inside conditions of the greenhouse.

For future work improve the simulation model of the greenhouse by incorporating crop inside, then all the conditions surrounding the crop must be taken into account (humidity, vapor pressure, latent heat of water vapor, temperature of soil ,…).

The model of the greenhouse with pcm may be improved by using metal fines inside the pcm containers to increasing its thermal

conductivity.

In addition using more accurate input data of outside temperature and solar radiation.

The orientation of the greenhouse is very important especially for winter planting, it depends on the location area that the solar radiation and direction of the wind, effect of the orientation can be investigated in future work.

Appendices

Appendix .A (program codes)

A.1 Solving dynamical equation of Greenhouse without PCM source code.

```
import java.io.*;
public class  DayNET
{
public static void main(String[]  args)
{
double dt=3600.0/2.0;  //time step
double hc=5.0;        // over all convectional    convectional
// heat transfer.
double Ti=20.0;        //initial Temperature inside the greenhouse
double vdot=1.0/3600.0; // ventilation rate
double Tout=20.0;       //initial outside temperature
double roha=1.5;        // air density
double ca=1044.0;       // air specific heat
double ap=.225;         //transmittance *solar coefficient
double NN=3.0;          // number of Guttered
double Ad=3.0;          // greenhouse height
double Bd=8.0;          // greenhouse width for 1 gutter.
double Cd=50.0;         // greenhouse length
double Hd=1.5;          // greenhouse height of the curve.
double Dd=10.0;         //greenhouse length of the curve for 1 gutter.
Af=Dd*Cd*NN;            // curved cover  area
double Ae=Ad*Cd*NN;     // East cover area
double Aw=Ae;           // west cover area= east cover area
double An=((2.0*Hd*Bd/3.0)+(Ad*Bd))*NN; // north  cover area
double As=An;           //south cover area= north cover area
double v=((2.0*Hd*Bd*Cd/3.0)+(Ad*Bd*Cd))*NN; //greenhouse volume.
Ai=Ae+Af+Aw+An+As;    // Total cover surface area.
double Z=3.8691;   // over all convectional heat transfer.
vdot=v*1.5/(3600.0); // ventilation rate
double ff=(1+Z*Ai*dt/(v*roha*ca)+dt*vdot/v);
//*******************************************************
PrintWriter pw=null;
try
{
pw=new PrintWriter(new FileOutputStream("d:/resaleh/greenw.dat"));
pw.println("A=[");
}
catch (IOException ex){}
pw.println("  " );
double TTTout[]=new double[13];
double TTout[]=new double[13];double H[]=new double[13];
double E[]=new double[13];double WW[]=new double[13];
double SS[]=new double[13];double N[]=new double[13];
// reading Tout ,Solar radiation each hour from a file//
TTout=find(1.0 ,"july.dat" ,13);   //Tout
H=find(2.0 ,"july.dat" ,13);   //solar radiation at horizontal surface
E=find(3.0 ,"july.dat" ,13);   //solar radiation at eastern surface
WW=find(4.0 ,"july.dat" ,13);  //solar radiation at western surface
N=find(5.0 ,"july.dat" ,13);   //solar radiation at nothern surface
SS=find(6.0 ,"july.dat" ,13);  //solar radiation at southern surface
```

```
for(int p=1;p<25;p++)
{
if(p==1){
Tout=TTout[0];
S=(H[0]*Af+E[0]*Ae+WW[0]*Aw+N[0]*An+SS[0]*As); // total solar radiation
// falling the greenhouse cover
// each hour
}
if ((p%2)==0.0)
{S=(H[p/2]*Af+E[p/2]*Ae+WW[p/2]*Aw+N[p/2]*An+SS[p/2]*As);
Tout=TTout[p/2];}
double W=Ti+(Tout*(Z*Ai*dt/(v*roha*ca)+dt*vdot/v)+(dt*ap*S)/(v*roha*ca));
double matr=W/ff;  // solution
pw.println("   "+matr);
Ti=matr;
}  //end  main for loop
// greenhouse night
vdot=v*1.5/(3600.0);
ff=(1+Z*Ai*dt/(v*roha*ca)+dt*vdot/v);
S=0.0;   // night solar radiation
for(int p=1;p<25;p++)
{
TTTout=find(7 ,"july.dat" ,13);  // reading night temperature
if ((p%2)==0.0)
{Tout=TTTout[p/2];}
if (p==1){Tout=TTTout[0];}
double W=Ti+(Tout*(Z*Ai*dt/(v*roha*ca)+dt*vdot/v)+dt*Ai*ap*0.0/(v*roha*ca));
double matr=W/ff;
pw.println("   "+matr);
Ti=matr;
}  //end  main for loop
if (pw!=null) pw.close();
pw.println("];");
pw.print("plot(A)");
if (pw!=null) pw.close();
} //end main**********************
//file reader  **************************
public static double[] find(double g ,String  fil,int x )
{
FileReader frs =null;
StreamTokenizer in=null;
String sname="slam";
double [] r=new double[x];
double mid=0.0;
try{
frs=new FileReader(fil);
in=new StreamTokenizer(frs);
while(in.ttype!=StreamTokenizer.TT_EOF)
{
if (in.ttype==StreamTokenizer.TT_WORD)
sname=in.sval;
else
while (in.nextToken()==StreamTokenizer.TT_NUMBER)
{
mid=in.nval;
if (mid==g){
for (int i=0;i<x;i++)
```

```
{
if (in.nextToken()==StreamTokenizer.TT_NUMBER)
mid=in.nval;
r[i]=mid;
}
}
}
in.nextToken();}
}
catch(FileNotFoundException ex) {
System.out.println("receved" );
}
catch(IOException ex)
{
System.out.println(ex.getMessage());
}
finally
{
try
{
if (frs!=null) frs.close();
}
catch(IOException ex)
{
System.out.println(ex);
}
}
return r;
}
};//end main class  (the end of  the program)
```

