FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

AREVA

Master-Thesis

Algorithm to Solve Inverse Heat Conduction Problem for Evaluating Thermal Stratification

Author: Helal Uddin Chowdhury

Computational Engineering

LTM

AREVA Supervisors: Dipl.-Ing. Benoit Jouan

Dr.-Ing. Jürgen Rudolph
PEEA-G, Mechanical Analysis

FAU Supervisor: Prof. Dr.-Ing. Julia Mergheim

Chair of Applied Mechanics

University of Erlangen-Nuremberg

Declaration

I hereby declare that this thesis being submitted for the partial fulfillment of the M.Sc. degree in Computational Engineering is my own work.

I also declare that, as required by thesis regulations, all materials contained herein that are not original to the work are fully cited and duly acknowledged.

_____

Author

Erlangen, June 10, 2013

Acknowledgement

I would like to take the opportunity to express the deepest appreciation to all of my supervisors who guided me from the beginning. First and foremost, I cannot find words to express my gratitude to Mr. Benoit Jouan for his persistent guidance and suggestions, without his effort this thesis would not have been possible; he continually and convincingly conveyed a spirit of adventure in regard to the work.

It is with immense gratitude that I acknowledge the unique support of Dr. Jürgen Rudolph, who pointed out some important issues during the work and who managed time to read and correct the report in spite of his busy hours.

I consider it an honor to work with Prof. Dr. Julia Mergheim since even a small discussion with her always shows a new dimension in reshaping my thinking. So, it gives me great pleasure in acknowledging her invaluable cooperation not only during the work but also from the beginning of my master study.

I am indebted to many colleagues who assisted me in numerous supporting works during the period.

Abstract

The concept of an inverse problem has gained widespread acceptance in modern applied Mathematics and have been extensively studied over the last 50 years. They have numerous applications in many branches of science and technology. Fields of applications comprise the development of new materials, the casting and welding in steel or polymer processing, the development of a sophisticated temperature measurement related to lifetime analysis of plants, the development of transient calorimeters, ill-posed problems in rheometry, geophysical inverse problems etc.

In thermal analysis, this kind of problems mostly relates determining the temperature, heat flux or heat source at inaccessible parts of the boundary of a 2- or 3-dimensional body from corresponding data - called 'Cauchy data' - on accessible parts of the boundary. It is well-known that inverse heat conduction problems (IHCP) are severely ill-posed which means that small perturbations in the data may cause extremely large errors in the solution. [1]

Heat conduction phenomena are governed by the well known transport equation. In this work, a generic algorithmic approach based on ‘Principle of Superposition’ to solve a second order linear homogeneous PDE for direct and inverse heat conduction has been attempted. The objective is to evaluate thermal stratification correctly for thermal fatigue assessment in the context of nuclear power plant components.

The algorithm is tested extensively for circular domain which, however, does not imply that it is limited for only circular domain. The results for 2-D circular domain ensure that the method is stable with respect to perturbations in the Cauchy data. Many irregular domains of fairly complex shape can be successfully handled without altering the form of the heat equation and the algorithm can be applied.

An object oriented console application has been developed based on C++ for solving direct and inverse problem by implementing these generalized algorithms. The module is supposed to be used for further numerous tests and, if necessary, can be extended with new modules for further thermo-structural analysis.