A.2 the greenhouse with phase change material (PCM) source code.

```
import java.io.*;
public class  WinterFinal
{
private static   int t;
private static   int Q;
private  static   int l;
private static   double[] matrix=new double[Q];
private static   int q;
public static void main(String[]  args)
{
PrintWriter pw=null;
try
{
pw=new PrintWriter(new FileOutputStream("d:/resaleh/finalWinter.dat"));
```

```
}
catch (IOException ex){}
pw.println("  " );
double TTT=0.0;;
int OK;
Q=9;
double[] Pm=new double[5];
Pm=find(3.0 ,"d:/resaleh/pcm.dat" ,5);
for (int tm=0;tm<5;tm++){
pw.print("%  " +(double) (Pm[tm]));}
double Tm=Pm[0]; // melting Temperature of the PCM [c].
double dt=300.0; // time step [s]
double  dx=0.02; // control volume width.
double d=Pm[4];  // conductivity of PCM.
double c=1000.0*Pm[3]; // spesific heat of PCM.
double roh=Pm[1];      // density of PCM
double  alpa;          // thermal difusivity.
alpa=d/(roh*c);
pw.print(" ccc " +(double) (c));
double f;f=alpa*dt/(Math.pow(dx,2));  // fourier number.
double  L=1000.0*Pm[2]; // latent heat of PCM.
double[] t=new double[4];
int pl=1000;
double hc=5.0;
double upcm=5.0;   // heat transfer coefiecint of PCM.
double Apcm=400.0; // total surface area of PCM.
double fg=upcm*dx/d; // biot number.
// all the variables are mentioned in the program DayNET.java
double Ti=15.0;double Tout=12.0;double roha=1.5;double ca=1044.0;double ap=.28;
double NN=3.0;
double Z=6.6;
double Ad=3.0;double Bd=8.0;double Cd=50.0;double Hd=1.5;double Dd=10.0;
Af=Dd*Cd*NN;double Ae=Ad*Cd*NN;
double Aw=Ae;
double An=((2.0*Hd*Bd/3.0)+(Ad*Bd))*NN;
double As=An;
```

```
double v=(((2.0*Hd*Bd*Cd)/3.0)+(Ad*Bd*Cd))*NN;
Ai=Ae+Af+Aw+An+As;
vdot=v*1.5/(3600.0);
pw.println("%the greenhouse with  pcm");
pw.println("   %v   " +(double) (v));
pw.println("   %An   " +(double) (An));
pw.println("   %Ae   " +(double) (Ae));
pw.println("A=[");
double ff=(1+Z*Ai*dt/(v*roha*ca)+dt*vdot/v+dt*Apcm*upcm/(v*roha*ca));
double fm=dt*upcm*Apcm/(v*roha*ca);
double Bi=upcm*dx/d;   // biot number
////end of mentioned variables
double W=Ti+Tout*(Z*Ai*dt/(v*roha*ca)+dt*vdot/v)+dt*Ae*ap*S/(v*roha*ca);
System.out.println("   "+W);
double z=15.0;   // initial temperature of the PCM nodes.
double Fa=d*dt/(roh*L*Math.pow(dx,2));
double DT=((1/(2*alpa*(1+Bi)))*Math.pow(dx,2));
pw.println("%f(1+Bi)"+(double)(DT));
int G=0;
int GR=0;
// solution of [A]*[T]=[b]
//[A] matrix.
double[][] h=new double[Q][Q+1];
h[0][Q]=W;
h[Q-1][Q]=h[0][Q];h[0][0]=ff;
h[Q-1][Q-1]=h[0][0];
h[0][1]=-fm;h[Q-1][Q-2]=h[0][1];
h[1][Q]=(z);h[Q-2][Q]=h[1][Q];
h[1][1]=(1+2*f*(1+Bi)); h[Q-2][Q-2]=h[1][1];
h[1][2]=-2*f;
h[Q-2][Q-3]=h[1][2];
h[1][0]=-2*f*Bi;
h[Q-2][Q-1]=h[1][0];

for(int k=2;k<(Q-2);k++){
h[k][k]=(1+2*f);
```

```
h[k][k+1]=-f;
h[k][k-1]=h[k][k+1];
h[k][Q]=z;}
double[] X=new double[Q];
for(int i=0;i<Q;i++)
{X[i]=z;}
double fl[][]=new double[pl][((Q-3)/2)+1];
double F[][]=new double[pl][((Q-3)/2)+1];
for(int b=0;b<pl;b++)
{
for(int a=0;a<((Q-3)/2)+1;a++)
{
F[b][a]=0.0;              // initial liquid fracitons (solid PCM)
fl[b][a]=0.0;}}
double R[ ][  ]=new double[pl][((Q-1)/2)+1];
for (int i=0;i<pl;i++)
{
for (int j=0;j<((Q-1)/2)+1;j++)
{
R[i][j]=z;              //initial temperatures of the PCM.
}
}
double TNT=12.0;
double TTTout[ ]=new double[15];
double TTout[ ]=new double[11];double H[ ]=new double[11];
double E [ ]=new double[11];double WW[]=new double[11];
double SS[ ]=new double[11];double N[]=new double[11];
// reading Tout ,solar radiation each hour.
// mentioned in DayNET.java.
TTout=find(1.0 ,"febr.dat" ,11);
H=find(2.0 ,"febr.dat" ,11);
E=find(3.0 ,"febr.dat" ,11);
WW=find(4.0 ,"febr.dat" ,11);
N=find(5.0 ,"febr.dat" ,11);
SS=find(6.0 ,"febr.dat" ,11);
pw.println("  ");
```

```
pw.println("  ");
//gauss(double A[][],double[] X,int FALSE,int TRUE,double TOL,int k,int NN,int N)
TTTout=find(7 ,"febr.dat" ,15);
for (int p=1 ; p<289 ; p++)
{
if(p==1){Tout=TTout[0];
S=(H[0]*Af+E[0]*Ae+WW[0]*Aw+N[0]*An+SS[0]*As);}
if(p>1 && p<=120){    // day time
if ((p%(12))==0)
{S=(H[p/12]*Af+E[p/12]*Ae+WW[p/12]*Aw+N[p/12]*An+SS[p/12]*As); //solar
Tout=TTout[p/12];}
}
if(p>=121 && p<=288)
{
//night  time.
vdot=v/(3.0*3600.0);
S=0.0;
ff=(1+Z*Ai*dt/(v*roha*ca)+dt*vdot/v+dt*Apcm*upcm/(v*roha*ca));
h[0][0]=ff;
h[Q-1][Q-1]=h[0][0];
if((p%12)==0.0) {Tout=TTTout[(p/12)-10];}   //Tout each hour
if (p==121){Tout=TTTout[0];}
}
// if needed ventalation
/**
if ((p>=48  && p<=100))
{
upcm=20.0;
fm=(dt*upcm*Apcm)/(v*roha*ca);
Bi=upcm*dx/d;
Fa=d*dt/(roh*L*Math.pow(dx,2));
DT=((1/(2*alpa*(1+Bi)))*Math.pow(dx,2));
ff=(1+(Z*Ai*dt)/(v*roha*ca)+dt*vdot/v+(dt*Apcm*upcm)/(v*roha*ca));

h[0][1]=-fm;h[Q-1][Q-2]=h[0][1];
if(R[p-1][1]!=Tm){
```

```
h[1][1]=(1+2*f*(1+Bi)); h[Q-2][Q-2]=h[1][1];
h[1][0]=-2*f*Bi;
h[Q-2][Q-1]=h[1][0];
h[1][2]=-2*f;
h[Q-2][Q-3]=h[1][2];
h[Q-2][Q-1]=h[1][0];
h[1][Q]=R[p-1][1];
h[Q-2][Q]=h[1][Q];}

h[Q-1][Q]=h[0][Q];h[0][0]=ff;
h[Q-1][Q-1]=h[0][0];
h[0][1]=-fm;h[Q-1][Q-2]=h[0][1];
}
if (p>=180  && p<=288)
{
Bc=3.0;
upcm=20.0;
fm=(dt*upcm*Apcm)/(v*roha*ca);
Bi=upcm*dx/d;
Fa=d*dt/(roh*L*Math.pow(dx,2));
DT=((1/(2*alpa*(1+Bi)))*Math.pow(dx,2));
ff=(1+(Z*Ai*dt)/(v*roha*ca)+dt*vdot/v+(dt*Apcm*upcm)/(v*roha*ca));
h[0][1]=-fm;h[Q-1][Q-2]=h[0][1];
if(R[p-1][1]!=Tm){
h[1][1]=(1+2*f*(1+Bi)); h[Q-2][Q-2]=h[1][1];
h[1][0]=-2*f*Bi;
h[Q-2][Q-1]=h[1][0];
h[1][2]=-2*f;
h[Q-2][Q-3]=h[1][2];
h[Q-2][Q-1]=h[1][0];
h[1][Q]=R[p-1][1];
h[Q-2][Q]=h[1][Q];}
h[Q-1][Q]=h[0][Q];h[0][0]=ff;
h[Q-1][Q-1]=h[0][0];
h[0][1]=-fm;h[Q-1][Q-2]=h[0][1];}
*/ /end of 'if needed ventalation'.
```

```
h[0][Q]=Ti+Tout*(Z*Ai*dt/(v*roha*ca)+dt*vdot/v)+dt*ap*S/(v*roha*ca);
// gauss sidel solution.
matrix= gauss( h, X,0,1,0.00000000000001,1000000,Q);
for(int r=1;r<((Q-1)/2)+1;r++)
{
R[p][r]=matrix[r];
X[r]=matrix[r];
}
X[0]=matrix[0];
for (int i=1;i<((Q-1)/2)+1;i++)
{
X[Q-i]=X[i-1];}
for (int i=0;i<((Q-1)/2)+1;i++){        //out put to file
pw.print("  " +(double) (matrix[i]));}
pw.println("    ;   ");
//updating liguid fraction for the first node.
if(R[p][1]==Tm  &&  fl[p-1][0]<1.0  &&  fl[p-1][0]>0.0)
{
fl[p][0]=fl[p-1][0]+2*Bi*Fa*matrix[0]-2*Fa*(1+Bi)*Tm+2*Fa*R[p][2];
if(fl[p][0]<=0.0)
{fl[p][0]=0.0;}
if(fl[p][0]>=1.0)
{fl[p][0]=1.0;}
}
//updating liguid fraction for the remain nodes.
for(int  k=2;k<(Q-((Q+3)/2))+1;k++)
{
if(R[p][k]==Tm  &&    fl[p-1][k-1]<1.0  &&    fl[p-1][k-1]>0.0)
{
fl[p][k-1]=fl[p-1][k-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][k-1]-2*Tm+R[p][k+1]);
if(fl[p][k-1]<=0.0)
{fl[p][k-1]=0.0;}
if(fl[p][k-1]>=1.0)
{fl[p][k-1]=1.0;}
} ///end  if
} //end for
```

```
////updating liguid fraction for the central  node.
if(R[p][(Q-((Q+3)/2))+1]==Tm  &&  fl[p-1][(Q-((Q+3)/2))]<1.0  &&  fl[p-1][(Q-
((Q+3)/2))]>0.0)
{
fl[p][Q-((Q+3)/2)]=fl[p-1][Q-((Q+3)/2)]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][Q-((Q+3)/2)]-
2*Tm+R[p][Q-((Q+3)/2)]);
if(fl[p][Q-((Q+3)/2)]<=0.0)
{fl[p][Q-((Q+3)/2)]=0.0;}
if(fl[p][Q-((Q+3)/2)]>=1.0)
{fl[p][Q-((Q+3)/2)]=1.0;}
}
//test of melting for the first CV.
if ((R[p][1]>=Tm)  &&   (R[p-1][1]< Tm))
{
fl[p][0]=fl[p-1][0]+2*Bi*Fa*matrix[0]-2*Fa*(1+Bi)*Tm+2*Fa*R[p][2]-c/L*(Tm-R[p-1][1]);
//updating liquid fraction if start of melting .
if (fl[p][0]<=0.0)
{fl[p][0]=0.0;}
if (fl[p][0]>=1.0)
{fl[p][0]=1.0;}
// updating the coefecients of the node.
h[1][0]=0.0;h[1][1]=1.0;h[1][2]=0.0;h[1][Q]=Tm;
h[Q-2][Q-1]=h[1][0];h[Q-2][Q-2]=h[1][1];h[Q-2][Q-3]=h[1][2];h[Q-2][Q]=h[1][Q];
matrix= gauss( h, X,0,1,0.00000000000001,1000000,Q);
//resolving  the problem at this time step.
pw.println("%correctionTT000");
for (int i=0;i<(Q+1)/2;i++){
pw.print("      " +(double) (matrix[i]));}
pw.println("       ");
for(int r=1;r<(Q+1)/2;r++)
{
R[p][r]=matrix[r];
X[r]=matrix[r];
}
X[0]=matrix[0];
for (int i=1;i<((Q-1)/2)+1;i++)
```

```
{
X[Q-i]=X[i-1];}
for(int j=2;j<(Q-((Q+1)/2));j++)
{
if (fl[p-1][j-1]!=1.0){
fl[p][j-1]=fl[p-1][j-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][j-1]-2*Tm+R[p][j+1]);
}
if (fl[p][j-1]<=0.0)
{fl[p][j-1]=0.0;}
if (fl[p][j-1]>=1.0)
{fl[p][j-1]=1.0;}}
}//if
// testing the end of melting at the first node.
if ((fl[p][0]>=1.0)  &&  (fl[p-1][0]< 1.0))
{
G=12;
// if end of melting updat the coeffecient of this node.
h[1][0]=-2*f*Bi;h[1][1]=1+2*f*(1+Bi);h[1][2]=-2*f;h[1][Q]=(Tm-L/c*(1-fl[p-1][0]));
h[Q-2][Q-1]=h[1][0];h[Q-2][Q-2]=h[1][1];h[Q-2][Q-3]=h[1][2];h[Q-2][Q]=h[1][Q];
X[1]=(Tm-L/c*(1-fl[p-1][0]));
//X[q]=(Tm);
X[Q-2]=X[1];
// resolve at the same time step.
matrix= gauss( h, X,0,1,0.00000000000001,10000000,Q);
pw.println("%end of meltingrrrr "+(int)  (p));
for (int i=0;i<5;i++){
pw.print("      " +(double) (matrix[i]));}
pw.println("       ");
for(int r=1;r<(Q-4);r++)
{
R[p][r]=matrix[r];
X[r]=matrix[r];
}
X[0]=matrix[0];
for (int i=1;i<((Q-1)/2)+1;i++)
{
```

```
X[Q-i]=X[i-1];}
}//end if
// test if other nodes start of melting while ending at node 1 .
for(int q=2;q<(Q-((Q-1)/2))-1;q++)
{
if ((R[p][q]>=Tm)  &&   (R[p-1][q]< Tm))
{
fl[p][q-1]=fl[p-1][q-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][q-1]-2*Tm+R[p][q+1])-
(c/L*(Tm-R[p-1][q]));
if (fl[p][q-1]<=0.0)
{fl[p][q-1]=0.0;}
if (fl[p][q-1]>=1.0)
{fl[p][q-1]=1.0;}
h[q][q-1]=0.0;h[q][q]=1.0;h[q][Q]=(Tm);h[q][q+1]=0.0;
h[Q-(q+1)][Q-(q+1)]=1.0;h[Q-(q+1)][Q-(q+2)]=0.0;h[Q-(q+1)][Q]=(Tm);h[Q-(q+1)][Q-
(q)]=0.0;
X[q]=(Tm);
X[Q-(q+1)]=X[q];
matrix= gauss( h, X,0,1,0.00000000000001,1000000,Q);
pw.println("%correctionTT00011111");
for (int i=0;i<(Q+1)/2;i++){
pw.print("      " +(double) (matrix[i]));}
pw.println("       ");
for(int r=1;r<(Q+1)/2;r++)
{
R[p][r]=matrix[r];
X[r]=matrix[r];
}
X[0]=matrix[0];
for (int i=1;i<((Q-1)/2)+1;i++)
{
X[Q-i]=X[i-1];}
for(int j=2;j<(Q-((Q+1)/2));j++)
{
if(j!=q)     //continue;
{
```

```
if (fl[p-1][j-1]!=1.0  ){
fl[p][j-1]=fl[p-1][j-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][j-1]-2*Tm+R[p][j+1]);
}
if (fl[p][j-1]<=0.0)
{fl[p][j-1]=0.0;}

if (fl[p][j-1]>=1.0)
{fl[p][j-1]=1.0;}}}
if (fl[p-1][(Q-((Q+1)/2))-1]!=1.0 && fl[p-1][(Q-((Q+1)/2))-1]!=0.0){
fl[p][(Q-((Q+1)/2))-1]=fl[p-1][(Q-((Q+1)/2))-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][(Q-
((Q+1)/2))-1]-2*Tm+R[p][(Q-((Q+1)/2))-1]);
}
if (fl[p][(Q-((Q+1)/2))-1]<=0.0)
{fl[p][(Q-((Q+1)/2))-1]=0.0;}
if (fl[p][(Q-((Q+1)/2))-1]>=1.0)
{fl[p][(Q-((Q+1)/2))-1]=1.0;}
}  ///end if
}   // end  for
// test of start of melting at the center node.
if ((R[p][(Q-((Q+1)/2))]>=Tm)  &&   (R[p-1][(Q-((Q+1)/2))]< Tm))
{if (fl[p-1][(Q-((Q+1)/2))-1]!=1.0){
fl[p][Q-((Q+1)/2)-1]=fl[p-1][Q-((Q+1)/2)-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][Q-
((Q+1)/2)]-2*Tm+R[p][Q-((Q+1)/2)])-(c/L*(Tm-R[p-1][Q-((Q+1)/2)]));
//correction of liquid fraction if startting of malting.
}
if (fl[p][(Q-((Q+1)/2))-1]<=0.0)
{fl[p][(Q-((Q+1)/2))-1]=0.0;}
if (fl[p][(Q-((Q+1)/2))-1]>=1.0)
{fl[p][(Q-((Q+1)/2))-1]=1.0;}
// update node coefecients.
h[Q-((Q+1)/2)][Q-((Q+1)/2)-1]=0.0;h[Q-((Q+1)/2)][Q-((Q+1)/2)]=1.0;h[Q-
((Q+1)/2)][Q]=(Tm);h[Q-((Q+1)/2)][Q-((Q+1)/2)+1]=0.0;

X[Q-((Q+1)/2)]=(Tm);
X[Q-(Q-((Q+1)/2)+1)]=X[Q-((Q+1)/2)];
// resolve at the same time step.
```

```
matrix= gauss( h, X,0,1,0.00000000000001,1000000,Q);
pw.println("%correctionTT000");
for (int i=0;i<((Q-1)/2+1);i++){
pw.print("      " +(double) (matrix[i]));}
pw.println("       ");
for(int r=1;r<((Q-1)/2+1);r++)
{
R[p][r]=matrix[r];
X[r]=matrix[r];
}
X[0]=matrix[0];
for (int i=1;i<((Q-1)/2)+1;i++)
{
X[Q-i]=X[i-1];}
} ///end if
for(int u=0;u<(Q-3)/2+1;u++)
{if (fl[p][u] <= 0.0)
{
fl[p][u]=0.0;}
if (fl[p][u] >= 1.0){
fl[p][u]=1.0;}
F[p][u]=fl[p][u];}
//test if any CV end of melting
for(int  q=2;q<(Q-((Q+3)/2))+1;q++){
if ((fl[p][q-1]>=1.0)  &&  (fl[p-1][q-1]< 1.0)  &&  (fl[p-1][q-1]>0.0)){
GR=q-1;
int aa=q;
// update the coeffecients of the node that ends of melting.
System.out.print("uiewrtwe");
h[q][q-1]=-f;h[q][q]=(1+2*f);h[q][Q]=(Tm-L/c*(1-fl[p-1][q-1]));h[q][q+1]=-f;
h[Q-(q+1)][Q-(q+1)]=(1+2*f);h[Q-(q+1)][Q-(q+2)]=-f;h[Q-(q+1)][Q]=(Tm-L/c*(1-fl[p-1][q-
1]));h[Q-(q+1)][Q-(q)]=-f;
X[q]=(Tm-L/c*(1-fl[p-1][q-1]));
//X[q]=(Tm);
X[Q-(q+1)]=X[q];
matrix= gauss( h, X,0,1,0.00000000000001,1000000,Q);
```

```
pw.println("%end of meltingccck "+(double)(fl[p-1][q-1]));
for (int i=0;i<5;i++){
pw.print("      " +(double) (matrix[i]));}
pw.println("       ");
for(int r=1;r<(Q-4);r++) {
R[p][r]=matrix[r];
X[r]=matrix[r];}
X[0]=matrix[0];
for (int i=1;i<((Q-1)/2)+1;i++){
X[Q-i]=X[i-1];}
//test for start of melting of other nodes.
for(int e=aa+1;e<(Q-((Q-1)/2))-1;e++)
{
if ((R[p][e]>=Tm)  &&   (R[p-1][e]< Tm))
{
if (fl[p-1][e-1]!=1.0){    /////new *****************
fl[p][e-1]=fl[p-1][e-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][e-1]-2*Tm+R[p][e+1])-(c/L*(Tm-
R[p-1][e]));
}
if (fl[p][e-1]<=0.0)
{fl[p][e-1]=0.0;}
if (fl[p][e-1]>=1.0)
{fl[p][e-1]=1.0;}
h[e][e-1]=0.0;h[e][e]=1.0;h[e][Q]=(Tm);h[e][e+1]=0.0;
h[Q-(e+1)][Q-(e+1)]=1.0;h[Q-(e+1)][Q-(e+2)]=0.0;h[Q-(e+1)][Q]=(Tm);h[Q-(e+1)][Q-(e)]=0.0;
X[e]=(Tm);
X[Q-(e+1)]=X[e];
matrix= gauss( h, X,0,1,0.00000000000001,1000000,Q);
pw.println("%correctionTTdsa"+(double) (fl[p][e-1]));
for (int i=0;i<(Q+1)/2;i++){
pw.print("      " +(double) (matrix[i]));}
pw.println("       ");
for(int r=1;r<(Q+1)/2;r++)
{
R[p][r]=matrix[r];
X[r]=matrix[r];
```

```
}
X[0]=matrix[0];
for (int i=1;i<((Q-1)/2)+1;i++)
{
X[Q-i]=X[i-1];}
for(int j=aa;j<(Q-((Q+1)/2));j++)
{
if(j!=e)     //continue;
{
if (R[p][j]==Tm   &&   fl[p-1][j-1]!=1.0){
fl[p][j-1]=fl[p-1][j-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][j-1]-2*Tm+R[p][j+1]);
}
if (fl[p][j-1]<=0.0)
{fl[p][j-1]=0.0;}

if (fl[p][j-1]>=1.0)
{fl[p][j-1]=1.0;}
}
}
if (R[p][(Q-((Q+1)/2))-1]==Tm  &&   fl[p-1][(Q-((Q+1)/2))-1]!=1.0){
fl[p][(Q-((Q+1)/2))-1]=fl[p-1][(Q-((Q+1)/2))-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][(Q-
((Q+1)/2))-1]-2*Tm+R[p][(Q-((Q+1)/2))-1]);
}
if (fl[p][(Q-((Q+1)/2))-1]<=0.0)
{fl[p][(Q-((Q+1)/2))-1]=0.0;}
if (fl[p][(Q-((Q+1)/2))-1]>=1.0)
{fl[p][(Q-((Q+1)/2))-1]=1.0;}
}  ///end if
}   // end  for
}   ///if
}//for
//test for end of melting at the central CV .
if ((fl[p][(Q-((Q+1)/2))-1]>=1.0)  &&  (fl[p-1][(Q-((Q+1)/2))-1]<1.0))
{
// update coeffecient  if the CV endes of melting.
h[(Q-((Q+1)/2))][(Q-((Q+1)/2))]=(1+2*f);h[(Q-((Q+1)/2))][(Q-((Q+1)/2))-1]=-f;h[(Q-
```

```
((Q+1)/2))][Q]=(Tm-L/c*(1-fl[p-1][(Q-((Q+1)/2))-2]));h[(Q-((Q+1)/2))][(Q-((Q+1)/2))+1]=-f;
X[(Q-((Q+1)/2))]=(Tm-L/c*(1-fl[p-1][(Q-((Q+1)/2))-1]));  //////****take  care ****
matrix= gauss( h, X,0,1,0.00000000000001,1000000,Q);
pw.println("%end of melting "+(double)((Q-((Q+1)/2))));
for (int i=0;i<(Q-4);i++){
pw.print("      " +(double) (matrix[i]));}
pw.println("       ");
for(int r=1;r<(Q-4);r++)
{
R[p][r]=matrix[r];
X[r]=matrix[r];
}
X[0]=matrix[0];
for (int i=1;i<((Q-1)/2)+1;i++)
{
X[Q-i]=X[i-1];}
}//if
for(int u=0;u<(Q-3)/2+1;u++)
{
if (fl[p][u] <= 0.0)
{
fl[p][u]=0.0;}
if (fl[p][u] >= 1.0){
fl[p][u]=1.0;}
F[p][u]=fl[p][u];
}
//TEST FOR SOLIDIFICATIN.
TNT=12.0;
double NT=78.0;
double TN=20.0;
fl[p][0]=fl[p-1][0]+2*Bi*Fa*matrix[0]-2*Fa*(1+Bi)*Tm+2*Fa*R[p][2];
for(int j=2;j<(Q-((Q+1)/2));j++)
{
fl[p][j-1]=fl[p-1][j-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][j-1]-2*Tm+R[p][j+1]);
}
fl[p][(Q-((Q+1)/2))-1]=fl[p-1][(Q-((Q+1)/2))-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][(Q-
```

```
((Q+1)/2))-1]-2*Tm+R[p][(Q-((Q+1)/2))-1]);
for(int u=0;u<(Q-3)/2+1;u++)
{
if (fl[p][u] <= 0.0)
{
fl[p][u]=0.0;}
if (fl[p][u] >= 1.0){
fl[p][u]=1.0;}
F[p][u]=fl[p][u];
}
//test for starting of soidification
if ((R[p][1]<=Tm)  &&   (R[p-1][1]> Tm))
{
// update fl if start of solidification
fl[p][0]=fl[p-1][0]+2*Bi*Fa*matrix[0]-2*Fa*(1+Bi)*Tm+2*Fa*R[p][2]-c/L*(Tm-R[p-1][1]);
if (fl[p-1][0]<=0.0)
{fl[p][0]=0.0;}
if (fl[p][0]>=1.0)
{fl[p][0]=1.0;}
//update node  coefficients.
h[1][0]=0.0;h[1][1]=1.0;h[1][2]=0.0;h[1][Q]=Tm;
h[Q-2][Q-1]=h[1][0];h[Q-2][Q-2]=h[1][1];h[Q-2][Q-3]=h[1][2];h[Q-2][Q]=h[1][Q];
//resolve at the same time step.
matrix= gauss( h, X,0,1,0.00000000000001,1000000,Q);
pw.println("%correctionTT000");
for (int i=0;i<(Q+1)/2;i++){              // print the result.
pw.print("      " +(double) (matrix[i]));}
pw.println("       ");
for(int r=1;r<(Q+1)/2;r++)
{
R[p][r]=matrix[r];
X[r]=matrix[r];
}
X[0]=matrix[0];
for (int i=1;i<((Q-1)/2)+1;i++)
{
```

```
X[Q-i]=X[i-1];}
for(int j=2;j<(Q-((Q+1)/2));j++)
{
if (fl[p-1][j-1]!=0.0){
fl[p][j-1]=fl[p-1][j-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][j-1]-2*Tm+R[p][j+1]);
}
if (fl[p][j-1]<=0.0)
{fl[p][j-1]=0.0;}
if (fl[p][j-1]>=1.0)
{fl[p][j-1]=1.0;}
}
fl[p][(Q-((Q+1)/2))-1]=fl[p-1][(Q-((Q+1)/2))-2]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][(Q-
((Q+1)/2))-1]-2*Tm+R[p][(Q-((Q+1)/2))-1]);
if (fl[p][(Q-((Q+1)/2))-1]<=0.0)
{fl[p][(Q-((Q+1)/2))-1]=0.0;}
if (fl[p][(Q-((Q+1)/2))-1]>=1.0)
{fl[p][(Q-((Q+1)/2))-1]=1.0;}
}//if
// testing start of solidification at other nodes
for( q=2;q<(Q-((Q+1)/2));q++)
{
if ((R[p][q]<=Tm)  &&   (R[p-1][q]> Tm))
{
//// update fl if start of solidification
if (fl[p-1][q-1]!=0.0){
fl[p][q-1]=fl[p-1][q-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][q-1]-2*Tm+R[p][q+1])-
(c/L*(Tm+R[p-1][q]));
}
if (fl[p][q-1] <= 0.0)
{
fl[p][q-1]=0.0;}
if (fl[p][q-1] >= 1.0){
fl[p][q-1]=1.0;}
// update coefficients of the node if starting of solidification
h[q][q-1]=0.0;h[q][q]=1.0;h[q][Q]=Tm;h[q][q+1]=0.0;
h[Q-(q+1)][Q-(q+1)]=1.0;h[Q-(q+1)][Q-(q+2)]=0.0;h[Q-(q+1)][Q]=Tm;h[Q-(q+1)][Q-(q)]=0.0;
```

```
X[q]=Tm;
X[Q-(q+1)]=X[q];
// resolve at the same time step
matrix= gauss( h, X,0,1,0.00000000000001,1000000,Q);
//printX(matrix);
for(int r=1;r<(Q-((Q-1)/2));r++)
{
R[p][r]=matrix[r];
X[r]=matrix[r];
}
X[0]=matrix[0];
for(int i=1;i<((Q-1)/2+1);i++)
{
X[Q-i]=X[i-1];}
System.out.println("  ");
pw.println("%correctionTTT "+(double) (q) );
for (int i=0;i<(Q-((Q-1)/2));i++){
pw.print("%13" +(double) (matrix[i]));}
pw.println("       ");
for(int j=2;j<(Q-((Q+1)/2));j++)
{
if (fl[p-1][j-1]!=0.0){
//if(j==q) continue;
if (j!=q)
fl[p][q-1]=fl[p-1][q-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][q-1]-2*Tm+R[p][q+1]);
}
}
if (fl[p-1][(Q-((Q+1)/2))-1]!=0.0  ){
fl[p][(Q-((Q+1)/2))-1]=fl[p-1][(Q-((Q+1)/2))-2]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][(Q-
((Q+1)/2))-1]-2*Tm+R[p][(Q-((Q+1)/2))-1]);
}
for(int u=0;u<(Q-3)/2+1;u++)
{
if (fl[p][u] <= 0.0)
{
fl[p][u]=0.0;}
```

```
if (fl[p][u] >= 1.0){
fl[p][u]=1.0;}
F[p][u]=fl[p][u];
}
}///if
}///for
//test of starting of solidification at the center of PCM.
if ((R[p][(Q-((Q+1)/2))]<=Tm)   &&  ( R[p-1][(Q-((Q+1)/2))]>Tm))
{
//update liquid fraction
fl[p][(Q-((Q+1)/2))-1]=fl[p-1][(Q-((Q+1)/2))-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][(Q-
((Q+1)/2))-1]-2*Tm+R[p][(Q-((Q+1)/2))-1])-(c/L*(Tm-R[p-1][(Q-((Q+1)/2))]));
h[(Q-((Q+1)/2))][(Q-((Q+1)/2))]=1.0;h[(Q-((Q+1)/2))][(Q-((Q+1)/2))-1]=0;h[(Q-
((Q+1)/2))][Q]=Tm;h[(Q-((Q+1)/2))][(Q-((Q+1)/2))+1]=0;
matrix= gauss( h, X,0,1,0.00000000000001,1000000,Q);
pw.println("%correctionTT"+(double) ((Q-((Q+1)/2))));
for (int i=0;i<(Q-3)/2+2;i++){
pw.print("%14" +(double) (matrix[i]));}
pw.println("    ;   ");
for(int r=1;r<(Q-3)/2+2;r++)
{
R[p][r]=matrix[r];
X[r]=matrix[r];
}
X[0]=matrix[0];
for(int i=1;i<((Q-1)/2+1);i++)
{
X[Q-i]=X[i-1];}
}//if 1
for(int u=0;u<(Q-3)/2+1;u++)
{
if (fl[p][u] <= 0.0)
{
fl[p][u]=0.0;}
if (fl[p][u] >= 1.0){
fl[p][u]=1.0;}
```

```
F[p][u]=fl[p][u];
}
//test for end of solidification at node 1 .
if ((fl[p][0]<=0.0)  &&  (fl[p-1][0]>0.0)  )
{
G=4;
// update the coefficients of the node.
h[1][0]=-2*f*Bi;h[1][1]=1+2*f*(1+Bi);h[1][2]=-2*f;h[1][Q]=(Tm+L/c*(fl[p-1][0]));
h[Q-2][Q-1]=h[1][0];h[Q-2][Q-2]=h[1][1];h[Q-2][Q-3]=h[1][2];h[Q-2][Q]=h[1][Q];
X[1]=(Tm+L/c*(fl[p-1][0]));
//X[q]=(Tm);
X[Q-2]=X[1];
//resolve at the same time step
matrix= gauss( h, X,0,1,0.00000000000001,1000000,Q);
pw.println("%end of solidificationoneSS "+(double) fl[p-1][0]);
for (int i=0;i<(Q-4);i++){
pw.print("      " +(double) (matrix[i]));}
pw.println("       ");
for(int r=1;r<(Q-4);r++)
{
R[p][r]=matrix[r];
X[r]=matrix[r];
}
X[0]=matrix[0];
for (int i=1;i<((Q-1)/2)+1;i++)
{
X[Q-i]=X[i-1];}
for(int j=2;j<(Q-((Q+1)/2));j++)
{
if (fl[p-1][j-1]!=0.0){
//if(j==q) continue;
fl[p][j-1]=fl[p-1][j-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][j-1]-2*Tm+R[p][j+1]);
}
pw.println("%15" +(double) (fl[p][j-1]));
}
if (fl[p-1][(Q-((Q+1)/2))-1]!=0.0){
```

```
fl[p][(Q-((Q+1)/2))-1]=fl[p-1][(Q-((Q+1)/2))-2]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][(Q-
((Q+1)/2))-1]-2*Tm+R[p][(Q-((Q+1)/2))-1]);
}
//test for start of solidification while ending at first CV.
for( q=2;q<(Q-((Q+1)/2));q++)
{
if ((R[p][q]<=Tm)  &&   (R[p-1][q]> Tm))
{
if (fl[p-1][q-1]!=0.0){
fl[p][q-1]=fl[p-1][q-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][q-1]-2*Tm+R[p][q+1])-
(c/L*(Tm+R[p-1][q]));
}
if (fl[p][q-1] <= 0.0)
{
fl[p][q-1]=0.0;}
if (fl[p][q-1] >= 1.0){
fl[p][q-1]=1.0;}
h[q][q-1]=0.0;h[q][q]=1.0;h[q][Q]=Tm;h[q][q+1]=0.0;
h[Q-(q+1)][Q-(q+1)]=1.0;h[Q-(q+1)][Q-(q+2)]=0.0;h[Q-(q+1)][Q]=Tm;h[Q-(q+1)][Q-(q)]=0.0;
X[q]=Tm;
X[Q-(q+1)]=X[q];
matrix= gauss( h, X,0,1,0.00000000000001,1000000,Q);
for(int r=1;r<(Q-((Q-1)/2));r++)
{
R[p][r]=matrix[r];
X[r]=matrix[r];
}
X[0]=matrix[0];
for(int i=1;i<((Q-1)/2+1);i++)
{
X[Q-i]=X[i-1];}
System.out.println("  ");
pw.println("%correctionTTT "+(double) (q) );
for (int i=0;i<(Q-((Q-1)/2));i++){
pw.print("%13" +(double) (matrix[i]));}
pw.println("       ");
```

```
for(int j=2;j<(Q-((Q+1)/2));j++)
{
if (fl[p-1][j-1]!=0.0){
//if(j==q) continue;
if (j!=q)
fl[p][q-1]=fl[p-1][q-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][q-1]-2*Tm+R[p][q+1]);
}
}
if (fl[p-1][(Q-((Q+1)/2))-1]!=0.0){
fl[p][(Q-((Q+1)/2))-1]=fl[p-1][(Q-((Q+1)/2))-2]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][(Q-
((Q+1)/2))-1]-2*Tm+R[p][(Q-((Q+1)/2))-1]);
}
for(int u=0;u<(Q-3)/2+1;u++)
{
if (fl[p][u] <= 0.0)
{
fl[p][u]=0.0;}
if (fl[p][u] >= 1.0){
fl[p][u]=1.0;}
F[p][u]=fl[p][u];
}
}///if
}///for
}//end if
//test of ending of solidification at other nodes.
for( q=2;q<(Q-((Q+1)/2));q++)
{
// update the liquid fraction
if ((fl[p][q-1]<=0.0)  &&  (fl[p-1][q-1]>0.0)  &&  (fl[p-1][q-1]<1.0))
{
int MP=q;
//update the CV coefficients.
h[q][q-1]=-f;h[q][q]=(1+2*f);h[q][Q]=(Tm+(L/c*(fl[p-1][q-1])));h[q][q+1]=-f;
h[Q-(q+1)][Q-(q+1)]=(1+2*f);h[Q-(q+1)][Q-(q+2)]=-f;h[Q-(q+1)][Q]=h[q][Q];h[Q-(q+1)][Q-
(q)]=-f;
X[q]=(Tm+(L/c*(fl[p-1][q-1])));
```

```
X[Q-(q+1)]=X[q];
//resolve at the same time step.
matrix= gauss( h, X,0,1,0.00000000000001,1000000,Q);
pw.println("%end of solidifcationth"+(double) (fl[p-1][q-1]));
for (int i=0;i<(Q-((Q-1)/2));i++){
pw.print("%15" +(double) (matrix[i]));
}
pw.println("    ;   ");
for(int r=1;r<(Q-3)/2+2;r++)
{
R[p][r]=matrix[r];
X[r]=matrix[r];
}
X[0]=matrix[0];
for(int i=1;i<((Q-1)/2+1);i++)
{
X[Q-i]=X[i-1];}
//update liquid fractions
for(int j=2;j<(Q-((Q+1)/2));j++)
{
if (fl[p-1][j-1]!=0.0){
if(j==q) continue;
fl[p][q-1]=fl[p-1][q-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][q-1]-2*Tm+R[p][q+1]);
}
}
if (fl[p-1][(Q-((Q+1)/2))-1]!=0.0){
fl[p][(Q-((Q+1)/2))-1]=fl[p-1][(Q-((Q+1)/2))-2]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][(Q-
((Q+1)/2))-1]-2*Tm+R[p][(Q-((Q+1)/2))-1]);
}
//test start of solidification
for(int e=MP;e<(Q-((Q-1)/2))-1;e++)
{
if ((R[p][e]<=Tm)  &&   (R[p-1][e]> Tm))
{
if (fl[p][e-1]!=0.0){    /////new *****************
fl[p][e-1]=fl[p-1][e-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][e-1]-2*Tm+R[p][e+1])-(c/L*(Tm-
```

```
R[p-1][e]));
}
if (fl[p][e-1]<=0.0)
{fl[p][e-1]=0.0;}
if (fl[p][e-1]>=1.0)
{fl[p][e-1]=1.0;}
h[e][e-1]=0.0;h[e][e]=1.0;h[e][Q]=(Tm);h[e][e+1]=0.0;
h[Q-(e+1)][Q-(e+1)]=1.0;h[Q-(e+1)][Q-(e+2)]=0.0;h[Q-(e+1)][Q]=(Tm);h[Q-(e+1)][Q-(e)]=0.0;
X[e]=(Tm);
X[Q-(e+1)]=X[e];
matrix= gauss( h, X,0,1,0.00000000000001,1000000,Q);
pw.println("%correctionTT000");
for (int i=0;i<(Q+1)/2;i++){
pw.print("      " +(double) (matrix[i]));}
pw.println("       ");
for(int r=1;r<(Q+1)/2;r++)
{
R[p][r]=matrix[r];
X[r]=matrix[r];
}
X[0]=matrix[0];
for (int i=1;i<((Q-1)/2)+1;i++)
{
X[Q-i]=X[i-1];}
for(int j=2;j<(Q-((Q+1)/2));j++)
{
if(j!=e)     //continue;
{
if (fl[p][j-1]!=0.0){
fl[p][j-1]=fl[p-1][j-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][j-1]-2*Tm+R[p][j+1]);
}
if (fl[p][j-1]<=0.0)
{fl[p][j-1]=0.0;}

if (fl[p][j-1]>=1.0)
{fl[p][j-1]=1.0;}
```

```
}
}
if (fl[p-1][(Q-((Q+1)/2))-1]!=0.0){
fl[p][(Q-((Q+1)/2))-1]=fl[p-1][(Q-((Q+1)/2))-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][(Q-
((Q+1)/2))-1]-2*Tm+R[p][(Q-((Q+1)/2))-1]);
}
if (fl[p][(Q-((Q+1)/2))-1]<=0.0)
{fl[p][(Q-((Q+1)/2))-1]=0.0;}

if (fl[p][(Q-((Q+1)/2))-1]>=1.0)
{fl[p][(Q-((Q+1)/2))-1]=1.0;}
}  ///end if
}   // end  for
}//if
}//for
//test of end of solidification at the center.
if ((fl[p][(Q-((Q+1)/2))-1]>=0.0)  &&  (fl[p-1][(Q-((Q+1)/2))-1]<0.0))
{
h[(Q-((Q+1)/2))][(Q-((Q+1)/2))]=(1+2*f);h[(Q-((Q+1)/2))][(Q-((Q+1)/2))-1]=-f;h[(Q-
((Q+1)/2))][Q]=(Tm-L/c*(-fl[l-1][(Q-((Q+1)/2))-1]));h[(Q-((Q+1)/2))][(Q-((Q+1)/2))+1]=-f;
X[(Q-((Q+1)/2))]=(Tm-L/c*(-fl[p-1][(Q-((Q+1)/2))-1]));   ////take  care******
//X[(Q-((Q+1)/2))]=(Tm);
matrix= gauss( h, X,0,1,0.00000000000001,1000000,Q);
pw.println("%end of solidifcation"+(double) ((Q-((Q+1)/2))));
for (int i=0;i<(Q-((Q-1)/2));i++){
pw.print("   16   " +(double) (matrix[i]));
}
for(int r=1;r<(Q-3)/2+2;r++)
{
R[p][r]=matrix[r];
X[r]=matrix[r];
}
X[0]=matrix[0];
for(int i=1;i<((Q-1)/2+1);i++)
{X[Q-i]=X[i-1];}
fl[p+1][(Q-((Q+1)/2))-1]=0.0;
```

```
fl[p-1][(Q-((Q+1)/2))-1]=0.0;
NT=99.0;
for(int u=0;u<(Q-3)/2+1;u++)
{
if (fl[p][u] <=0.0)
{
fl[p][u]=0.0;}
if (fl[p][u] >= 1.0){
fl[p][u]=1.0;}
F[p][u]=fl[p][u];
}
}//// if
for(int u=0;u<(Q-3)/2+1;u++)
{
if (fl[p][u] <= 0.0)
{
fl[p][u]=0.0;}
if (fl[p][u] >= 1.0){
fl[p][u]=1.0;}
F[p][u]=fl[p][u];
}
Ti=matrix[0];
h[1][Q]=R[p][1];
h[Q-2][Q]=h[1][Q];
for(int i=3;i<((Q-1)/2+1);i++)
{
h[i-1][Q]=R[p][i-1];
h[Q-i][Q]=h[i-1][Q];}
h[(Q-((Q+1)/2))][Q]=R[p][(Q-((Q+1)/2))];
if(p>1 && p<=120){
if ((p%(12))==0)
{S=(H[p/12]*Af+E[p/12]*Ae+WW[p/12]*Aw+N[p/12]*An+SS[p/12]*As);
Tout=TTout[p/12];}
}
if(p>=121 && p<=288)
{
```

```
vdot=v/(3600.0);
S=0.0;
ff=(1+Z*Ai*dt/(v*roha*ca)+dt*vdot/v+dt*Apcm*upcm/(v*roha*ca));
h[0][0]=ff;
h[Q-1][Q-1]=h[0][0];
if((p%12)==0.0) {Tout=TTTout[(p/12)-10];}
if (p==121){Tout=TTTout[0];}
}
Ti=matrix[0];
}  //end  main for loop
pw.println("];");
pw.print("plot(A(:,1))");
if (pw!=null) pw.close();
} //end main**********************

///METHODS  USED
// gauss sidel itterative procedure
public static double[]  gauss(double A[][],double[] X,int FALSE,int TRUE,double TOL,int
NN,int N)
{
int k=1;
double b[]=new double[N];
int OK=0 ;
double ERR;
while (( OK == FALSE)  &&   (k<= NN ))
{
ERR = 0.0;
for (int I = 0 ;I< N;I++)
{
double S = 0.0;
for (int J = 0 ;J< N;J++)
{
S = S-A[I][J]*X[J];
}
S = (S+A[I][N])/(A[I][I]);
if (Math.abs(S) > ERR)
```

```
{
ERR  = Math.abs(S);
}
X[I] = X[I] + S;
}
// process is complete
if (ERR > TOL)
{
OK=FALSE;
k=k+1;}
else if (ERR <= TOL)  break;
}
for (int o=0;o<N;o++)
{
b[o]=X[o];
}
return (double[])( b);
}   //end METHOD  GAUSS ITERATION
public static void printX(double[] m,int d)
{
for(int i=0;i<d;i++)
System.out.print("   "+m[i]);
}
public static void prinXx(double[][] m,int d)
{
for(int i=0;i<d;i++)
{
for(int j=0;j<(d+1);j++)
System.out.print("   "+m[i][j]);
}
System.out.println("");
}
//file reader  ***************************
public static double[] find(double g ,String  fil,int x )
{
FileReader frs =null;
```

```
StreamTokenizer in=null;
String sname="slam";
double [] r=new double[x];
double mid=0.0;
try{
frs=new FileReader(fil);
in=new StreamTokenizer(frs);
while(in.ttype!=StreamTokenizer.TT_EOF)
{
if (in.ttype==StreamTokenizer.TT_WORD)
sname=in.sval;
else
while (in.nextToken()==StreamTokenizer.TT_NUMBER)
{
mid=in.nval;
if (mid==g){
for (int i=0;i<x;i++)
{
if (in.nextToken()==StreamTokenizer.TT_NUMBER)
mid=in.nval;
r[i]=mid;
//System.out.println(sname+" "  +r[i]);
//in.nextToken();
}
}
}
in.nextToken();}
}
catch(FileNotFoundException ex) {
System.out.println("receved" );
}
catch(IOException ex)
{
System.out.println(ex.getMessage());
}
finally
```

```
{
try
{
if (frs!=null) frs.close();
}
catch(IOException ex)
{
System.out.println(ex);
}
}
return r;
}
};
```