Table of Contents

List of Figures

Symbol and notation

Symbol	***Description***
$\boldsymbol{c}$	*Specific heath*
$\boldsymbol{k}$	*Thermal conductivity*
$\boldsymbol{\rho}$	*Density*
$\boldsymbol{T}$	*Temperature*
$\boldsymbol{n}$	*Normal vector on a boundary surface*
$\boldsymbol{h}$	*Heat Transfer coefficient*
$\boldsymbol{t}$	*Time*
$\boldsymbol{t_f}$	*Final time*
$\boldsymbol{r}$	*Radial distance*
$\boldsymbol{T_o(r)}$	*Initial Temperature at point r*
$\boldsymbol{T_s}$	*Temperature at a boundary surface*
$\boldsymbol{\dot{q}_v}$	*Heat Source*
$\boldsymbol{q}$	*Heat flux*
$\boldsymbol{\Omega}$	*Domain*
$\boldsymbol{\partial\Omega}$	*boundary of the domain*
$\boldsymbol{\partial\Omega_D, \partial\Omega_N, \partial\Omega_R}$	*D for Dirichlet, N for Neumann, R for Robin boundary*
$\Delta\boldsymbol{T}(\boldsymbol{r}, \boldsymbol{t_1})$	*Temperature change at point r during global time step* t_1
$\boldsymbol{t_i}$	*Global time step*
$\boldsymbol{\lambda_i}$	*Local time step*
$\Delta\boldsymbol{T^{ref}}$	*Some reference 'temperature change'*
$\Delta\boldsymbol{T_i^r}$	*Response at ith subdomain due to* $\Delta\boldsymbol{T^{ref}}$
$\boldsymbol{\Phi_i^r}$	*Response function at ith subdomain due to* $\Delta\boldsymbol{T^{ref}}$
$\boldsymbol{[\,]}$	*Used to indicate a vector quantity*

List of Abbreviations

Abbreviation	Definition
AFC.	*AREVA Fatigue Concept*
APDL	*ANSYS Parametric Design Language*
BC	*Boundary Condition*
CDS	*Condition for Determining Solution*
CG	*Conjugate Gradient*
EAF	*Environmentally Assisted Fatigue*
EPR™	*European Pressurized Reactor*
FAMOS	*FAtigue MOnitoring System*
FHCP	*Forward Heat Conduction Problem*
IHCP	*Inverse Heat Conduction Problem*
KTA	*Kerntechnischer Ausschuss*
MA	*Moving Agerage*
NPP	*Nuclear Power Plant*
PDE	*Partial Differential Equation*
PDS	*Problems for Determining solution*
PWR	*Pressurized Water Reactor*
TAM	*Thermal Analysis Module*
UML	*Unified Modeling Language*
UTM	*Unit Transient Method*

1. Introduction and background

1.1 Thermal Stratification

As early as 1979, a through wall crack was detected in a *pressurized water reactor (PWR)* plant. This crack was initiated at the adjacent to the weld joint attaching the pipe to the steam generator feed water nozzle [Fig.-1.1]. Subsequent inspection of the remaining feed water piping revealed cracking in the same vicinity but these were limited to partial wall penetration. As a result of this incident, the US Nuclear Regulatory Commission issued a directive to all operating plants requiring them to perform inspection of their feed water lines. An exhaustive investigation was undertaken subsequently and this revealed that the primary cause of cracking was due to a fatigue loading mechanism induced by *thermal stratification* and high cycle thermal oscillations (*striping*) during low flow conditions.[2] Many similar locations are crack sensitive and thus the point of studying fatigue assessment during the operation of power plants. Figure 1.2 shows some possible crack sensitive points and fatigue analysis plane nearby.