A.3 Constant convectional heat transfer at end walls source code:

```
import java.io.*;
public class  ConvLIQSOL
{
private static   int t;
private static   int Q;
private  static   int l;
private static   double[] matrix=new double[Q];
private static   int q;
public static void main(String[]  args)
{
PrintWriter pw=null;
try
{
pw=new PrintWriter(new FileOutputStream("d:/resaleh/convconstant.dat"));
}
catch (IOException ex){}
pw.println("  " );
double TTT=0.0;;
int OK;
Q=13;
double[] Pm=new double[5];
```

```
Pm=find(3.0 ,"d:/resaleh/pcm.dat" ,5);
for (int tm=0;tm<5;tm++){
pw.print("%  " +(double) (Pm[tm]));
}
double Tm=Pm[0];          //pcm melting temperature
double dt=5.0;            // time step
double  dx=0.002;         //space between nodes
double d=Pm[4];           // pcm conductivity
double c=1000.0*Pm[3]; // pcm specific heat
double roh=Pm[1];      // pcm density
double  alpa;
alpa=d/(roh*c);        // thermal diffusivity
pw.print(" ccc " +(double) (c));
double f ; f=alpa*dt/(Math.pow(dx,2));
double  L=1000.0*Pm[2];    // pcm latent heat
double[] t=new double[4];
int pl=1000;
double upcm=16.0; //convectional heat transfer between pcm and air
double fg=upcm*dx/d;
double Ti=20.0;
pw.println("A=[");
double Bi=upcm*dx/d;                    // Biot number
double B=Bi;
double z=17.4;                          // initial condition
double Fa=d*dt/(roh*L*Math.pow(dx,2));
double DT=((1/(2*alpa*(1+Bi)))*Math.pow(dx,2));
pw.println("%f(1+Bi)"+(double)(DT));
int G=0;
int GR=0;
double[][] h=new double[Q][Q+1];
// coefficients of  node 1 and 2
h[0][Q]=60.0;// boundary condition for melting
h[Q-1][Q]=h[0][Q];h[0][0]=1.0;
h[Q-1][Q-1]=h[0][0];
h[0][1]=0.0;h[Q-1][Q-2]=h[0][1];
h[1][Q]=(z);h[Q-2][Q]=h[1][Q];
```

```
h[1][1]=(1+2*f*(1+Bi)); h[Q-2][Q-2]=h[1][1];
h[1][2]=-2*f;
h[Q-2][Q-3]=h[1][2];
h[1][0]=-2*f*Bi;
h[Q-2][Q-1]=h[1][0];
// coefficients of internal  nodes
for(int k=2;k<(Q-2);k++){
    h[k][k]=(1+2*f);
    h[k][k+1]=-f;
    h[k][k-1]=h[k][k+1];
    h[k][Q]=z;}
double[] X=new double[Q];
for(int i=0;i<Q;i++)
   {X[i]=z;}
pl=4000;
 double fl[][]=new double[pl][((Q-3)/2)+1];
 double F[][]=new double[pl][((Q-3)/2)+1];
 for(int b=0;b<pl;b++)
 {
for(int a=0;a<((Q-3)/2)+1;a++)
 {
 F[b][a]=0.0;
 fl[b][a]=0.0;
 }
}
double R[][]=new double[pl][((Q-1)/2)+1];
for (int i=0;i<pl;i++)
{
for (int j=0;j<((Q-1)/2)+1;j++)
{
R[i][j]=z;
}
}
double TNT=12.0;
pw.println("  ");
//gauss(double A[][],double[] X,int FALSE,int TRUE,double TOL,int k,int NN,int N)
```

```
 int gt=3000;
for (int p=1 ; p<1560 ; p++)
{
if (p>1560)                    // boundary condition for solidification
  { h[0][Q]=10.0;}
 matrix= gauss( h, X,0,1,0.00000000000001,10000,Q);
for(int r=1;r<((Q-1)/2)+1;r++)
 {
 R[p][r]=matrix[r];
  X[r]=matrix[r];
 }
X[0]=matrix[0];
for (int i=1;i<((Q-1)/2)+1;i++)
{
 X[Q-i]=X[i-1];}
for (int i=0;i<((Q-1)/2)+1;i++){
pw.print("  " +(double) (matrix[i]));}
pw.println("    ;   ");
// update liquid fraction of node 1  after start of melting
if(R[p][1]==Tm  &&  fl[p-1][0]<1.0  && fl[p-1][0]>0.0)
{
if (fl[p-1][0]< fl[p-2][0]){
fl[p][0]=fl[p-1][0]+2*Bi*Fa*matrix[0]-2*Fa*(1+Bi)*Tm+2*Fa*R[p][2];
}
if (fl[p-1][0]> fl[p-2][0]){
fl[p][0]=fl[p-1][0]+2*Bi*Fa*matrix[0]-2*Fa*(1+Bi)*Tm+2*Fa*R[p][2];
}
if(fl[p][0]<=0.0)
{fl[p][0]=0.0;}
if(fl[p][0]>=1.0)
{fl[p][0]=1.0;}
}
// update liquid fraction of internal nodes  after start of melting
for(int  k=2;k<(Q-((Q+3)/2))+1;k++)
{
if(R[p][k]==Tm  &&    fl[p-1][k-1]<1.0  && fl[p-1][k-1]>0.0)
```

```
{
if(fl[p-1][k-1]>fl[p-2][k-1]){
fl[p][k-1]=fl[p-1][k-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][k-1]-2*Tm+R[p][k+1]);
}
if(fl[p-1][k-1]<fl[p-2][k-1]){
fl[p][k-1]=fl[p-1][k-1]+dl*dt/(rohl*L*Math.pow(dx,2))*(R[p][k-1]-2*Tm+R[p][k+1]);
}
if(fl[p][k-1]<=0.0)
{fl[p][k-1]=0.0;}
if(fl[p][k-1]>=1.0)
{fl[p][k-1]=1.0;}
} ///end  if
} //end for
// update liquid fraction of central  node   after start of melting
if(R[p][(Q-((Q+3)/2))+1]==Tm  &&  fl[p-1][(Q-((Q+3)/2))]<1.0 && fl[p-1][(Q-((Q+3)/2))]>0.0)
{
fl[p][Q-((Q+3)/2)]=fl[p-1][Q-((Q+3)/2)]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][Q-((Q+3)/2)]-
2*Tm+R[p][Q-((Q+3)/2)]);
if(fl[p][Q-((Q+3)/2)]<=0.0)
{fl[p][Q-((Q+3)/2)]=0.0;}
if(fl[p][Q-((Q+3)/2)]>=1.0)
{fl[p][Q-((Q+3)/2)]=1.0;}
}
// check start of melting for node 1
if ((R[p][1]>=Tm)  && (R[p-1][1]< Tm))
{
fl[p][0]=fl[p-1][0]+2*Bi*Fa*matrix[0]-2*Fa*(1+Bi)*Tm+2*Fa*R[p][2]-c/L*(Tm-R[p-1][1]);
if (fl[p][0]<=0.0)
  {fl[p][0]=0.0;}
if (fl[p][0]>=1.0)
  {fl[p][0]=1.0;}
// update node 1 coefficients
h[1][0]=0.0;h[1][1]=1.0;h[1][2]=0.0;h[1][Q]=Tm;
h[Q-2][Q-1]=h[1][0];h[Q-2][Q-2]=h[1][1];h[Q-2][Q-3]=h[1][2];h[Q-2][Q]=h[1][Q];
matrix= gauss( h, X,0,1,0.00000000000001,1000000,Q);
pw.println("%correctionTT000");
```

```
for (int i=0;i<(Q+1)/2;i++){
pw.print("      " +(double) (matrix[i]));}
pw.println("       ");
 for(int r=1;r<(Q+1)/2;r++)
 {
 R[p][r]=matrix[r];
  X[r]=matrix[r];
 }
X[0]=matrix[0];
for (int i=1;i<((Q-1)/2)+1;i++)
{
 X[Q-i]=X[i-1];}
for(int j=2;j<(Q-(((Q+1)/2));j++)
{
if (fl[p-1][j-1]!=1.0){
fl[p][j-1]=fl[p-1][j-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][j-1]-2*Tm+R[p][j+1]);
}
if (fl[p][j-1]<=0.0)
  {fl[p][j-1]=0.0;}
if (fl[p][j-1]>=1.0)
  {fl[p][j-1]=1.0;}
}
}//if
// check start of solidification for node 1
if ((R[p][1]<=Tm)  && (R[p-1][1]> Tm))
{
fl[p][0]=fl[p-1][0]+2*Bi*Fa*matrix[0]-2*Fa*(1+Bi)*Tm+2*Fa*R[p][2]-c/L*(Tm-R[p-1][1]);
if (fl[p][0]<=0.0)
  {fl[p][0]=0.0;}
if (fl[p][0]>=1.0)
  {fl[p][0]=1.0;}
// update coefficients of node 1.
h[1][0]=0.0;h[1][1]=1.0;h[1][2]=0.0;h[1][Q]=Tm;
h[Q-2][Q-1]=h[1][0];h[Q-2][Q-2]=h[1][1];h[Q-2][Q-3]=h[1][2];h[Q-2][Q]=h[1][Q];
matrix= gauss( h, X,0,1,0.00000000000001,1000000,Q);
pw.println("%correctionTT000");
```

```
for (int i=0;i<(Q+1)/2;i++){
pw.print("      " +(double) (matrix[i]));}
pw.println("       ");
for(int r=1;r<(Q+1)/2;r++)
 {
 R[p][r]=matrix[r];
  X[r]=matrix[r];
 }
X[0]=matrix[0];
for (int i=1;i<((Q-1)/2)+1;i++)
{
 X[Q-i]=X[i-1];}
for(int j=2;j<(Q-((Q+1)/2));j++)
{
if (fl[p-1][j-1]!=0.0){
fl[p][j-1]=fl[p-1][j-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][j-1]-2*Tm+R[p][j+1]);
}
if (fl[p][j-1]<=0.0)
  {fl[p][j-1]=0.0;}
if (fl[p][j-1]>=1.0)
  {fl[p][j-1]=1.0;}
}
}//if
// check end of melting of node 1
if ((fl[p][0]>=1.0)  &&  (fl[p-1][0]< 1.0))
{
 G=12;
// update coefficients of node 1
h[1][0]=(-2*f*Bi);h[1][1]=(1+2*f*(1+Bi));h[1][2]=-2*f;h[1][Q]=(Tm-L/c*(1-fl[p-1][0]));
h[Q-2][Q-1]=h[1][0];h[Q-2][Q-2]=h[1][1];h[Q-2][Q-3]=h[1][2];h[Q-2][Q]=h[1][Q];
X[1]=(Tm-L/c*(1-fl[p-1][0]));
X[Q-2]=X[1];
 matrix= gauss( h, X,0,1,0.00000000000001,10000000,Q);
pw.println("%end of meltingrrrr "+(int)  (p));
for (int i=0;i<((Q-1)/2)+1;i++){
pw.print("      " +(double) (matrix[i]));}
```

```
pw.println("       ");
for(int i=p;i<gt;i++)
{ fl[i][0]=1.0;}
for(int r=1;r<((Q-1)/2)+1;r++)
 {
 R[p][r]=matrix[r];
  X[r]=matrix[r];
 }
X[0]=matrix[0];
for (int i=1;i<((Q-1)/2)+1;i++)
{
 X[Q-i]=X[i-1];}
  }//end if
if (fl[p][0]>=1.0){ fl[p][0]=1.0;}
if (fl[p][0]<=0.0){ fl[p][0]=0.0;}
//check end of solidification of node 1
if ((fl[p][0]<=0.0)  &&  (fl[p-1][0]>0.0)  &&  (fl[p-1][0]<=1.0))
{
 G=12;
// update node 1 coefficients
h[1][0]=(-2*f*Bi);h[1][1]=(1+2*f*(1+Bi));h[1][2]=-2*f;h[1][Q]=(Tm-L/c*(-fl[p-1][0]));
h[Q-2][Q-1]=h[1][0];h[Q-2][Q-2]=h[1][1];h[Q-2][Q-3]=h[1][2];h[Q-2][Q]=h[1][Q];
X[1]=(Tm-L/c*(-fl[p-1][0]));
for(int i=p;i<3000;i++)
   {fl[i][0]=0.0;}
X[Q-2]=X[1];
 matrix= gauss( h, X,0,1,0.00000000000001,10000000,Q);
pw.println("%end of melrrsolid "+(int)  (p));
for (int i=0;i<((Q-1)/2)+1;i++){
pw.print("      " +(double) (matrix[i]));}
pw.println("       ");
for(int r=1;r<((Q-1)/2)+1;r++)
 {
 R[p][r]=matrix[r];
  X[r]=matrix[r];
 }
```

```
X[0]=matrix[0];
for (int i=1;i<((Q-1)/2)+1;i++)
{
 X[Q-i]=X[i-1];}
  }//end if
for(int q=2;q<(Q-((Q-1)/2))-1;q++)
{
if (R[p][q-1]>Tm){        // check start of melting
if ((R[p][q]>=Tm)  &&   (R[p-1][q]< Tm))
{
   fl[p][q-1]=fl[p-1][q-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][q-1]-2*Tm+R[p][q+1])-(c/L*(Tm-
R[p-1][q]));
 if (fl[p][q-1]<=0.0)
  {fl[p][q-1]=0.0;}
if (fl[p][q-1]>=1.0)
  {fl[p][q-1]=1.0;}
// update coefficients of this node
h[q][q-1]=0.0;h[q][q]=1.0;h[q][Q]=(Tm);h[q][q+1]=0.0;
h[Q-(q+1)][Q-(q+1)]=1.0;h[Q-(q+1)][Q-(q+2)]=0.0;h[Q-(q+1)][Q]=(Tm);h[Q-(q+1)][Q-(q)]=0.0;
X[q]=(Tm);
X[Q-(q+1)]=X[q];
matrix= gauss( h, X,0,1,0.00000000000001,1000000,Q);
pw.println("%correctionTT00011111");
for (int i=0;i<(Q+1)/2;i++){
pw.print("      " +(double) (matrix[i]));}
pw.println("       ");
for(int r=1;r<(Q+1)/2;r++)
 {
 R[p][r]=matrix[r];
  X[r]=matrix[r];
 }
X[0]=matrix[0];
for (int i=1;i<((Q-1)/2)+1;i++)
{
 X[Q-i]=X[i-1];}
for(int j=2;j<(Q-((Q+1)/2));j++)
```

```
{
if(j!=q)     //continue;
{
if (fl[p-1][j-1]!=1.0){
fl[p][j-1]=fl[p-1][j-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][j-1]-2*Tm+R[p][j+1]);
}
if (fl[p][j-1]<=0.0)
  {fl[p][j-1]=0.0;}
if (fl[p][j-1]>=1.0)
  {fl[p][j-1]=1.0;}
}
}
if (fl[p-1][(Q-((Q+1)/2))-1]!=1.0){
fl[p][(Q-((Q+1)/2))-1]=fl[p-1][(Q-((Q+1)/2))-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][(Q-
((Q+1)/2))-1]-2*Tm+R[p][(Q-((Q+1)/2))-1]);
}
if (fl[p][(Q-((Q+1)/2))-1]<=0.0)
  {fl[p][(Q-((Q+1)/2))-1]=0.0;}
if (fl[p][(Q-((Q+1)/2))-1]>=1.0)
  {fl[p][(Q-((Q+1)/2))-1]=1.0;}
}  ///end if
} ///end if new
}   // end  for
///solidification
for(int q=2;q<(Q-((Q-1)/2))-1;q++)
{
if (R[p][q-1]<Tm){        //check start of solidification
if ((R[p][q]<=Tm)  &&   (R[p-1][q]> Tm))
{
   //correction of liquid fraction
fl[p][q-1]=fl[p-1][q-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][q-1]-2*Tm+R[p][q+1])-(c/L*(Tm-
R[p-1][q]));
 if (fl[p][q-1]<=0.0)
  {fl[p][q-1]=0.0;}
if (fl[p][q-1]>=1.0)
  {fl[p][q-1]=1.0;}
```

```
h[q][q-1]=0.0;h[q][q]=1.0;h[q][Q]=(Tm);h[q][q+1]=0.0;
h[Q-(q+1)][Q-(q+1)]=1.0;h[Q-(q+1)][Q-(q+2)]=0.0;h[Q-(q+1)][Q]=(Tm);h[Q-(q+1)][Q-(q)]=0.0;
X[q]=(Tm);
X[Q-(q+1)]=X[q];
matrix= gauss( h, X,0,1,0.00000000000001,1000000,Q);
pw.println("%correctionTT000solid1111");
for (int i=0;i<(Q+1)/2;i++){
pw.print("      " +(double) (matrix[i]));}
pw.println("       ");
for(int r=1;r<(Q+1)/2;r++)
 {
 R[p][r]=matrix[r];
  X[r]=matrix[r];
 }
X[0]=matrix[0];
for (int i=1;i<((Q-1)/2)+1;i++)
{
 X[Q-i]=X[i-1];}
for(int j=2;j<(Q-((Q+1)/2));j++)
{
if(j!=q)     //continue;
{
if (fl[p-1][j-1]!=0.0){
fl[p][j-1]=fl[p-1][j-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][j-1]-2*Tm+R[p][j+1]);
}
if (fl[p][j-1]<=0.0)
  {fl[p][j-1]=0.0;}
if (fl[p][j-1]>=1.0)
  {fl[p][j-1]=1.0;}
}
}
if (fl[p-1][(Q-((Q+1)/2))-1]!=0.0){
fl[p][(Q-((Q+1)/2))-1]=fl[p-1][(Q-((Q+1)/2))-1]+dl*dt/(roh*L*Math.pow(dx,2))*(R[p][(Q-
((Q+1)/2))-1]-2*Tm+R[p][(Q-((Q+1)/2))-1]);
}
if (fl[p][(Q-((Q+1)/2))-1]<=0.0)
```

```
  {fl[p][(Q-((Q+1)/2))-1]=0.0;}
if (fl[p][(Q-((Q+1)/2))-1]>=1.0)
  {fl[p][(Q-((Q+1)/2))-1]=1.0;}
}  ///end if
} ///end if new
}   // end  for
// check start of melting of central  node
if ((R[p][(Q-((Q+1)/2))]>=Tm)  &&   (R[p-1][(Q-((Q+1)/2))]< Tm))
{
if (fl[p-1][(Q-((Q+1)/2))-1]!=1.0){
fl[p][Q-((Q+1)/2)-1]=fl[p-1][Q-((Q+1)/2)-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][Q-((Q+1)/2)]-
2*Tm+R[p][Q-((Q+1)/2)])-(c/L*(Tm-R[p-1][Q-((Q+1)/2)]));
 }
if (fl[p][(Q-((Q+1)/2))-1]<=0.0)
  {fl[p][(Q-((Q+1)/2))-1]=0.0;}
if (fl[p][(Q-((Q+1)/2))-1]>=1.0)
  {fl[p][(Q-((Q+1)/2))-1]=1.0;}
h[Q-((Q+1)/2)][Q-((Q+1)/2)-1]=0.0;h[Q-((Q+1)/2)][Q-((Q+1)/2)]=1.0;h[Q-
((Q+1)/2)][Q]=(Tm);h[Q-((Q+1)/2)][Q-((Q+1)/2)+1]=0.0;
X[Q-((Q+1)/2)]=(Tm);
X[Q-(Q-((Q+1)/2)+1)]=X[Q-((Q+1)/2)];
matrix= gauss( h, X,0,1,0.00000000000001,1000000,Q);
pw.println("%correctionTTcenter");
for (int i=0;i<((Q-1)/2+1);i++){
pw.print("      " +(double) (matrix[i]));}
pw.println("       ");
for(int r=1;r<((Q-1)/2+1);r++)
 {
 R[p][r]=matrix[r];
  X[r]=matrix[r];
 }
X[0]=matrix[0];
for (int i=1;i<((Q-1)/2)+1;i++)
{
 X[Q-i]=X[i-1];}
 } ///end if
```

```
///check start of solidification of central node
if ((R[p][(Q-((Q+1)/2))]<=Tm)  &&   (R[p-1][(Q-((Q+1)/2))]> Tm))
{
if (fl[p-1][(Q-((Q+1)/2))-1]!=0.0){
fl[p][Q-((Q+1)/2)-1]=fl[p-1][Q-((Q+1)/2)-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][Q-((Q+1)/2)]-
2*Tm+R[p][Q-((Q+1)/2)])-(c/L*(Tm-R[p-1][Q-((Q+1)/2)]));
 }
if (fl[p][(Q-((Q+1)/2))-1]<=0.0)
  {fl[p][(Q-((Q+1)/2))-1]=0.0;}
if (fl[p][(Q-((Q+1)/2))-1]>=1.0)
  {fl[p][(Q-((Q+1)/2))-1]=1.0;}
h[Q-((Q+1)/2)][Q-((Q+1)/2)-1]=0.0;h[Q-((Q+1)/2)][Q-((Q+1)/2)]=1.0;h[Q-
((Q+1)/2)][Q]=(Tm);h[Q-((Q+1)/2)][Q-((Q+1)/2)+1]=0.0;
X[Q-((Q+1)/2)]=(Tm);
X[Q-(Q-((Q+1)/2)+1)]=X[Q-((Q+1)/2)];
matrix= gauss( h, X,0,1,0.00000000000001,1000000,Q);
pw.println("%correctionTTcenter");
for (int i=0;i<((Q-1)/2+1);i++){
pw.print("      " +(double) (matrix[i]));}
pw.println("       ");
for(int r=1;r<((Q-1)/2+1);r++)
 {
 R[p][r]=matrix[r];
  X[r]=matrix[r];
 }
X[0]=matrix[0];
for (int i=1;i<((Q-1)/2)+1;i++)
{
 X[Q-i]=X[i-1];}
 } ///end if
for(int u=0;u<(Q-3)/2+1;u++)
{
if (fl[p][u] <= 0.0)
  {
  fl[p][u]=0.0;}
if (fl[p][u] >= 1.0){
```

```
  fl[p][u]=1.0;}
 F[p][u]=fl[p][u];
}
//check  end of melting of internal nodes
for(int  q=2;q<(Q-((Q+3)/2))+1;q++)
{
if ((fl[p][q-1]>=1.0)  &&  (fl[p-1][q-1]< 1.0))
{
GR=q-1;
int aa=q;
System.out.print("uiewrtwe");
h[q][q-1]=-f;h[q][q]=(1+2*f);h[q][Q]=(Tm-L/c*(1-fl[p-1][q-1]));h[q][q+1]=-f;
h[Q-(q+1)][Q-(q+1)]=(1+2*f);h[Q-(q+1)][Q-(q+2)]=-f;h[Q-(q+1)][Q]=(Tm-L/c*(1-fl[p-1][q-
1]));h[Q-(q+1)][Q-(q)]=-f;
X[q]=(Tm-L/c*(1-fl[p-1][q-1]));
//X[q]=(Tm);
X[Q-(q+1)]=X[q];
 matrix= gauss( h, X,0,1,0.00000000000001,1000000,Q);
int yy;
for( yy=p;yy<gt;yy++)
{ fl[yy][q-1]=1.0;}
pw.println("%end of meltingccc "+(int)(q));
for (int i=0;i<((Q-1)/2)+1;i++){
pw.print("      " +(double) (matrix[i]));}
pw.println("       ");
for(int r=1;r<((Q-1)/2)+1;r++)
 {
 R[p][r]=matrix[r];
  X[r]=matrix[r];
 }
X[0]=matrix[0];
for (int i=1;i<((Q-1)/2)+1;i++)
{
 X[Q-i]=X[i-1];}
//check start of melting of other nodes
for(int e=aa+1;e<(Q-((Q-1)/2))-1;e++)
```

```
{
if ((R[p][e]>=Tm)  &&   (R[p-1][e]< Tm))
{
if (fl[p-1][e-1]!=1.0){
fl[p][e-1]=fl[p-1][e-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][e-1]-2*Tm+R[p][e+1])-(c/L*(Tm-
R[p-1][e]));
 }
if (fl[p][e-1]<=0.0)
  {fl[p][e-1]=0.0;}
if (fl[p][e-1]>=1.0)
  {fl[p][e-1]=1.0;}
h[e][e-1]=0.0;h[e][e]=1.0;h[e][Q]=(Tm);h[e][e+1]=0.0;
h[Q-(e+1)][Q-(e+1)]=1.0;h[Q-(e+1)][Q-(e+2)]=0.0;h[Q-(e+1)][Q]=(Tm);h[Q-(e+1)][Q-(e)]=0.0;
X[e]=(Tm);
X[Q-(e+1)]=X[e];
matrix= gauss( h, X,0,1,0.00000000000001,1000000,Q);
pw.println("%correctionTT0002");
for (int i=0;i<(Q+1)/2;i++){
pw.print("      " +(double) (matrix[i]));}
pw.println("       ");
//printX(matrix);
 for(int r=1;r<(Q+1)/2;r++)
 {
 R[p][r]=matrix[r];
  X[r]=matrix[r];
 }
X[0]=matrix[0];
for (int i=1;i<((Q-1)/2)+1;i++)
{
 X[Q-i]=X[i-1];}
for(int j=aa;j<(Q-((Q+1)/2));j++)
{
if(j!=e)     //continue;
{
if (fl[p-1][j-1]!=1.0){
fl[p][j-1]=fl[p-1][j-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][j-1]-2*Tm+R[p][j+1]);
```

```
}
if (fl[p][j-1]<=0.0)
  {fl[p][j-1]=0.0;}
if (fl[p][j-1]>=1.0)
  {fl[p][j-1]=1.0;}
}
}
if (fl[p-1][(Q-((Q+1)/2))-1]!=1.0){
fl[p][(Q-((Q+1)/2))-1]=fl[p-1][(Q-((Q+1)/2))-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][(Q-
((Q+1)/2))-1]-2*Tm+R[p][(Q-((Q+1)/2))-1]);
}
if (fl[p][(Q-((Q+1)/2))-1]<=0.0)
  {fl[p][(Q-((Q+1)/2))-1]=0.0;}
if (fl[p][(Q-((Q+1)/2))-1]>=1.0)
  {fl[p][(Q-((Q+1)/2))-1]=1.0;}
}  ///end if
}   // end  for
}   ///if
}//for
//check end of solidification  of internal nodes
for(int  q=2;q<(Q-((Q+3)/2))+1;q++)
{
if ((fl[p][q-1]<=0.0)  &&  (fl[p-1][q-1]>0.0)  &&  (fl[p-1][q-1]<1.0))
{
GR=q-1;
int aa=q;
pw.println("%flll"+(double)(fl[p-1][q-1]));
System.out.print("uiewrtwe");
h[q][q-1]=-f;h[q][q]=(1+2*f);h[q][Q]=(Tm+L/c*(fl[p-1][q-1]));h[q][q+1]=-f;
h[Q-(q+1)][Q-(q+1)]=(1+2*f);h[Q-(q+1)][Q-(q+2)]=-f;h[Q-(q+1)][Q]=(Tm+L/c*(fl[p-1][q-
1]));h[Q-(q+1)][Q-(q)]=-f;
//X[q]=(Tm+L/cl*(fl[p-1][q-1]));
X[Q-(q+1)]=X[q];
 matrix= gauss( h, X,0,1,0.000000000000001,1000000,Q);
for(int i=p;i<3000;i++)
   {fl[i][q-1]=0.0;}
```

```
pw.println("%end of solidccc "+(double)(q));
for (int i=0;i<((Q-1)/2)+1;i++){
pw.print("      " +(double) (matrix[i]));}
pw.println("       ");
for(int r=1;r<((Q-1)/2)+1;r++)
 {
 R[p][r]=matrix[r];
  X[r]=matrix[r];
 }
X[0]=matrix[0];
for (int i=1;i<((Q-1)/2)+1;i++)
{
 X[Q-i]=X[i-1];}
// check start of melting of other nodes
for(int e=aa+1;e<(Q-((Q-1)/2))-1;e++)
{
if ((R[p][e]<=Tm)  &&   (R[p-1][e]>Tm))
{
if (fl[p-1][e-1]!=0.0){
fl[p][e-1]=fl[p-1][e-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][e-1]-2*Tm+R[p][e+1])-(c/L*(Tm-
R[p-1][e]));
 }
if (fl[p][e-1]<=0.0)
  {fl[p][e-1]=0.0;}
if (fl[p][e-1]>=1.0)
  {fl[p][e-1]=1.0;}
h[e][e-1]=0.0;h[e][e]=1.0;h[e][Q]=(Tm);h[e][e+1]=0.0;
h[Q-(e+1)][Q-(e+1)]=1.0;h[Q-(e+1)][Q-(e+2)]=0.0;h[Q-(e+1)][Q]=(Tm);h[Q-(e+1)][Q-(e)]=0.0;
X[e]=(Tm);
X[Q-(e+1)]=X[e];
matrix= gauss( h, X,0,1,0.00000000000001,1000000,Q);
pw.println("%corrsolid0002");
for (int i=0;i<(Q+1)/2;i++){
pw.print("      " +(double) (matrix[i]));}
pw.println("       ");
for(int r=1;r<(Q+1)/2;r++)
```

```
 {
 R[p][r]=matrix[r];
  X[r]=matrix[r];
 }
X[0]=matrix[0];
for (int i=1;i<((Q-1)/2)+1;i++)
{
 X[Q-i]=X[i-1];}
for(int j=aa;j<(Q-((Q+1)/2));j++)
{
if(j!=e)     //continue;
{
if (fl[p-1][j-1]!=1.0){
fl[p][j-1]=fl[p-1][j-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][j-1]-2*Tm+R[p][j+1]);
}
if (fl[p][j-1]<=0.0)
  {fl[p][j-1]=0.0;}

if (fl[p][j-1]>=1.0)
  {fl[p][j-1]=1.0;}
}
}
if (fl[p-1][(Q-((Q+1)/2))-1]!=0.0){
fl[p][(Q-((Q+1)/2))-1]=fl[p-1][(Q-((Q+1)/2))-1]+dl*dt/(rohl*L*Math.pow(dx,2))*(R[p][(Q-
((Q+1)/2))-1]-2*Tm+R[p][(Q-((Q+1)/2))-1]);
}
if (fl[p][(Q-((Q+1)/2))-1]<=0.0)
  {fl[p][(Q-((Q+1)/2))-1]=0.0;}
if (fl[p][(Q-((Q+1)/2))-1]>=1.0)
  {fl[p][(Q-((Q+1)/2))-1]=1.0;}
}  ///end if
}   // end  for
}   ///if
}//for
if(R[p][(Q-((Q+1)/2)-1)]>Tm)  // check start of melting of central node
{
```

```
if ((R[p][(Q-((Q+1)/2))]>=Tm)  &&   (R[p-1][(Q-((Q+1)/2))]< Tm))
{
if (fl[p-1][(Q-((Q+1)/2))-1]!=1.0){
fl[p][Q-((Q+1)/2)-1]=fl[p-1][Q-((Q+1)/2)-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][Q-((Q+1)/2)]-
2*Tm+R[p][Q-((Q+1)/2)])-(c/L*(Tm-R[p-1][Q-((Q+1)/2)]));
 }
if (fl[p][(Q-((Q+1)/2))-1]<=0.0)
  {fl[p][(Q-((Q+1)/2))-1]=0.0;}
if (fl[p][(Q-((Q+1)/2))-1]>=1.0)
  {fl[p][(Q-((Q+1)/2))-1]=1.0;}
h[Q-((Q+1)/2)][Q-((Q+1)/2)-1]=0.0;h[Q-((Q+1)/2)][Q-((Q+1)/2)]=1.0;h[Q-
((Q+1)/2)][Q]=(Tm);h[Q-((Q+1)/2)][Q-((Q+1)/2)+1]=0.0;
X[Q-((Q+1)/2)]=(Tm);
X[Q-(Q-((Q+1)/2)+1)]=X[Q-((Q+1)/2)];
matrix= gauss( h, X,0,1,0.00000000000001,1000000,Q);
pw.println("%correctionTTcenter");
for (int i=0;i<((Q-1)/2+1);i++){
pw.print("      " +(double) (matrix[i]));}
pw.println("       ");
for(int r=1;r<((Q-1)/2+1);r++)
 {
 R[p][r]=matrix[r];
  X[r]=matrix[r];
 }
X[0]=matrix[0];
for (int i=1;i<((Q-1)/2)+1;i++)
{
 X[Q-i]=X[i-1];}
 } ///end if
}
//solid
if(R[p][(Q-((Q+1)/2)-1)]<Tm) // check start of solidification of central node.
{
if ((R[p][(Q-((Q+1)/2))]<=Tm)  &&   (R[p-1][(Q-((Q+1)/2))]> Tm))
{
if (fl[p-1][(Q-((Q+1)/2))-1]!=0.0){
```

```
fl[p][Q-((Q+1)/2)-1]=fl[p-1][Q-((Q+1)/2)-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][Q-((Q+1)/2)]-
2*Tm+R[p][Q-((Q+1)/2)])-(c/L*(Tm-R[p-1][Q-((Q+1)/2)]));
 }
if (fl[p][(Q-((Q+1)/2))-1]<=0.0)
  {fl[p][(Q-((Q+1)/2))-1]=0.0;}
if (fl[p][(Q-((Q+1)/2))-1]>=1.0)
  {fl[p][(Q-((Q+1)/2))-1]=1.0;}
h[Q-((Q+1)/2)][Q-((Q+1)/2)-1]=0.0;h[Q-((Q+1)/2)][Q-((Q+1)/2)]=1.0;h[Q-
((Q+1)/2)][Q]=(Tm);h[Q-((Q+1)/2)][Q-((Q+1)/2)+1]=0.0;
X[Q-((Q+1)/2)]=(Tm);
X[Q-(Q-((Q+1)/2)+1)]=X[Q-((Q+1)/2)];
matrix= gauss( h, X,0,1,0.00000000000001,1000000,Q);
pw.println("%corresolidcenter");
for (int i=0;i<((Q-1)/2+1);i++){
pw.print("      " +(double) (matrix[i]));}
pw.println("       ");
for(int r=1;r<((Q-1)/2+1);r++)
 {
 R[p][r]=matrix[r];
  X[r]=matrix[r];
 }
X[0]=matrix[0];
for (int i=1;i<((Q-1)/2)+1;i++)
{
 X[Q-i]=X[i-1];}
 } ///end if
}
//check end of melting of central node
if ((fl[p][(Q-((Q+1)/2))-1]>=1.0)  &&  (fl[p-1][(Q-((Q+1)/2))-1]<1.0))
 {
h[(Q-((Q+1)/2))][(Q-((Q+1)/2))]=(1+2*f);h[(Q-((Q+1)/2))][(Q-((Q+1)/2))-1]=-f;h[(Q-
((Q+1)/2))][Q]=(Tm-L/c*(1-fl[p-1][(Q-((Q+1)/2))-1]));h[(Q-((Q+1)/2))][(Q-((Q+1)/2))+1]=-f;
X[(Q-((Q+1)/2))]=(Tm-L/c*(1-fl[p-1][(Q-((Q+1)/2))-1]));  //////****take  care ****
matrix= gauss( h, X,0,1,0.00000000000001,1000000,Q);
  pw.println("%end of melting "+(double)(fl[p-1][(Q-((Q+1)/2))-1]));
for (int i=0;i<((Q-1)/2)+1;i++){
```

```
pw.print("      " +(double) (matrix[i]));}
pw.println("       ");
for(int r=1;r<((Q-1)/2)+1;r++)
 {
 R[p][r]=matrix[r];
  X[r]=matrix[r];
 }
X[0]=matrix[0];
for (int i=1;i<((Q-1)/2)+1;i++)
{
 X[Q-i]=X[i-1];}
for(int k=2;k<(Q-2);k++){
    h[k][k]=(1+2*fliq);
    h[k][k+1]=-fliq;
    h[k][k-1]=h[k][k+1];
    }
 }//if
//check end of solidification of center node.
if ((fl[p][(Q-((Q+1)/2))-1]<=0.0)  &&  (fl[p-1][(Q-((Q+1)/2))-1]>0.0)  &&  (fl[p-1][(Q-((Q+1)/2))-
1]<1.0))
 {
h[(Q-((Q+1)/2))][(Q-((Q+1)/2))]=(1+2*f);h[(Q-((Q+1)/2))][(Q-((Q+1)/2))-1]=-f;h[(Q-
((Q+1)/2))][Q]=(Tm-L/c*(-fl[p-1][(Q-((Q+1)/2))-1]));h[(Q-((Q+1)/2))][(Q-((Q+1)/2))+1]=-f;
X[(Q-((Q+1)/2))]=(Tm-L/cl*(-fl[p-1][(Q-((Q+1)/2))-1]));
matrix= gauss( h, X,0,1,0.00000000000001,1000000,Q);
  pw.println("%end of melting "+(double)(fl[p-1][(Q-((Q+1)/2))-1]));
for (int i=0;i<((Q-1)/2)+1;i++){
pw.print("      " +(double) (matrix[i]));}
pw.println("       ");
 for(int r=1;r<((Q-1)/2)+1;r++)
 {
 R[p][r]=matrix[r];
  X[r]=matrix[r];
 }
X[0]=matrix[0];
for (int i=1;i<((Q-1)/2)+1;i++)
```

```
{
 X[Q-i]=X[i-1];}
for(int i=p;i<3000;i++)
   {fl[i][(Q-((Q+1)/2))-1]=0.0;}
 h[1][1]=(1+2*f*(1+B)); h[Q-2][Q-2]=h[1][1];
h[1][2]=-2*f;
h[Q-2][Q-3]=h[1][2];
h[1][0]=-2*f*B;
h[Q-2][Q-1]=h[1][0];
for(int k=2;k<(Q-2);k++){
    h[k][k]=(1+2*f);
    h[k][k+1]=-f;
    h[k][k-1]=h[k][k+1];
    }
 }//if
for(int u=0;u<(Q-3)/2+1;u++)
{
if (fl[p][u] <= 0.0)
  {
  fl[p][u]=0.0;}
if (fl[p][u] >= 1.0){
  fl[p][u]=1.0;}
 F[p][u]=fl[p][u];
}
h[1][Q]=R[p][1];
 h[Q-2][Q]=h[1][Q];
for(int i=3;i<((Q-1)/2+1);i++)
{
  h[i-1][Q]=R[p][i-1];
 h[Q-i][Q]=h[i-1][Q];}
h[(Q-((Q+1)/2))][Q]=R[p][(Q-((Q+1)/2))];
}
pw.println("];");
pw.print("plot(A(:,1))");
if (pw!=null) pw.close();
} //end main**********************
```

```
public static double[]  gauss(double A[][],double[] X,int FALSE,int TRUE,double TOL,int NN,int
N)
{
int k=1;
double b[]=new double[N];
int OK=0 ;
double ERR;
while (( OK == FALSE)  &&   (k<= NN ))
{
  ERR = 0.0;
for (int I = 0 ;I< N;I++)
 {
 double S = 0.0;
 for (int J = 0 ;J< N;J++)
 {
 S = S-A[I][J]*X[J];
 }
 S = (S+A[I][N])/(A[I][I]);
 if (Math.abs(S) > ERR)
 {
  ERR  = Math.abs(S);
 }
 X[I] = X[I] + S;
 }
//STEP 4
 // process is complete
  if (ERR > TOL)
 {
// STEP 5
  OK=FALSE;
  k=k+1;
//STEP 6 - is not used since only one vector is required
 }
 else if (ERR <= TOL)  break;
}
for (int o=0;o<N;o++)
```

```
{
b[o]=X[o];
}
 return (double[])( b);
}   //end METHOD  GAUSS ITERATION
public static void printX(double[] m,int d)
{
for(int i=0;i<d;i++)
  System.out.print("   "+m[i]);
}
public static void prinXx(double[][] m,int d)
{
for(int i=0;i<d;i++)
{
for(int j=0;j<(d+1);j++)
System.out.print("   "+m[i][j]);
}
System.out.println("");
}
//file reader  **************************
public static double[] find(double g ,String  fil,int x )
{
FileReader frs =null;
StreamTokenizer in=null;
String sname="slam";
double [] r=new double[x];
double mid=0.0;
try{
frs=new FileReader(fil);
in=new StreamTokenizer(frs);
while(in.ttype!=StreamTokenizer.TT_EOF)
{
if (in.ttype==StreamTokenizer.TT_WORD)
 sname=in.sval;
 else
while (in.nextToken()==StreamTokenizer.TT_NUMBER)
```

```
{
mid=in.nval;
 if (mid==g){
for (int i=0;i<x;i++)
{
if (in.nextToken()==StreamTokenizer.TT_NUMBER)
mid=in.nval;
r[i]=mid;
}
 }
}
in.nextToken();}
}
catch(FileNotFoundException ex) {
System.out.println("receved" );
}
catch(IOException ex)
{
System.out.println(ex.getMessage());
}
finally
{
try
{
if (frs!=null) frs.close();
}
catch(IOException ex)
{
System.out.println(ex);
}
}
return r;
}
};
///*************************************************************************
```

A.4 Constant temperature at the boundary condition source code:

```
import java.io.*;
public class  Melt
{
private static   int t;
private static   int Q;
private  static   int l;
private static   double[] matrix=new double[Q];
private static   int q;
public static void main(String[]  args)
{
PrintWriter pw=null;
try
{
pw=new PrintWriter(new FileOutputStream("d:/resaleh/Meltr.dat"));
}
catch (IOException ex){}
pw.println("  " );
double TTT=0.0;;
int OK;
Q=9;
double[] Pm=new double[5];
Pm=find(3.0 ,"d:/resaleh/pcm.dat" ,5);
for (int tm=0;tm<5;tm++){
pw.print("%  " +(double) (Pm[tm]));
}
double Ts=60.0;  // wall tempereatures
double Tm=Pm[0]; // pcm melting temperature
double dt=60.0;     // time step
double  dx=0.01;   //space between nodes
double d=Pm[4]; // pcm conductivity
double c=1000.0*Pm[3]; // pcm specific heat
double roh=Pm[1];        // pcm density
double  alpa;
alpa=d/(roh*c);          // thermal diffusivity
pw.print(" ccc " +(double) (c));
```

```
double f;f=alpa*dt/(Math.pow(dx,2));
double  L=1000.0*Pm[2];         // pcm latent heat
double[] t=new double[4];
int pl=1660;
pw.println("  ");
pw.println("A=[");
double z=10.0;
//double Fa=4*d*dt/(roh*L*Math.pow(dx,2));
double[][] h=new double[Q][Q+1];
h[0][Q]=(Ts);
h[Q-1][Q]=h[0][Q];h[0][0]=1.0;
h[Q-1][Q-1]=h[0][0];
//  the coeffecients of nodes
for(int k=1;k<(Q-2)+1;k++){
    h[k][k]=(1+2*f);
    h[k][k+1]=-f;
    h[k][k-1]=h[k][k+1];
    h[k][Q]=z;}
double[] X=new double[Q];
for(int i=0;i<Q;i++)
   {X[i]=z;}
double fl[][]=new double[pl][((Q-3)/2)+1];
 double F[][]=new double[pl][((Q-3)/2)+1];
 for(int b=0;b<pl;b++)
 {
for(int a=0;a<((Q-3)/2)+1;a++)     //liquid fraction
 {
 F[b][a]=0;
fl[b][a]=0;
 }
}
double R[][]=new double[pl][((Q-1)/2)+2];
for (int i=0;i<pl;i++)
{                                       // initial conditions
for (int j=0;j<((Q-1)/2)+2;j++)
{
```

```
R[i][j]=z;
}
}
for (int i=0;i<pl;i++)
{
R[i][0]=Ts;}                          //boundary conditions
double TNT=12.0;
pw.println("  ");
//gauss(double A[][],double[] X,int FALSE,int TRUE,double TOL,int k,int NN,int N)
 for (int p=1 ; p<200 ; p++)
{
for(int j=1;j<(Q-((Q+3)/2))+1;j++)
{
// update liquid fraction
fl[p][j-1]=fl[p-1][j-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][j-1]-2*Tm+R[p][j+1]);
}
// liquid fraction of  central node.
fl[p][(Q-((Q+1)/2))-1]=fl[p-1][(Q-((Q+1)/2))-]+
d*dt/(roh*L*Math.pow(dx,2))*(R[p][(Q-((Q+1)/2))-1]-2*Tm+R[p][(Q-((Q+1)/2))-1]);
for(int u=0;u<(Q-3)/2+1;u++)
{
if (fl[p][u] <= 0.0)
  {
  fl[p][u]=0.0;}
if (fl[p][u] >= 1.0){
  fl[p][u]=1.0;}
 F[p][u]=fl[p][u];
}
 // gauss method is gauss sidel iterative procedure
// which solves the system of liniar  equations
// to find the temperature at pcm nodes
matrix= gauss( h, X,0,1,0.0000000000001,1000000,Q);
for(int r=1;r<((Q-1)/2)+1;r++)
 {
 R[p][r]=matrix[r];
  X[r]=matrix[r];
```

```
 }
X[0]=matrix[0];
for (int i=1;i<((Q-1)/2)+1;i++)
{
 X[Q-i]=X[i-1];}*/
 if(p==1){
for (int i=0;i<((Q-1)/2)+1;i++){
pw.print("  " +(double) (matrix[i]));}
pw.println("    ;   ");}
for (int i=0;i<((Q-1)/2)+1;i++){
pw.print("  " +(double) (matrix[i]));}
pw.println("    ;   ");
for(int  k=1;k<(Q-((Q+3)/2))+1;k++)
{
if(R[p][k]==Tm  &&  fl[p-1][k-1]<1.0)   // update liquid fractions  of nodes that
                                                  // are  start of meltting
{
fl[p][k-1]=fl[p-1][k-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][k-1]-2*Tm+R[p][k+1]);
if(fl[p][k-1]<=0.0)
{fl[p][k-1]=0.0;}
if(fl[p][k-1]>=1.0)
{fl[p][k-1]=1.0;}
} ///end  if
} //end for
if(R[p][(Q-((Q+3)/2))+1]==Tm  &&  fl[p-1][(Q-((Q+3)/2))]<1.0)
{
fl[p][Q-((Q+3)/2)]=fl[p-1][Q-((Q+3)/2)]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][Q-((Q+3)/2)]-
2*Tm+R[p][Q-((Q+3)/2)]);
if(fl[p][Q-((Q+3)/2)]<=0.0)
{fl[p][Q-((Q+3)/2)]=0.0;}
if(fl[p][Q-((Q+3)/2)]>=1.0)
{fl[p][Q-((Q+3)/2)]=1.0;}
}
for(int q=1;q<(Q-((Q-1)/2))-1;q++)
{
if ((R[p][q]>=Tm)  &&   (R[p-1][q]< Tm))  // check the start of melting
```

```
{
if (fl[p][q-1]!=1.0){    /////new *****************
fl[p][q-1]=fl[p-1][q-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][q-1]-2*Tm+R[p][q+1])-(c/L*(Tm-
R[p-1][q]));
 }
if (fl[p][q-1]<=0.0)
  {fl[p][q-1]=0.0;}
if (fl[p][q-1]>=1.0)
  {fl[p][q-1]=1.0;}
h[q][q-1]=0.0;h[q][q]=1.0;h[q][Q]=(Tm);h[q][q+1]=0.0;
h[Q-(q+1)][Q-(q+1)]=1.0;h[Q-(q+1)][Q-(q+2)]=0.0;h[Q-(q+1)][Q]=(Tm);h[Q-(q+1)][Q-(q)]=0.0;
X[q]=(Tm);
X[Q-(q+1)]=X[q];
matrix= gauss( h, X,0,1,0.00000000000001,1000000,Q);
pw.println("%correctionTT000");
for (int i=0;i<(Q+1)/2;i++){
pw.print("      " +(double) (matrix[i]));}
pw.println("       ");
for(int r=1;r<(Q+1)/2;r++)
 {
 R[p][r]=matrix[r];
  X[r]=matrix[r];
 }
X[0]=matrix[0];
for (int i=1;i<((Q-1)/2)+1;i++)
{
 X[Q-i]=X[i-1];}
for(int j=2;j<(Q-((Q+1)/2));j++)
{
if(j!=q)     //continue;
{
if (fl[p][j-1]!=1.0){
fl[p][j-1]=fl[p-1][j-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][j-1]-2*Tm+R[p][j+1]);
}
if (fl[p][j-1]<=0.0)
  {fl[p][j-1]=0.0;}
```

```
if (fl[p][j-1]>=1.0)
  {fl[p][j-1]=1.0;}
}
}
if (fl[p][(Q-((Q+1)/2))-1]!=1.0){
fl[p][(Q-((Q+1)/2))-1]=fl[p-1][(Q-((Q+1)/2))-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][(Q-
((Q+1)/2))-1]-2*Tm+R[p][(Q-((Q+1)/2))-1]);
}
if (fl[p][(Q-((Q+1)/2))-1]<=0.0)
  {fl[p][(Q-((Q+1)/2))-1]=0.0;}
if (fl[p][(Q-((Q+1)/2))-1]>=1.0)
  {fl[p][(Q-((Q+1)/2))-1]=1.0;}
}  ///end if
}   // end  for
if ((R[p][(Q-((Q+1)/2))]>=Tm)  &&   (R[p-1][(Q-((Q+1)/2))]< Tm)) // center node check of
                                                                  // start of melting
{
if (fl[p][(Q-((Q+1)/2))-1]!=1.0){
fl[p][Q-((Q+1)/2)-1]=fl[p-1][Q-((Q+1)/2)-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][Q-((Q+1)/2)]-
2*Tm+R[p][Q-((Q+1)/2)])-(c/L*(Tm-R[p-1][Q-((Q+1)/2)]));
 }
if (fl[p][(Q-((Q+1)/2))-1]<=0.0)
  {fl[p][(Q-((Q+1)/2))-1]=0.0;}
if (fl[p][(Q-((Q+1)/2))-1]>=1.0)
  {fl[p][(Q-((Q+1)/2))-1]=1.0;}
h[Q-((Q+1)/2)][Q-((Q+1)/2)-1]=0.0;h[Q-((Q+1)/2)][Q-((Q+1)/2)]=1.0;h[Q-
((Q+1)/2)][Q]=(Tm);h[Q-((Q+1)/2)][Q-((Q+1)/2)+1]=0.0;
X[Q-((Q+1)/2)]=(Tm);
X[Q-(Q-((Q+1)/2)+1)]=X[Q-((Q+1)/2)];
matrix= gauss( h, X,0,1,0.00000000000001,1000000,Q);
pw.println("%correctionTT000");
for (int i=0;i<((Q-1)/2+1);i++){
pw.print("      " +(double) (matrix[i]));}
pw.println("       ");
for(int r=1;r<((Q-1)/2+1);r++)
 {
```

```
 R[p][r]=matrix[r];
  X[r]=matrix[r];
 }
X[0]=matrix[0];
for (int i=1;i<((Q-1)/2)+1;i++)
{
 X[Q-i]=X[i-1];}
 } ///end if
for(int u=0;u<(Q-3)/2+1;u++)
{
if (fl[p][u] <= 0.0)
  {
  fl[p][u]=0.0;}
if (fl[p][u] >= 1.0){
  fl[p][u]=1.0;}
 F[p][u]=fl[p][u];
}
for(int  q=1;q<(Q-((Q+3)/2))+1;q++)  // check end of melting
{
if ((fl[p][q-1]>=1.0)  &&  (fl[p-1][q-1]< 1.0))
{
int aa=q;
System.out.print("uiewrtwe");
h[q][q-1]=-f;h[q][q]=(1+2*f);h[q][Q]=(Tm-L/c*(1-fl[p-1][q-1]));h[q][q+1]=-f;
h[Q-(q+1)][Q-(q+1)]=(1+2*f);h[Q-(q+1)][Q-(q+2)]=-f;h[Q-(q+1)][Q]=(Tm-L/c*(1-fl[p-1][q-
1]));h[Q-(q+1)][Q-(q)]=-f;
X[q]=(Tm-L/c*(1-fl[p-1][q-1]));
X[Q-(q+1)]=X[q];
 matrix= gauss( h, X,0,1,0.00000000000001,1000000,Q);
pw.println("%end of meltingccc "+(double)(q));
for (int i=0;i<5;i++){
pw.print("      " +(double) (matrix[i]));}
pw.println("       ");
for(int r=1;r<(Q-4);r++)
 {
 R[p][r]=matrix[r];
```

```
  X[r]=matrix[r];
 }
X[0]=matrix[0];
for (int i=1;i<((Q-1)/2)+1;i++)
{
 X[Q-i]=X[i-1];}
 for(int e=aa+1;e<(Q-((Q-1)/2));e++)
{
if ((R[p][e]>=Tm)  &&   (R[p-1][e]< Tm))  // check start of melting
{
R[p][5]=R[p][3];
if (fl[p][e-1]!=1.0){    /////new *****************
fl[p][e-1]=fl[p-1][e-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][e-1]-2*Tm+R[p][e+1])-(c/L*(Tm-
R[p-1][e]));
 }
if (fl[p][e-1]<=0.0)
  {fl[p][e-1]=0.0;}
if (fl[p][e-1]>=1.0)
  {fl[p][e-1]=1.0;}
h[e][e-1]=0.0;h[e][e]=1.0;h[e][Q]=(Tm);h[e][e+1]=0.0;
h[Q-(e+1)][Q-(e+1)]=1.0;h[Q-(e+1)][Q-(e+2)]=0.0;h[Q-(e+1)][Q]=(Tm);h[Q-(e+1)][Q-(e)]=0.0;
X[e]=(Tm);
X[Q-(e+1)]=X[e];
matrix= gauss( h, X,0,1,0.00000000000001,1000000,Q);
pw.println("%correctionTT000");
for (int i=0;i<(Q+1)/2;i++){
pw.print("      " +(double) (matrix[i]));}
pw.println("       ");
for(int r=1;r<(Q+1)/2;r++)
 {
 R[p][r]=matrix[r];
  X[r]=matrix[r];
 }
X[0]=matrix[0];
for (int i=1;i<((Q-1)/2)+1;i++)
{
```

```
 X[Q-i]=X[i-1];}
for(int j=2;j<(Q-((Q+1)/2));j++)
{
if(j!=e)     //continue;
{
if (fl[p][j-1]!=1.0){
fl[p][j-1]=fl[p-1][j-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][j-1]-2*Tm+R[p][j+1]);
}
if (fl[p][j-1]<=0.0)
  {fl[p][j-1]=0.0;}
if (fl[p][j-1]>=1.0)
  {fl[p][j-1]=1.0;}
}
}
if (fl[p][(Q-((Q+1)/2))-1]!=1.0){
fl[p][(Q-((Q+1)/2))-1]=fl[p-1][(Q-((Q+1)/2))-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][(Q-
((Q+1)/2))-1]-2*Tm+R[p][(Q-((Q+1)/2))-1]);
}
if (fl[p][(Q-((Q+1)/2))-1]<=0.0)
  {fl[p][(Q-((Q+1)/2))-1]=0.0;}
if (fl[p][(Q-((Q+1)/2))-1]>=1.0)
  {fl[p][(Q-((Q+1)/2))-1]=1.0;}
}  ///end if
}   // end  for
}   ///if
}//for
//check  end of melting of central node.
 if ((fl[p][(Q-((Q+1)/2))-1]>=1.0)  &&  (fl[p-1][(Q-((Q+1)/2))-1]<1.0))
 {
h[(Q-((Q+1)/2))][(Q-((Q+1)/2))]=(1+2*f);h[(Q-((Q+1)/2))][(Q-((Q+1)/2))-1]=-f;h[(Q-
((Q+1)/2))][Q]=(Tm-L/c*(1-fl[p-1][(Q-((Q+1)/2))-1]));h[(Q-((Q+1)/2))][(Q-((Q+1)/2))+1]=-f;
pw.println("%fl "+(double) (fl[p-1][(Q-((Q+1)/2))-1]));
X[(Q-((Q+1)/2))]=(Tm-L/c*(1-fl[p-1][(Q-((Q+1)/2))-1]));  //////****take  care ****
matrix= gauss( h, X,0,1,0.00000000000001,1000000,Q);
  pw.println("%end of meltingeexxxx "+(double)((Q-((Q+1)/2))));
for (int i=0;i<(Q-4);i++){
```

```
pw.print("      " +(double) (matrix[i]));}
pw.println("       ");
 for(int r=1;r<(Q-4);r++)
 {
 R[p][r]=matrix[r];
  X[r]=matrix[r];
 }
X[0]=matrix[0];
for (int i=1;i<((Q-1)/2)+1;i++)
{
 X[Q-i]=X[i-1];}
 }//if
for(int u=0;u<(Q-3)/2+1;u++)
{
if (fl[p][u] <= 0.0)
  {
  fl[p][u]=0.0;}
if (fl[p][u] >= 1.0){
  fl[p][u]=1.0;}
 F[p][u]=fl[p][u];
}
for(int i=1;i<((Q-1)/2+1);i++)
{
 h[i-1][Q]=R[p][i-1];
 h[Q-i][Q]=h[i-1][Q];}
 h[(Q-((Q+1)/2))][Q]=R[p][(Q-((Q+1)/2))];
 }
}
pw.println("];");
pw.print("plot(A(:,1))");
if (pw!=null) pw.close();
} //end main
//methods
public static double[]  gauss(double A[][],double[] X,int FALSE,int TRUE,double TOL,int NN,int
N)
{
```

```
int k=1;
double b[]=new double[N];
int OK=0 ;
double ERR;
while (( OK == FALSE)  &&   (k<= NN ))
{
 ERR = 0.0;
for (int I = 0 ;I< N;I++)
 {
 double S = 0.0;
 for (int J = 0 ;J< N;J++)
 {
 S = S-A[I][J]*X[J];
 }
 S = (S+A[I][N])/(A[I][I]);
 if (Math.abs(S) > ERR)
 {
  ERR  = Math.abs(S);
 }
 X[I] = X[I] + S;
 }
//STEP 4
 // process is complete
  if (ERR > TOL)
 {
// STEP 5
  OK=FALSE;
  k=k+1;
//STEP 6 - is not used since only one vector is required
 }
 else if (ERR <= TOL)  break;
 }
for (int o=0;o<N;o++)
{
b[o]=X[o];
}
```

```
 return (double[])( b);
}   //end METHOD  GAUSS ITERATION
public static void printX(double[] m,int d)
{
for(int i=0;i<d;i++)
  System.out.print("   "+m[i]);
}
public static void prinXx(double[][] m,int d)
{
for(int i=0;i<d;i++)
{
for(int j=0;j<(d+1);j++)
System.out.print("   "+m[i][j]);
}
System.out.println("");
}
//file reader  **************************
public static double[] find(double g ,String  fil,int x )
{
FileReader frs =null;
StreamTokenizer in=null;
String sname="slam";
double [] r=new double[x];
double mid=0.0;
try{
frs=new FileReader(fil);
in=new StreamTokenizer(frs);
while(in.ttype!=StreamTokenizer.TT_EOF)
{
if (in.ttype==StreamTokenizer.TT_WORD)
 sname=in.sval;
 else
while (in.nextToken()==StreamTokenizer.TT_NUMBER)
{
mid=in.nval;
 if (mid==g){
```

```
for (int i=0;i<x;i++)
{
if (in.nextToken()==StreamTokenizer.TT_NUMBER)
mid=in.nval;
r[i]=mid;
}
 }
}
in.nextToken();}
}
catch(FileNotFoundException ex) {
System.out.println("receved" );
}
catch(IOException ex)
{
System.out.println(ex.getMessage());
}
finally
{
try
{
if (frs!=null) frs.close();
}
catch(IOException ex)
{
System.out.println(ex);
}
}
return r;
}
};
```