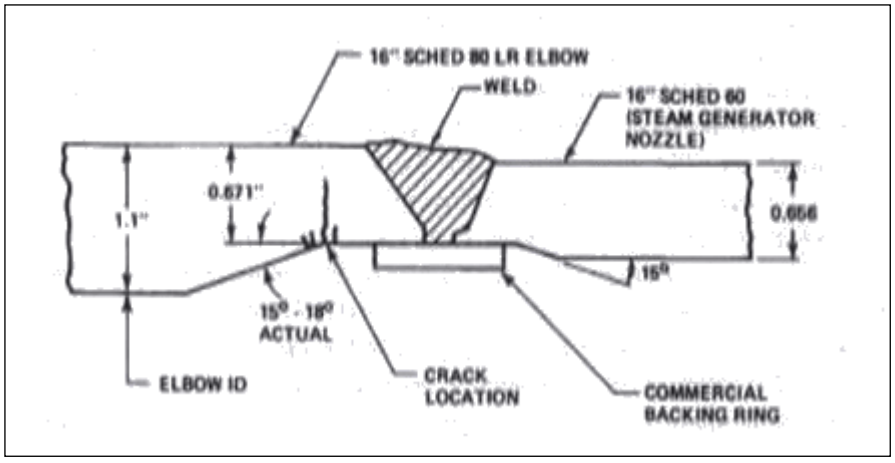


Fig-1.1: Location of Cracks in PWR Feed water Pipe to Nozzle Attachment Region. [2]

Thermal stratification phenomenon results from a temperature differential across the pipe cross-section with the top fluid stream hot and bottom stream relatively cold. During normal plant operations at low flow conditions, mixing effect of fluid streams cannot be controlled and thus thermal stratification appears. The difference in buoyancy between the hot and cold fluids inhibits their mixing so that the feed water becomes and remains thermally stratified. [2][3] Figure-1.3 shows a real example of thermal stratification in the nuclear power plant and figure-1.4 shows a typical effect on stress profile due to thermal stratification.

The complex fluid flow events occurring during the operation of NPPs are influenced by the automatic operational control processes.