A.5 constant convectional heat transfer at end walls with different solid and liquid thermal properties source code:

```
import java.io.*;
public class  ConvCONN
{
```

```
private static   int t;
private static   int Q;
private  static   int l;
private static   double[] matrix=new double[Q];
private static   int q;
public static void main(String[]  args)
{
PrintWriter pw=null;
try
{
pw=new PrintWriter(new FileOutputStream("d:/resaleh/convconstant.dat"));
}
catch (IOException ex){}
pw.println("  " );
double TTT=0.0;;
int OK;
Q=13;
double[] Pm=new double[5];
Pm=find(3.0 ,"d:/resaleh/pcm.dat" ,5);
for (int tm=0;tm<5;tm++){
pw.print("%  " +(double) (Pm[tm]));
}
double Tm=Pm[0];
double dt=5.0;
double  dx=0.002;
double d=Pm[4];
double c=1000.0*Pm[3];
double roh=Pm[1];
double  alpa;
alpa=d/(roh*c); //diffusivity for solid phase
pw.print(" ccc " +(double) (c));
double f;f=alpa*dt/(Math.pow(dx,2));
double  L=1000.0*Pm[2];
double[] t=new double[4];
int pl=1000;
double hc=5.0;
```

```
double upcm=16.0;double Apcm=0.05;
double fg=upcm*dx/d;
double Ti=20.0;double roha=1.5;double ca=1044.0;
double  alpal;
alpal=.53/(1530.0*2200.0);   // diffusivity for liquid phase
pw.print(" ccc " +(double) (c));
double fliq;fliq=alpal*dt/(Math.pow(dx,2));
double Bl=upcm*dx/0.53;    //Biot number for liquid phase
pw.println("A=[");
double Bi=upcm*dx/d;        //Biot number for solid phase
double z=17.0;
double Fa=d*dt/(roh*L*Math.pow(dx,2));
double DT=((1/(2*alpa*(1+Bi)))*Math.pow(dx,2));
pw.println("%f(1+Bi)"+(double)(DT));
int G=0;
int GR=0;
double[][] h=new double[Q][Q+1];
h[0][Q]=60.0;    // coefficients of node 1 and 2  at solid phase
h[Q-1][Q]=h[0][Q];h[0][0]=1.0;
h[Q-1][Q-1]=h[0][0];
h[0][1]=0.0;h[Q-1][Q-2]=h[0][1];
h[1][Q]=(z);h[Q-2][Q]=h[1][Q];
h[1][1]=(1+2*f*(1+Bi)); h[Q-2][Q-2]=h[1][1];
h[1][2]=-2*f;
h[Q-2][Q-3]=h[1][2];
h[1][0]=-2*f*Bi;
h[Q-2][Q-1]=h[1][0];
for(int k=2;k<(Q-2);k++){  // coefficients of other nodes at solid phase
   h[k][k]=(1+2*f);
   h[k][k+1]=-f;
   h[k][k-1]=h[k][k+1];
   h[k][Q]=z;}
double[] X=new double[Q];
for(int i=0;i<Q;i++)
  {X[i]=z;}
pl=2000;
```

```
double fl[][]=new double[pl][((Q-3)/2)+1];
 double F[][]=new double[pl][((Q-3)/2)+1];
 for(int b=0;b<pl;b++)
 {
for(int a=0;a<((Q-3)/2)+1;a++)
 {
 F[b][a]=0.0;
fl[b][a]=0.0;   //liquid fraction at solid phase
 }
}
pl=2000;
double R[][]=new double[pl][((Q-1)/2)+1];
for (int i=0;i<pl;i++)
{
for (int j=0;j<((Q-1)/2)+1;j++)
{
R[i][j]=z;         // initial condition
}
}
double TNT=12.0;
pw.println("  ");
//gauss(double A[][],double[] X,int FALSE,int TRUE,double TOL,int k,int NN,int N)
 for (int p=1 ; p<1050 ; p++)
{
matrix= gauss( h, X,0,1,0.00000000000001,1000000,Q);
for(int r=1;r<((Q-1)/2)+1;r++)
 {
 R[p][r]=matrix[r];
  X[r]=matrix[r];
 }
X[0]=matrix[0];
for (int i=1;i<((Q-1)/2)+1;i++)
{
 X[Q-i]=X[i-1];}
for (int i=0;i<((Q-1)/2)+1;i++){
pw.print("  " +(double) (matrix[i]));}
```

```
pw.println("    ;   ");
if(R[p][1]==Tm  &&  fl[p-1][0]<1.0)
{
fl[p][0]=fl[p-1][0]+2*Bi*Fa*matrix[0]-2*Fa*(1+Bi)*Tm+2*Fa*R[p][2];
if(fl[p][0]<=0.0)
{fl[p][0]=0.0;}
if(fl[p][0]>=1.0)
{fl[p][0]=1.0;}}
for(int  k=2;k<(Q-((Q+3)/2))+1;k++)
{
if(R[p][k]==Tm  &&    fl[p-1][k-1]<1.0)
{                          //update liquid fraction after node i start of melting
fl[p][k-1]=fl[p-1][k-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][k-1]-2*Tm+R[p][k+1]);
if(fl[p][k-1]<=0.0)
{fl[p][k-1]=0.0;}
if(fl[p][k-1]>=1.0)
{fl[p][k-1]=1.0;}
} ///end  if
} //end for
if(R[p][(Q-((Q+3)/2))+1]==Tm  &&  fl[p-1][(Q-((Q+3)/2))]<1.0)
{
fl[p][Q-((Q+3)/2)]=fl[p-1][Q-((Q+3)/2)]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][Q-((Q+3)/2)]-
2*Tm+R[p][Q-((Q+3)/2)]);
if(fl[p][Q-((Q+3)/2)]<=0.0)
{fl[p][Q-((Q+3)/2)]=0.0;}
if(fl[p][Q-((Q+3)/2)]>=1.0)
{fl[p][Q-((Q+3)/2)]=1.0;}
}
if ((R[p][1]>=Tm)  &&   (R[p-1][1]< Tm)) //check start of melting of node 1
{
//correction  of liquid fraction
fl[p][0]=fl[p-1][0]+2*Bi*Fa*matrix[0]-2*Fa*(1+Bi)*Tm+2*Fa*R[p][2]-c/L*(Tm-R[p-1][1]);
if (fl[p][0]<=0.0)
  {fl[p][0]=0.0;}
if (fl[p][0]>=1.0)
   {fl[p][0]=1.0;}
```

```
  // update coefficient of node 1
h[1][0]=0.0;h[1][1]=1.0;h[1][2]=0.0;h[1][Q]=Tm;
h[Q-2][Q-1]=h[1][0];h[Q-2][Q-2]=h[1][1];h[Q-2][Q-3]=h[1][2];h[Q-2][Q]=h[1][Q];
//calculate  temperature at the same time step
matrix= gauss( h, X,0,1,0.00000000000001,1000000,Q);
pw.println("%correctionTT000");
for (int i=0;i<(Q+1)/2;i++){
pw.print("      " +(double) (matrix[i]));}
pw.println("       ");
for(int r=1;r<(Q+1)/2;r++)
 {
 R[p][r]=matrix[r];
  X[r]=matrix[r];
 }
X[0]=matrix[0];
for (int i=1;i<((Q-1)/2)+1;i++)
{
 X[Q-i]=X[i-1];}
for(int j=2;j<(Q-((Q+1)/2));j++)
{
if (fl[p-1][j-1]!=1.0){
fl[p][j-1]=fl[p-1][j-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][j-1]-2*Tm+R[p][j+1]);
}
if (fl[p][j-1]<=0.0)
  {fl[p][j-1]=0.0;}
if (fl[p][j-1]>=1.0)
  {fl[p][j-1]=1.0;}
}
}//if
if ((fl[p][0]>=1.0)  &&  (fl[p-1][0]< 1.0)) //check end of melting
{
 G=12;
// update coefficients of node 1
h[1][0]=(-2*f*Bi);h[1][1]=(1+2*f*(1+Bi));h[1][2]=-2*f;h[1][Q]=(Tm-L/c*(1-fl[p-1][0]));
h[Q-2][Q-1]=h[1][0];h[Q-2][Q-2]=h[1][1];h[Q-2][Q-3]=h[1][2];h[Q-2][Q]=h[1][Q];
X[1]=(Tm-L/c*(1-fl[p-1][0]));
```

```
X[Q-2]=X[1];
 //calculate  temperature at the same time step
matrix= gauss( h, X,0,1,0.00000000000001,10000000,Q);
pw.println("%end of meltingrrrr "+(int)  (p));
for (int i=0;i<((Q-1)/2)+1;i++){
pw.print("      " +(double) (matrix[i]));}
pw.println("       ");
h[1][0]=(-2*fliq*Bl);h[1][1]=(1+2*fliq*(1+Bl));h[1][2]=-2*fliq;h[1][Q]=R[p][1];
h[Q-2][Q-1]=h[1][0];h[Q-2][Q-2]=h[1][1];h[Q-2][Q-3]=h[1][2];h[Q-2][Q]=h[1][Q];
X[1]=R[p][1];
for(int r=1;r<((Q-1)/2)+1;r++)
 {
 R[p][r]=matrix[r];
  X[r]=matrix[r];
 }
X[0]=matrix[0];
for (int i=1;i<((Q-1)/2)+1;i++)
{
 X[Q-i]=X[i-1];}
 }//end if
for(int q=2;q<(Q-((Q-1)/2))-1;q++)
{
if (R[p][q-1]>Tm){   // check start of melting of internal nodes
if ((R[p][q]>=Tm)  &&   (R[p-1][q]< Tm))
{
fl[p][q-1]=fl[p-1][q-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][q-1]-2*Tm+R[p][q+1])-(c/L*(Tm-
R[p-1][q]));
 if (fl[p][q-1]<=0.0)
  {fl[p][q-1]=0.0;}
if (fl[p][q-1]>=1.0)
  {fl[p][q-1]=1.0;}
// update coefficients of node  i
h[q][q-1]=0.0;h[q][q]=1.0;h[q][Q]=(Tm);h[q][q+1]=0.0;
h[Q-(q+1)][Q-(q+1)]=1.0;h[Q-(q+1)][Q-(q+2)]=0.0;h[Q-(q+1)][Q]=(Tm);h[Q-(q+1)][Q-(q)]=0.0;
X[q]=(Tm);
X[Q-(q+1)]=X[q];
```

```
matrix= gauss( h, X,0,1,0.00000000000001,1000000,Q);
pw.println("%correctionTT00011111");
for (int i=0;i<(Q+1)/2;i++){
pw.print("      " +(double) (matrix[i]));}
pw.println("       ");
for(int r=1;r<(Q+1)/2;r++)
 {
 R[p][r]=matrix[r];
  X[r]=matrix[r];
 }
X[0]=matrix[0];
for (int i=1;i<((Q-1)/2)+1;i++)
{
 X[Q-i]=X[i-1];}
for(int j=2;j<(Q-((Q+1)/2));j++)
{
if(j!=q)     //continue;
{
if (fl[p-1][j-1]!=1.0){
fl[p][j-1]=fl[p-1][j-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][j-1]-2*Tm+R[p][j+1]);
}
if (fl[p][j-1]<=0.0)
  {fl[p][j-1]=0.0;}
if (fl[p][j-1]>=1.0)
  {fl[p][j-1]=1.0;}
}
}
if (fl[p-1][(Q-((Q+1)/2))-1]!=1.0){
fl[p][(Q-((Q+1)/2))-1]=fl[p-1][(Q-((Q+1)/2))-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][(Q-
((Q+1)/2))-1]-2*Tm+R[p][(Q-((Q+1)/2))-1]);
}
if (fl[p][(Q-((Q+1)/2))-1]<=0.0)
  {fl[p][(Q-((Q+1)/2))-1]=0.0;}
if (fl[p][(Q-((Q+1)/2))-1]>=1.0)
  {fl[p][(Q-((Q+1)/2))-1]=1.0;}
}  ///end if
```

```
} ///end if new
}   // end  for
//check start of melting of central node
if ((R[p][(Q-((Q+1)/2))]>=Tm)  &&   (R[p-1][(Q-((Q+1)/2))]< Tm))
{
if (fl[p-1][(Q-((Q+1)/2))-1]!=1.0){ // correction of  liquid fraction
fl[p][Q-((Q+1)/2)-1]=fl[p-1][Q-((Q+1)/2)-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][Q-((Q+1)/2)]-
2*Tm+R[p][Q-((Q+1)/2)])-(c/L*(Tm-R[p-1][Q-((Q+1)/2)]));
 }
if (fl[p][(Q-((Q+1)/2))-1]<=0.0)
  {fl[p][(Q-((Q+1)/2))-1]=0.0;}
if (fl[p][(Q-((Q+1)/2))-1]>=1.0)
  {fl[p][(Q-((Q+1)/2))-1]=1.0;}
h[Q-((Q+1)/2)][Q-((Q+1)/2)-1]=0.0;h[Q-((Q+1)/2)][Q-((Q+1)/2)]=1.0;h[Q-
((Q+1)/2)][Q]=(Tm);h[Q-((Q+1)/2)][Q-((Q+1)/2)+1]=0.0;
X[Q-((Q+1)/2)]=(Tm);
X[Q-(Q-((Q+1)/2)+1)]=X[Q-((Q+1)/2)];
matrix= gauss( h, X,0,1,0.00000000000001,1000000,Q);
pw.println("%correctionTTcenter");
for (int i=0;i<((Q-1)/2+1);i++){
pw.print("      " +(double) (matrix[i]));}
pw.println("       ");
for(int r=1;r<((Q-1)/2+1);r++)
 {
 R[p][r]=matrix[r];
  X[r]=matrix[r];
 }
X[0]=matrix[0];
for (int i=1;i<((Q-1)/2)+1;i++)
{
 X[Q-i]=X[i-1];}
 } ///end if
for(int u=0;u<(Q-3)/2+1;u++)
{
if (fl[p][u] <= 0.0)
  {
```

```
  fl[p][u]=0.0;}
if (fl[p][u] >= 1.0){
  fl[p][u]=1.0;}
 F[p][u]=fl[p][u];
}
//check end of melting of node i
for(int  q=2;q<(Q-((Q+3)/2))+1;q++)
{
if ((fl[p][q-1]>=1.0)  &&  (fl[p-1][q-1]< 1.0))
{
GR=q-1;
int aa=q;  // update nodal coefficients
h[q][q-1]=-f;h[q][q]=(1+2*f);h[q][Q]=(Tm-L/c*(1-fl[p-1][q-1]));h[q][q+1]=-f;
h[Q-(q+1)][Q-(q+1)]=(1+2*f);h[Q-(q+1)][Q-(q+2)]=-f;h[Q-(q+1)][Q]=(Tm-L/c*(1-fl[p-1][q-
1]));h[Q-(q+1)][Q-(q)]=-f;
X[q]=(Tm-L/c*(1-fl[p-1][q-1]));
//X[q]=(Tm);
X[Q-(q+1)]=X[q];
// recalculate  temperature at the same time step
 matrix= gauss( h, X,0,1,0.00000000000001,1000000,Q);
pw.println("%end of meltingccc "+(double)(q));
for (int i=0;i<((Q-1)/2)+1;i++){
pw.print("      " +(double) (matrix[i]));}
pw.println("       ");
//printX(matrix);
 for(int r=1;r<((Q-1)/2)+1;r++)
 {
 R[p][r]=matrix[r];
  X[r]=matrix[r];
 }
X[0]=matrix[0];
for (int i=1;i<((Q-1)/2)+1;i++)
{
 X[Q-i]=X[i-1];}
h[q][q-1]=-fliq;h[q][q]=(1+2*fliq);h[q][q+1]=-fliq;h[q][Q]=R[p][q];
h[Q-(q+1)][Q-(q+1)]=(1+2*fliq);h[Q-(q+1)][Q-(q+2)]=-fliq;h[Q-(q+1)][Q-(q)]=-fliq;
```

```
 for(int e=aa+1;e<(Q-((Q-1)/2))-1;e++)
{    // check start of melting
if ((R[p][e]>=Tm)  &&   (R[p-1][e]< Tm))
{
if (fl[p-1][e-1]!=1.0){    /////new *****************
fl[p][e-1]=fl[p-1][e-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][e-1]-2*Tm+R[p][e+1])-(c/L*(Tm-
R[p-1][e]));
 }
if (fl[p][e-1]<=0.0)
  {fl[p][e-1]=0.0;}
if (fl[p][e-1]>=1.0)
  {fl[p][e-1]=1.0;}
h[e][e-1]=0.0;h[e][e]=1.0;h[e][Q]=(Tm);h[e][e+1]=0.0;
h[Q-(e+1)][Q-(e+1)]=1.0;h[Q-(e+1)][Q-(e+2)]=0.0;h[Q-(e+1)][Q]=(Tm);h[Q-(e+1)][Q-(e)]=0.0;
X[e]=(Tm);
X[Q-(e+1)]=X[e];
matrix= gauss( h, X,0,1,0.00000000000001,1000000,Q);
pw.println("%correctionTT0002");
for (int i=0;i<(Q+1)/2;i++){
pw.print("      " +(double) (matrix[i]));}
pw.println("       ");
for(int r=1;r<(Q+1)/2;r++)
 {
 R[p][r]=matrix[r];
  X[r]=matrix[r];
 }
X[0]=matrix[0];
for (int i=1;i<((Q-1)/2)+1;i++)
{
 X[Q-i]=X[i-1];}
for(int j=aa;j<(Q-((Q+1)/2));j++)
{
if(j!=e)     //continue;
{
if (fl[p-1][j-1]!=1.0){
fl[p][j-1]=fl[p-1][j-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][j-1]-2*Tm+R[p][j+1]);
```

```
}
if (fl[p][j-1]<=0.0)
  {fl[p][j-1]=0.0;}
if (fl[p][j-1]>=1.0)
  {fl[p][j-1]=1.0;}
}}
if (fl[p-1][(Q-((Q+1)/2))-1]!=1.0){
fl[p][(Q-((Q+1)/2))-1]=fl[p-1][(Q-((Q+1)/2))-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][(Q-
((Q+1)/2))-1]-2*Tm+R[p][(Q-((Q+1)/2))-1]);
}
if (fl[p][(Q-((Q+1)/2))-1]<=0.0)
  {fl[p][(Q-((Q+1)/2))-1]=0.0;}
if (fl[p][(Q-((Q+1)/2))-1]>=1.0)
  {fl[p][(Q-((Q+1)/2))-1]=1.0;}
}  ///end if
}   // end  for
}   ///if
}//for
// check start of melting of central node
if ((R[p][(Q-((Q+1)/2))]>=Tm)  &&   (R[p-1][(Q-((Q+1)/2))]< Tm))
{
if (fl[p-1][(Q-((Q+1)/2))-1]!=1.0){
fl[p][Q-((Q+1)/2)-1]=fl[p-1][Q-((Q+1)/2)-1]+d*dt/(roh*L*Math.pow(dx,2))*(R[p][Q-((Q+1)/2)]-
2*Tm+R[p][Q-((Q+1)/2)])-(c/L*(Tm-R[p-1][Q-((Q+1)/2)]));
 }
if (fl[p][(Q-((Q+1)/2))-1]<=0.0)
  {fl[p][(Q-((Q+1)/2))-1]=0.0;}
if (fl[p][(Q-((Q+1)/2))-1]>=1.0)
  {fl[p][(Q-((Q+1)/2))-1]=1.0;}
// update nodal coeffecients
h[Q-((Q+1)/2)][Q-((Q+1)/2)-1]=0.0;h[Q-((Q+1)/2)][Q-((Q+1)/2)]=1.0;h[Q-
((Q+1)/2)][Q]=(Tm);h[Q-((Q+1)/2)][Q-((Q+1)/2)+1]=0.0;
X[Q-((Q+1)/2)]=(Tm);
X[Q-(Q-((Q+1)/2)+1)]=X[Q-((Q+1)/2)];
matrix= gauss( h, X,0,1,0.00000000000001,1000000,Q);
pw.println("%correctionTTcenter");
```

```
for (int i=0;i<((Q-1)/2+1);i++){
pw.print("      " +(double) (matrix[i]));}
pw.println("       ");
for(int r=1;r<((Q-1)/2+1);r++)
 {
 R[p][r]=matrix[r];
  X[r]=matrix[r];
 }
X[0]=matrix[0];
for (int i=1;i<((Q-1)/2)+1;i++)
{
 X[Q-i]=X[i-1];}
 } ///end if
// check end of melting of central node
 if ((fl[p][(Q-((Q+1)/2))-1]>=1.0)  &&  (fl[p-1][(Q-((Q+1)/2))-1]<1.0))
 {
h[(Q-((Q+1)/2))][(Q-((Q+1)/2))]=(1+2*f);h[(Q-((Q+1)/2))][(Q-((Q+1)/2))-1]=-f;h[(Q-
((Q+1)/2))][Q]=(Tm-L/c*(1-fl[p-1][(Q-((Q+1)/2))-1]));h[(Q-((Q+1)/2))][(Q-((Q+1)/2))+1]=-f;
X[(Q-((Q+1)/2))]=(Tm-L/c*(1-fl[p-1][(Q-((Q+1)/2))-1]));  //////****take  care ****
matrix= gauss( h, X,0,1,0.00000000000001,1000000,Q);
  pw.println("%end of melting "+(double)(fl[p-1][(Q-((Q+1)/2))-1]));
//update nodal coefficients for liquid phase
h[(Q-((Q+1)/2))][(Q-((Q+1)/2))]=(1+2*fliq);h[(Q-((Q+1)/2))][(Q-((Q+1)/2))-1]=-fliq;h[(Q-
((Q+1)/2))][(Q-((Q+1)/2))+1]=-fliq;
for (int i=0;i<((Q-1)/2)+1;i++){
pw.print("      " +(double) (matrix[i]));}
pw.println("       ");
for(int r=1;r<((Q-1)/2)+1;r++)
 {
 R[p][r]=matrix[r];
  X[r]=matrix[r];
 }
X[0]=matrix[0];
for (int i=1;i<((Q-1)/2)+1;i++)
{
 X[Q-i]=X[i-1];}
```

```
 h[1][1]=(1+2*fliq*(1+Bl)); h[Q-2][Q-2]=h[1][1];
h[1][2]=-2*fliq;
h[Q-2][Q-3]=h[1][2];
h[1][0]=-2*fliq*Bl;
h[Q-2][Q-1]=h[1][0];
for(int k=2;k<(Q-2);k++){
    h[k][k]=(1+2*fliq);
    h[k][k+1]=-fliq;
    h[k][k-1]=h[k][k+1];
    }
 }//if
Ti=matrix[0];
for(int u=0;u<(Q-3)/2+1;u++)
{
if (fl[p][u] <= 0.0)
  {
  fl[p][u]=0.0;}
if (fl[p][u] >= 1.0){
  fl[p][u]=1.0;}
 F[p][u]=fl[p][u];
}
 h[1][Q]=R[p][1];
 h[Q-2][Q]=h[1][Q];
for(int i=3;i<((Q-1)/2+1);i++)
{
  h[i-1][Q]=R[p][i-1];
 h[Q-i][Q]=h[i-1][Q];}
 h[(Q-((Q+1)/2))][Q]=R[p][(Q-((Q+1)/2))];
 if (GR==1)
 {fl[p][GR]=1.0;}
if (G==12)
 {fl[p][0]=1.0;}
 if (GR!=0)
 {fl[p][GR]=1.0;}
Ti=matrix[0];
}
```

```
pw.println("];");
pw.print("plot(A(:,1))");
if (pw!=null) pw.close();
} //end main**********************
//methods
public static double[]  gauss(double A[][],double[] X,int FALSE,int TRUE,double TOL,int NN,int
N)
{
int k=1;
double b[]=new double[N];
int OK=0 ;
double ERR;
while (( OK == FALSE)  &&   (k<= NN ))
{
  ERR = 0.0;
for (int I = 0 ;I< N;I++)
 {
 double S = 0.0;
 for (int J = 0 ;J< N;J++)
 {
 S = S-A[I][J]*X[J];
 }
 S = (S+A[I][N])/(A[I][I]);
 if (Math.abs(S) > ERR)
 {
  ERR  = Math.abs(S);
 }
 X[I] = X[I] + S;
 }
//STEP 4
 // process is complete
  if (ERR > TOL)
 {
// STEP 5
  OK=FALSE;
  k=k+1;
```

```
//STEP 6 - is not used since only one vector is required
 }
else if (ERR <= TOL)  break;
 }
for (int o=0;o<N;o++)
{
b[o]=X[o];
}
return (double[])( b);
}   //end METHOD  GAUSS ITERATION
public static void printX(double[] m,int d)
{
for(int i=0;i<d;i++)
  System.out.print("   "+m[i]);
}
public static void prinXx(double[][] m,int d)
{
for(int i=0;i<d;i++)
{
for(int j=0;j<(d+1);j++)
System.out.print("   "+m[i][j]);
}
System.out.println("");
}
//file reader  **************************
public static double[] find(double g ,String  fil,int x )
{
FileReader frs =null;
StreamTokenizer in=null;
String sname="slam";
double [] r=new double[x];
double mid=0.0;
try{
frs=new FileReader(fil);
in=new StreamTokenizer(frs);
while(in.ttype!=StreamTokenizer.TT_EOF)
```

```
{
if (in.ttype==StreamTokenizer.TT_WORD)
 sname=in.sval;
 else
while (in.nextToken()==StreamTokenizer.TT_NUMBER)
{
mid=in.nval;
 if (mid==g){
for (int i=0;i<x;i++)
{
if (in.nextToken()==StreamTokenizer.TT_NUMBER)
mid=in.nval;
r[i]=mid;
}}}
in.nextToken();}
}
catch(FileNotFoundException ex) {
System.out.println("receved" );
}
catch(IOException ex)
{
System.out.println(ex.getMessage());
}
finally
{
try
{
if (frs!=null) frs.close();
}
catch(IOException ex)
{
System.out.println(ex);
}
}
return r;
}};
```