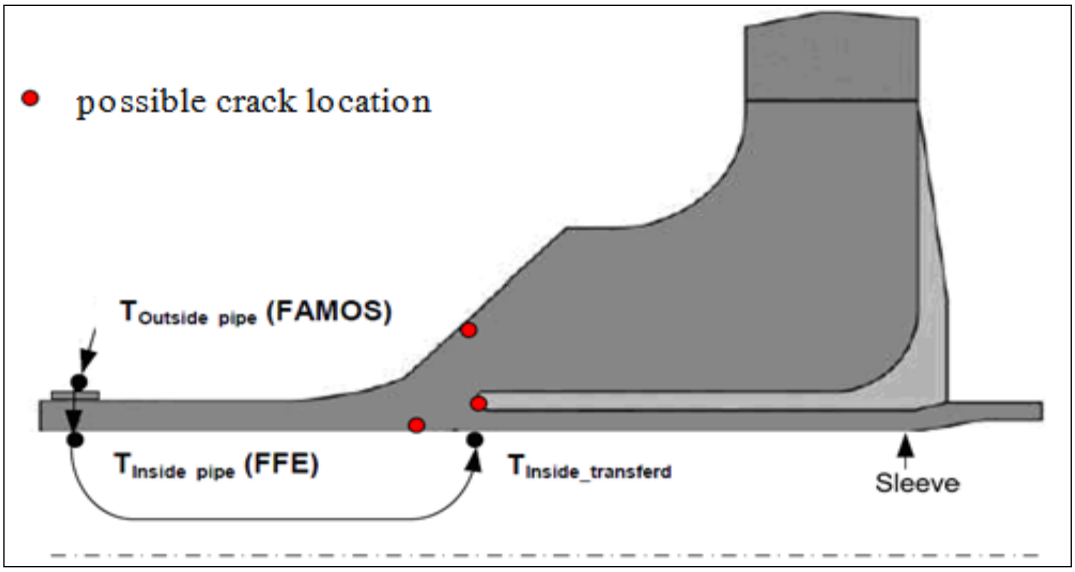


Fig-1.2: Possible crack and Fatigue relevant locations in the NPPs

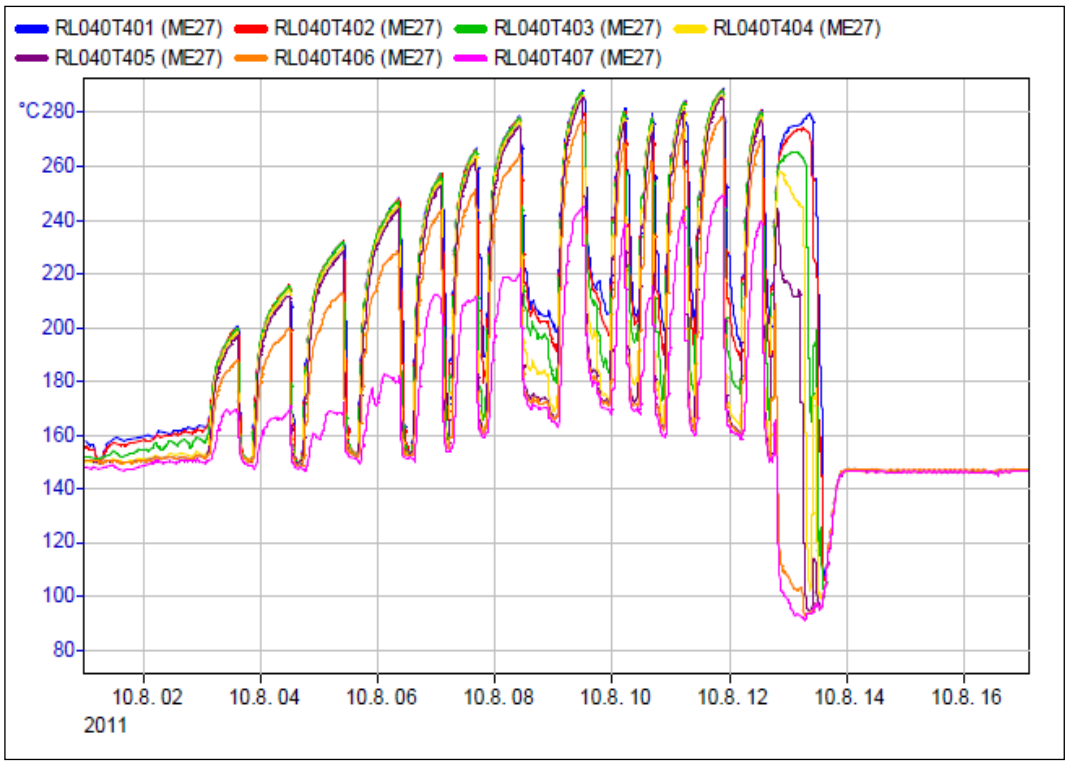


Fig-1.3: an example of thermal stratification in the NPP

The effect of the thermal stratification on the state of stress in the pipe is manifested in two ways: (a) the difference in temperature between the top and bottom of the pipe causes greater thermal expansion at the top tending to bow the pipe. When such bowing is restrained global bending stresses result. In other words, Thermal stratification causes a stress distribution in a pipe that is similar to what happens in a bimetallic strip. In the hot upper region compressive stresses develop as a result of constrained expansion, with the tensile stresses occurring in the lower