A.6 The Truncation Error :

Second Order Backward Difference

$$T\left(x_i, t_j - \Delta t\right) = T_{i,j-1} \qquad (1)$$

Taylor series expansion of $T_{i,j-1}$

$$T_{i,j-1} = T_{i,j} - \left.\frac{\partial T}{\partial t}\right|_{ij} \Delta t^1 + \left.\frac{\partial^2 T}{\partial t^2}\right|_{ij} \frac{\Delta t^2}{2!} - \left.\frac{\partial^3 T}{\partial t^3}\right|_{ij} \frac{\Delta t^3}{3!} + \cdots\cdots (2)$$

$$\text{from (2)}: \left.\frac{\partial T}{\partial t}\right|_{ij} = \frac{T_{i+1,j} - T_{ij}}{\Delta t} + O(\Delta t) \qquad (3)$$

$O(\Delta t)$ *is called the truncation error*

$$O(\Delta t) = \left.\frac{\partial^2 T}{\partial t^2}\right|_{ij} \frac{\Delta t^1}{2!} - \left.\frac{\partial^3 T}{\partial t^3}\right|_{ij} \frac{\Delta t^2}{3!} + \cdots\cdots \qquad (4)$$

$$T\left(x_i + \Delta x, t_j\right) = T_{i+1,j}$$

Taylor series expansion of $T_{i+1,j}$

$$T_{i+1,j} = T_{i,j} + \left.\frac{\partial T}{\partial x}\right|_{ij} \Delta x^1 + \left.\frac{\partial^2 T}{\partial x^2}\right|_{ij} \frac{\Delta x^2}{2!} + \left.\frac{\partial^3 T}{\partial x^3}\right|_{ij} \frac{\Delta x^3}{3!} + \cdots\cdots (1)$$

$$T\left(x_i - \Delta x, t_j\right) = T_{i-1,j}$$

Taylor series expansion of $T_{i-1,j}$

$$T_{i-1,j} = T_{i,j} - \left.\frac{\partial T}{\partial x}\right|_{ij} \Delta x + \left.\frac{\partial^2 T}{\partial x^2}\right| \frac{\Delta x^2}{2!} - \left.\frac{\partial^3 T}{\partial x^3}\right|_{ij} \frac{\Delta x^3}{3!} + \cdots\cdots\cdots (2)$$

by adding eq.1 and eq.2

$$T_{i+1,j} + T_{i-1,j} = 2T_{i,j} + 2\left.\frac{\partial^2 T}{\partial x^2}\right|_{ij} \frac{\Delta x^2}{2!} + 2\left.\frac{\partial^4 T}{\partial x^4}\right|_{ij} \frac{\Delta x^4}{4!} + - \ldots\ldots\ldots$$

$$\left.\frac{\partial^2 T}{\partial x^2}\right|_{ij} = \frac{T_{i+1,j} + T_{i-1,j} - 2T_{i,j}}{\Delta x^2} - 2\left.\frac{\partial^4 T}{\partial x^4}\right|_{ij} \frac{\Delta x^2}{4!} - \ldots\ldots\ldots$$

$$O(\Delta x^2) = -2\left.\frac{\partial^4 T}{\partial x^4}\right|_{ij} \frac{\Delta x^2}{4!} - \ldots\ldots\ldots$$

where $O(\Delta x^2)$ *is the truncation erro*

For the heat equation :

$$\frac{\partial T}{\partial x} = \alpha \frac{\partial^2 T}{\partial x^2}$$

by using the approximat ions in equations

$$\alpha \frac{\partial^2 T}{\partial x^2}\Big|_{ij} = \alpha \frac{T_{i+1,j} + T_{i-1,j} - 2T_{i,j}}{\Delta x^2} - 2\alpha \frac{\partial^4 T}{\partial x^4}\Big|_{ij} \frac{\Delta x^2}{4!} - .....$$

in this approximat ion

$$O(\Delta x^2) = -2\alpha \frac{\partial^4 T}{\partial x^4}\Big|_{ij} \frac{\Delta x^2}{4!} - .....$$

$$\frac{T_{i+1,j} - T_{ij}}{\Delta t} = \alpha \frac{T_{i+1,j} + T_{i-1,j} - 2T_{i,j}}{\Delta x^2} + O(\Delta x^2) + O(\Delta t)$$

The total trancation error $= O(\Delta x^2) + O(\Delta t)$

Appendix B:

Table .B.1. The temperature at Al_Aroub each two hours[20]

درجة الحرارة التقريبية لكل ساعتين لمحطة العروب 12 كم شمال مدينة الخليل .												الوقت ساعة
كانون1	تشرين2	تشرين1	أيلول	آب	تموز	حزيران	أيار	نيسان	آذار	شباط	كانون2	
9.30	10.90	15.50	16.10	19.30	20.10	18.80	15.60	12.30	7.70	5.20	7.30	0
8.70	9.90	14.50	15.00	18.40	19.10	17.80	14.50	11.60	7.10	4.70	6.60	2
8.20	9.00	13.40	13.80	17.40	18.00	16.70	13.40	10.90	6.50	4.20	6.00	4
7.80	8.30	12.70	13.00	16.70	17.30	16.00	12.70	10.50	6.10	3.90	5.50	6
8.50	9.60	14.10	14.60	18.00	18.70	17.40	14.10	11.40	6.90	4.60	6.40	8
11.70	15.10	20.10	21.20	23.60	24.70	23.40	20.30	15.10	10.20	7.40	10.20	10
13.60	18.30	23.60	23.10	27.00	28.20	26.90	23.90	17.30	12.10	9.10	12.40	12
14.50	19.90	25.30	27.00	28.60	29.90	28.60	25.70	18.40	13.10	9.90	13.50	14
13.90	18.90	24.30	25.80	27.60	28.90	27.60	24.60	17.70	12.50	9.40	12.80	16
12.50	16.40	21.40	22.70	25.00	26.00	24.70	21.70	16.00	11.00	8.10	11.10	18
10.80	13.50	18.30	19.20	22.00	22.90	21.60	18.70	14.00	9.20	6.60	9.10	20
9.90	11.90	16.60	17.30	20.30	21.20	19.90	16.70	12.90	8.20	5.70	8.00	22

Table.B.2 direct and diffuse solar radiation intensity falling at vertical and horizontal Surfaces (W/m^2) at 30^o N [20]

Date	Orian-tation	Daily Mean	Sun Time 03.00	04.00	05.00	06.00	07.00	08.00	09.00	10.00	11.00	12.00	13.00	14.00	15.00	16.00	17.00	18.00	19.00	20.00
June 21	N	35				150	155	90	15	0	0	0	0	0	15	90	155	150		
	NE	90				385	350	515	405	255	90	0	0	0	0	0	0	0		
	E	115				395	600	635	555	410	220	0	0	0	0	0	0	0		
	SE	80				175	315	385	380	325	220	75	0	0	0	0	0	0		
	S	15				0	0	0	0	45	90	105	90	45	0	0	0	0		
	SW	80				0	0	0	0	0	0	75	220	325	380	385	315	175		
	W	115				0	0	0	0	0	0	0	220	410	555	635	600	395		
	NW	90				0	0	0	0	0	0	0	90	255	405	515	530	385		
	Diff	25				20	35	40	45	50	55	55	55	50	45	40	35	20		
	H	345				130	345	560	745	900	1000	1025	1000	900	745	560	345	130		
July 23 and May 21	N	25				115	120	55	0	0	0	0	0	0	0	55	120	115		
	NE	85				340	505	940	380	230	60	0	0	0	0	0	0	0		
	E	115				365	595	640	565	420	225	0	0	0	0	0	0	0		
	SE	85				175	335	415	415	360	255	110	0	0	0	0	0	0		
	S	30				0	0	0	25	90	140	155	140	90	25	0	0	0		
	SW	85				0	0	0	0	0	0	110	255	360	415	415	335	175		
	W	115				0	0	0	0	0	0	0	225	420	565	640	595	365		
	NW	85				0	0	0	0	0	0	0	60	230	380	490	505	340		
	Diff	25				20	35	40	45	50	55	55	55	50	45	40	35	20		
	H	340				110	30	540	735	890	990	1020	990	890	735	540	320	110		
August 24 and April 20	N	5				45	30	0	0	0	0	0	0	0	0	0	30	45		
	NE	65				205	420	420	310	155	0	0	0	0	0	0	0	0		
	E	110				245	565	645	580	430	230	0	0	0	0	0	0	0		
	SE	105				145	375	490	510	455	350	200	25	0	0	0	0	0		
	S	70				0	0	50	140	215	265	285	265	215	140	50	0	0		
	SW	105				0	0	0	0	0	25	200	350	455	510	490	375	145		
	W	110				0	0	0	0	0	0	0	230	430	580	645	565	245		
	NW	65				0	0	0	0	0	0	0	0	155	310	420	420	205		
	Diff	20				10	30	40	45	50	55	55	55	50	45	40	30	10		
	H	315				50	255	480	680	835	940	985	940	835	680	480	255	50		
September 22 and March 22	N	0				0	0	0	0	0	0	0	0	0	0	0	0	0		
	NE	35				0	275	305	200	40	0	0	0	0	0	0	0	0		
	E	95				0	450	605	570	425	320	0	0	0	0	0	0	0		
	SE	123				0	365	550	605	565	460	315	140	0	0	0	0	0		
	S	130				0	60	175	285	370	425	445	425	370	285	175	60	0		
	SW	125				0	0	0	0	0	140	315	460	565	605	550	365	0		
	W	95				0	0	0	0	0	0	0	230	425	570	605	450	0		
	NW	35				0	0	0	0	0	0	0	0	40	200	305	275	0		
	Diff	20				0	25	35	40	45	50	50	50	45	40	35	25	0		
	H	255				0	150	375	575	735	835	870	835	735	575	375	150	0		
October 23 and February 23	N	0					0	0	0	0	0	0	0	0	0	0	0			
	NE	20					140	190	95	0	0	0	0	0	0	0	0			
	E	80					280	515	520	400	215	0	0	0	0	0	0			
	SE	135					260	540	640	625	535	395	230	55	0	0	0			
	S	170					85	245	385	480	540	560	540	480	385	245	85			
	SW	135					0	0	0	55	230	395	535	625	640	540	260			
	W	80					0	0	0	0	0	0	215	400	520	515	280			
	NW	20					0	0	0	0	0	0	0	0	590	190	140			
	Diff	15					15	30	40	45	45	45	45	45	40	30	15			
	H	205					65	260	455	610	705	740	705	610	455	260	65			
November 21 and January 21	N	0					0	0	0	0	0	0	0	0	0	0	0			
	NE	5					35	95	15	0	0	0	0	0	0	0	0			
	E	60					95	395	450	360	195	0	0	0	0	0	0			
	SE	135					95	460	620	635	565	440	290	125	0	0	0			
	S	180					45	260	425	540	605	625	605	540	425	260	45			
	SW	135					0	0	0	125	290	440	565	635	620	460	95			
	W	60					0	0	0	0	0	0	195	360	450	395	95			
	NW	5					0	0	0	0	0	0	0	0	15	95	35			
	Diff	15					5	25	35	40	40	45	40	40	35	25	5			
	H	160					15	165	345	940	580	610	580	940	345	165	15			
December 22	N	0					0	0	0	0	0	0	0	0	0	0	0			
	NE	5					5	65	0	0	0	0	0	0	0	0	0			
	E	55					15	340	415	340	185	0	0	0	0	0	0			
	SE	130					15	415	595	625	570	450	305	150	10	0	0			
	S	180					10	245	425	550	615	640	615	550	425	245	10			
	SW	130					0	0	10	150	305	450	570	625	595	415	15			
	W	55					0	0	0	0	0	0	185	340	415	340	15			
	NW	5					0	0	0	0	0	0	0	0	0	65	5			
	Diff	15					0	20	30	40	40	40	40	40	30	20	0			
	H	140					0	130	295	435	525	560	525	435	293	130	0			

Table. B.3 . ***Thermophysical propertiesof*** $CaCl_2 . 6H_2O$ ***[3]***

melting temperature Tm **= 29°C**
latent heat L **= 190.0kJ/kg**
<u>thermal conductivity</u> ***solid phase*** **= 1.09 W/mK** ***liquid phase*** **= 0.53 W/mK**
<u>specific heat</u> ***solid phase*** c **= 1.4 kJ/kgK** ***liquid phase*** c **= 2.2 kJ/kgK**
<u>density</u> ***solid phase*** **= 1710 kg/m^3** ***liquid phase*** **= 1530 kg/m^3**

Table B.4 Glazing Characteristics. [23]

Glass—single layer

*Light transmission**: 85-90%

*R-value***: 0.9

Advantages:
• Lifespan indefinite if not broken
• Tempered glass is stronger and requires fewer support bars

Disadvantages:
• Fragile, easily broken
• May not withstand weight of snow
• Requires numerous supports
• Clear glass does not diffuse light

Polyethylene—single layer

*Light transmission**: 80-90% - new material

*R-value***: single film 0.87

Advantages:
• Heat loss significantly reduced when a blower is used to provide an air space between the two layers
• IR films have treatment to reduce heat loss
• No-drop films are treated to resist condensation
• Treatment with ethyl vinyl acetate results in resistance to cracking in the cold and tearing
• Easy to install, precise framing not required
• Lowest cost glazing material

Disadvantages:
• Easily torn
• Cannot see through
• UV-resistant polyethylene lasts only 1–2 years
• Light transmission decreases over time
• Expand and sag in warm weather, then shrink in cold weather

Polyethylene—double layer

*Light transmission**: 60-80%

*R-value*** double films: 5ml film 1.5, 6ml film 1.7

Advantages:
• Heat loss significantly reduced when a blower is used to provide an air space between the two layers
• IR films have treatment to reduce heat loss
• No-drop films are treated to resist condensation
• Treatment with ethyl vinyl acetate results in resistance to cracking in the cold to tearing
• Easy to install, precise framing not required
• Lowest cost glazing material

Disadvantages:
• Easily torn
• Cannot see through
• UV-resistant polyethylene lasts only 1–2 years
• Light transmission decreases over time
• Expand and sag in warm weather, then shrink in cold weather

cont. tableB.4

Fiber reinforced plastic (FRP)

*Light transmission**: 85-90% - new material

*R-value***: single layer 0.83

Advantages:

- The translucent nature of this material diffuses and distributes light evenly
- Tedlar treated panels are resistant to weather, sunlight, and acids
- Can last 5 to 20 years

Disadvantages:

- Light transmission decreases over time
- Poor weather resistance
- Most flammable of the rigid glazing materials
- Insulation ability does not cause snow

* note that framing decreases the amount of light that can pass through and is available as solar energy

** R-Value is a common measure of insulation (hr°Fsq.ft/BTU)

Table B.5 Optimal Values of Air Temperature in Greenhouses for Vegetable Cultivation. [24]

Vegetable	Inside Air Temperature (°C)						
	Germination	Development			Harvesting		Young
		Day*	Day*	Night	Day	Night	Plants
Cucumbers	17-18	22-25	27-30	17-18	25-30	18-20	13-15
Watermelon and melons	17-18	22-25	27-30	17-18	25-30	18-20	13-15
Tomatoes, apple, paprika, and beans	10-12	20-22	25-27	10-13	22-28	15-17	8-10

*** Inside design temperature ranges for different crops.**

Table.B.6. Statistical analysis (T-Test) of the results:

Fig. no	Time	curves	Means	variances	t-value	p-value	conclusion
2.8	(0-120)min	our numerical solution experemental data	33.29429 33.75336	70.89335 66.58945	0.19576	0.84563	Not significantly different
	(80-120)min	our numerical solution experemental data	42.79519 43.2	36.60625 17.65	0.16487	0.87111	Not significantly different
3.3	6am-6pm	Inside temperature Outside temperature	40.11665 25.05714	20.51802 146.7976	4.35619	1.8766E-4	**Significantly different**
	7pm-6am	Inside temperature Outside temperature	20.84286 21.48365	7.79495 16.28597	0.48859	0.62922	Not significantly different
3.4	6am-10am	V=1 V=6	33.99857 30.84928	189.01548 125.993	-0.39677	0.70191	Not significantly different
	11am-4pm	V=1 V=6	54.58909 48.45382	7.17848 4.93288	-4.3183	0.00152	Significantly different
	6pm-6am	V=1 V=6	21.3469 21.13933	19.25757 15.09887	-0.12768	0.89946	Not significantly different
3.5	6am-11am	Singlepoly doublepoly	34.12824 36.57449	154.70081 203.73989	0.3165	0.75813	Not significantly different
	12am-4pm	Singlepoly doublepoly	36.57449 53.83629	203.73989 6.40372	2.91678	0.01539	Significantly different
	5pm-6am	Singlepoly doublepoly	22.43215 22.84664	36.89609 48.21722	0.1681	0.8678	Not significantly different
	11am-4pm	Single glass singlepoly	48.47902 49.3849	4.94086 5.2349	0.06956	0.50252	Not significantly different
4.4	6am-11am	Without pcm With TH29+fans	34.6578 32.757	127.23368 103.38194	-0.41513	0.68246	Not significantly different
	12am-4pm	Without pcm With TH29+fans	51.47868 45.26683	4.01383 2.41872	-7.34767	1.64178E-6	Significantly different
	11pm-5.5am	Without pcm With TH29+fans	19.00137 21.59922	1.27573 1.15666	6.4338	5.74422E-6	Significantly different
	12pm-4pm	Without pcm With TH29	51.47868 49.47686	4.01383 1.56591	-2.54236	0.02174	Significantly different
	12am-6am	Without pcm With TH29	18.74437 19.85963	0.95128 0.80415	3.035	0.00571	Not significantly different
4.5	6am-12am	Without pcm With pcm21	39.01874 35.97415	171.35461 127.69615	0.63479	0.053157	Not significantly different
	10am-4pm	Without pcm With pcm21	53.63505 49.70382	6.26226 6.61686	3.94963	5.98367E-7	Significantly different
	6pm-6am	Without pcm With pcm21	21.22366 21.358	13.87453 13.34474	-0.12875	0.89809	Not significantly different
	6am-10am	Without pcm With TH29	32.95619 30.6277	121.71738 85.79425	-0.48492	0.6343	Not significantly different
	10am-4pm	Without pcm With TH29	53.63505 50.39617	6.26226 5.84895	-3.35562	0.00263	Significantly different
	6pm-6am	Without pcm With TH29	21.22366 22.39424	13.87453 12.63202	1.13683	0.26125	Not significantly different
	1am-6am	Without pcm With TH29	18.52046 19.97364	0.63867 0.51485	4.48749	2.25343E-4	Significantly different
4.7	6am-11am	Doublepoly+TH29 Singlepoly+TH29	33.92665 32.66567	122.81451 102.17831	-0.27882	0.78325	Not significantly different
	11am-4pm	Doublepoly+TH29 Singlepoly+TH29	51.39077 49.16494	1.61146 1.622	-3.82641	0.00124	Significantly different
	6pm-6am	Doublepoly+TH29 Singlepoly+TH29	25.37077 24.77807	70.11025 61.95634	-0.27774	0.78223	Not significantly different
4.10	11am-4pm	Without pcm With pcm21+fans	26.34504 23.36994	4.88669 1.25029	-3.9831	7.31887E-4	Significantly different
	8pm-7am	Without pcm With pcm21+fans	7.03511 8.79093	1.38035 1.46109	4.99547	9.78737E-6	Significantly different
	11pm-4am	Without pcm With pcm21	26.34504 25.87864	4.88669 2.62698	-0.56433	0.5788	Not significantly different
	8pm-7am	Without pcm With pcm21	7.03511 7.4876	1.38035 1.92027	1.19448	0.23869	Not significantly different

Cont. table.B.6

Fig. no	Time	curves	Means	variances	t-value	p-value	conclusion
4.11	10am-5pm	Without pcm With pcm21	24.72917 24.40543	14.72189 9.4565	-0.25499	0.8006	Not significantly different
	8pm-7am	Without pcm With pcm21	7.03511 7.4876	1.38035 1.92027	1.19448	0.23869	Not significantly different
	8pm-7am	Without pcm With pcm17	7.03511 7.86118	1.38035 1.62523	2.28516	0.0271	Significantly different
4.16	7pm-12am	Without pcm With pcm21	6.77792 7.62405	1.38672 2.02151	1.6525	0.11146	Not significantly different
	12am-7am	Without pcm With pcm21	4.59581 4.91827	0.26699 0.35279	1.58639	0.12388	Not significantly different
	7pm-12am	Without pcm With pcm17	4.59581 5.01524	0.26699 0.41078	1.97321	0.05842	Not significantly different
	12am-7am	Without pcm With pcm17	17.41275 16.20828	40.0277 34.57482	-0.46251	0.64871	Not significantly different
	7pm-12am	Without pcm With pcm13	17.41275 16.20826	40.0277 32.77336	-0.51704	0.6108	Not significantly different
	12am-7am	Without pcm With pcm13	4.59581 5.46759	0.2669 0.21588	-4.85896	4.07843 E-5	Significantly different

For the statistical analysis the t-test method is done in table for asignificant level 0.05 , if the p-value > 0.05 for two data sets then they are not significantly different , if the p-value ≤ 0.05 then they are significantly different .

References:

[1] J. S. Langer., July, 1998, Final report on the DOE-NSF National Workshop of Advanced Scientific Computing. Technical report, 30-31.

[2] Hanan, J.J., 1998, *Advanced technology for protected horticulture*. C.R.C. Press, NY, 684 pp.

[3] B. Zivkovic et al., 2001, *An analysis of isothermal phase change of phase change material within rectangular and cylindrical containers*. Solar Energy, Vol. 70, No.1, pp. 51-61.

[4] Lane.G.A ,1983,Solar heat storage :Latent heat materials ,volume 1Ph.D,CRC Press,Inc.Boca Roton,Florida.

[5]Piia Lamberg, 2004, Approximate analytical model for two-phase solidification problem in a finned phase-change material storage, Applied Energy, 77, 131– 152, www.sciencedirict.com.

[6] Hawes D.W., Feldman D., Banu D, 1993, Latent heat storage in building materials, *Energy and Buildings, 20*, pp. 77-86.

[7] Alexiades V., Solomon A.D., 1993, Mathematical modelling of melting and freezing processes, Hemisphere publishing corporation, USA.

[8] Velraj R, Seeniraj R, Hafner B, Faber C, Schwarzer K., 1999, Heat-transfer enhancement in a latent-heat storage system. Solar Energy;65(3):171–180.

[9] Satzger P, Eska P, Ziegler F., 1997, Matrix-heat-exchanger for latent-heat-cold-storage. In: Proceedings 7th International conference on thermal-energystorage. p. 307–311.

[10] Humphries W, Griggs E., 1977, A design handbook for phase-change thermal control and energy storage devices. NASA TP-1074.

[11] Stritih U, Novak P., August 28–September 1, 2000, Heat transfer enhancement in phase-change processes.In: Proceedings Terrastock 8th International Conference on Thermal-EnergyStorage, vol. 1,. p. 333–8.

[12] Rathjen KA, Latif MJ., 1971, Heat conduction with melting or freezing in a corner. Journal of Heat Transfer;February:101–109.

[13] Lamberg P, Siren K., 2003, Analytical model for melting in a semi-

finite PCM storage with an internal fin. Heat and Mass Transfer;39:167–176.
[14] Voller V R, 1990, Fast implicit finite-difference method for the analysis of phase change problems. *Numerical Heat Transfer, Part B* 17, 155-169.

[15]F. AGHBALOU, J. ILLA, F. BADIA, L.F CABEZA AND J. ROCA, 2002, *Design and analysis of athermal solar energy storage using RT40-PCM ,* FIER', Tétouan – Macro,http://www.fst.ac.ma/fier/

[16]*Phase change materials and eutectic solutions, http://www.epsltd.co.uk.*

[17] Druma, A.M., 1998: Dynamic climate model of a greenhouse. Report 3 in G*eothermal Training in Iceland.* UNU G.T.P., Iceland, 51-85.

[18] Muthafar S. Emeish, 1999, *Geothermal heating system for Jordainian greenhouses*, http://www.os.is/Apps/WebObjects/Orkustofnun.woa/swdocument/1839/02Muthafar.pdf

[19] Gene A. Giacomelli ,William .Roberts,1993,Greenhouse covering systems, NJAES Paper No. D-03130-17-92 (http://www.hortglaz.pap)

[20] Palestinian ministry of local government, 2004, Guidelines for energy efficient building design ,Beesan corporation Rammala ,Palestine.

[21] Rafferty, K.D., 1998: Greenhouses. In: Lund J.W., Lienau, P.J., and Lunis, B.C.(editors),*Geothermal direct use engineering and design guidebook. 3*rd edition,Geo-Heat Center, Klamath Falls, OR, 307-332.

[22] Chao, K. and R.S. Gates., 1999, Fuzzy control simulation of plant and animal *enviroments*, Paper No. 993136, An ASAE Meeting Presentation. http://www.bae.uky.edu.

[23] Appropriate technology transfer for rural areas,April 2003, Solar greenhouses. http://attra.ncat.org/attra-pub/solar-gh.html.

[24] Geo-heat center,2004,Greenhouse climate factors, http://geoheat.oit.edu/bulletin/bull18-1/art36.html