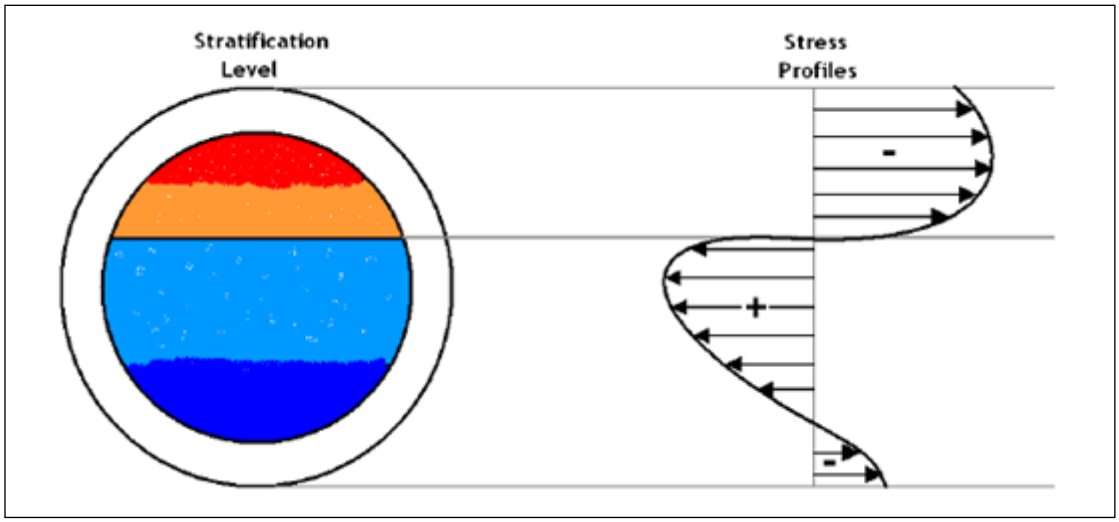


Fig-1.4: Stress distribution due to thermal stratification

region. Each of the stratification profiles produces a complex stress each point in the respective area is subjected to varying stress state; (b) the interface between the two fluid layers causes a local stress in the pipe due to thermal discontinuity across the pipe section. The fatigue damage produced by thermal stratification and the associated thermal striping are a good indication of the contribution of these phenomena to the observed feed water line cracking.

In the specific case of thermal stress, the load is due to a non-uniform temperature distribution, no-free expansion or external constrains. A good explanation of thermal stress into a material is given by Bruno Boley: "*Imagine a body as made up of a number of small cubical elements of equal size which fit together to form the given continuous body. If the temperature of the body is raised uniformly, and if it's bounding surfaces are unrestrained, then each element will expand an equal amount uniformly in all direction. At the end, they still fit together to form a continuous body, and no stresses arise. If, however, the temperature rise is not uniform, each element will tend to expand by different amount that is one proportional to its own temperature rise. The resulting different-sized cubes cannot, fit together; however, the body must remain continuous, each element must restrain the distortions of its neighbors, or, in other words, stresses must arise*" [4].

During operation of NPPs, fatigue relevant thermal cyclic loadings occur due to transient states of operation e.g. respective cold or hot feed conditions occur during start-up and shutdown as well as testing conditions of the safety equipment. Furthermore, permanently occurring mixing events of hot and cold flows at junction locations may induce high cycle fatigue loads. Certain plant conditions may induce temperature stratification events within larger pipes at lower flow rates and an existing temperature difference. These phenomena may equally induce cyclic loads in the pipelines and the attached components. [5]

Not only the crack initiation, the major cause of growth of the cracks is due to the thermal stratification cycles, which occur during low flows, primarily at hot standby. The thermal striping phenomenon or the oscillations occurring at the interface between hot and cold fluids has some influence on the crack growth, but it certainly impacts the crack initiation predictions. Damages have been observed in the main feed water lines, pressurizer spray lines, branch piping connected to reactor coolant piping [see fig. 1.5], and pressurizer surge lines with evidences linked to thermal stratification. [2]

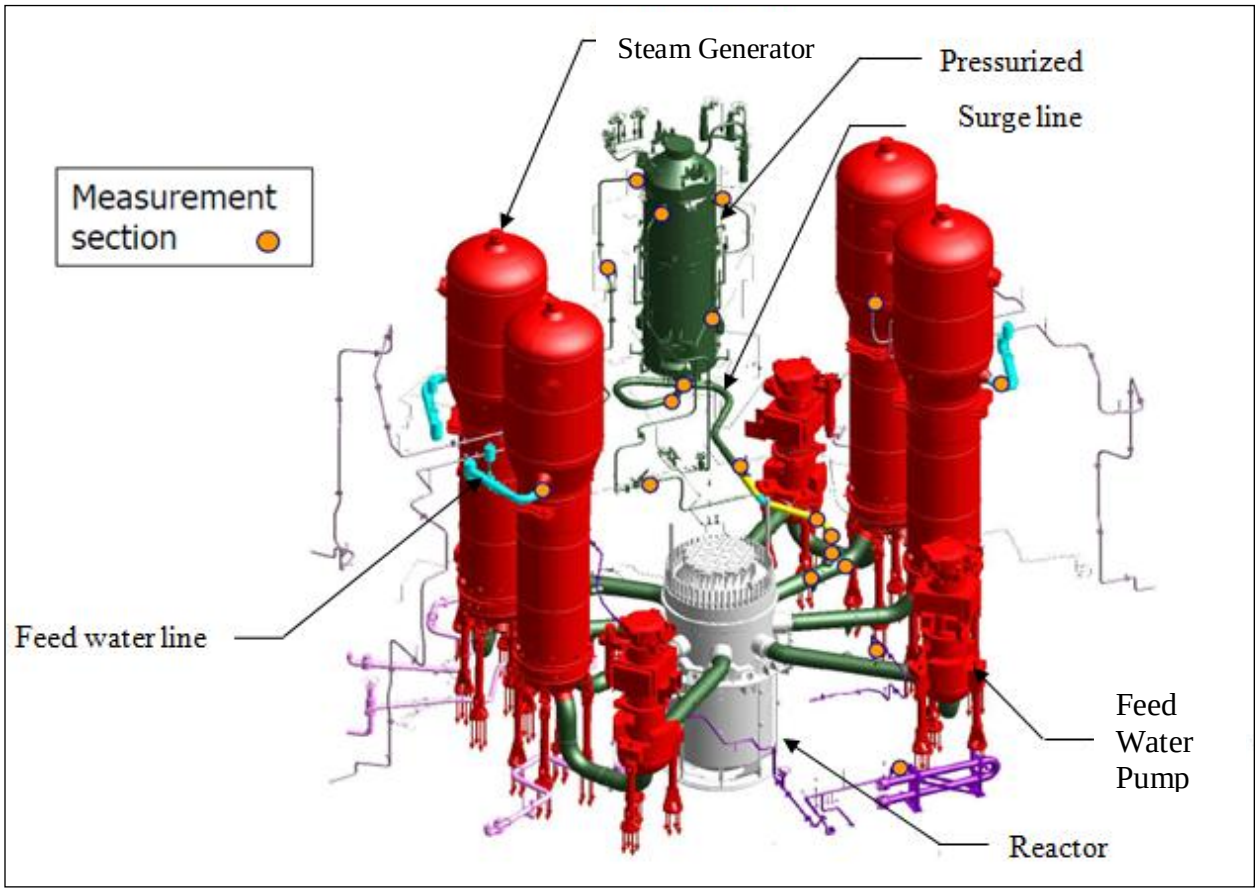


Fig-1.5: FAMOS measurement sections in a PWR[7]

1.2 AREVA Fatigue Concept (AFC)

A comprehensive ageing management process is necessary for NPPs since ageing and lifetime management plays a key role in the continuously conforming licensing process till the end of their operational lifetime. One of the main tasks in this regard is to assure structural integrity of the systems and components. In this regard, AREVA adopted a powerful tool AFC, integrated with hardware and software, to deal with life cycle management of NPP's by conforming licensing and operational requirements, ensuring structural integrity, safety and security. In fact, it is an integrated process of data acquisition, analysis, qualitative assessment, fatigue tendency evaluation, monitoring etc.; which provides different code-conforming fatigue analyses (e.g. according to the wide spread ASME code based on realistic loads. The AFC uses two different

methods: the global fatigue monitoring and the local fatigue monitoring. It has two main stages; (a) the building's stage, which starts with designed data and operation and (b) operation's stage, which starts with measured data [see Fig-1.6]. The transient catalogue [not shown in the figure] constitutes the basis of the design fatigue checks. This first fatigue check is part of the licensing documents and should also indicate fatigue relevant positions and plant processes. If necessary, modifications of the plant processes and/or components are carried out in the design phase with the aim of eliminating potentially critical positions. The FAMOS is the central module of the AFC, which is highly automated process that able to monitor and record the real local operating loads [5], monitors fatigue into three stages known as simplified, fast and detail fatigue evaluation respectively.

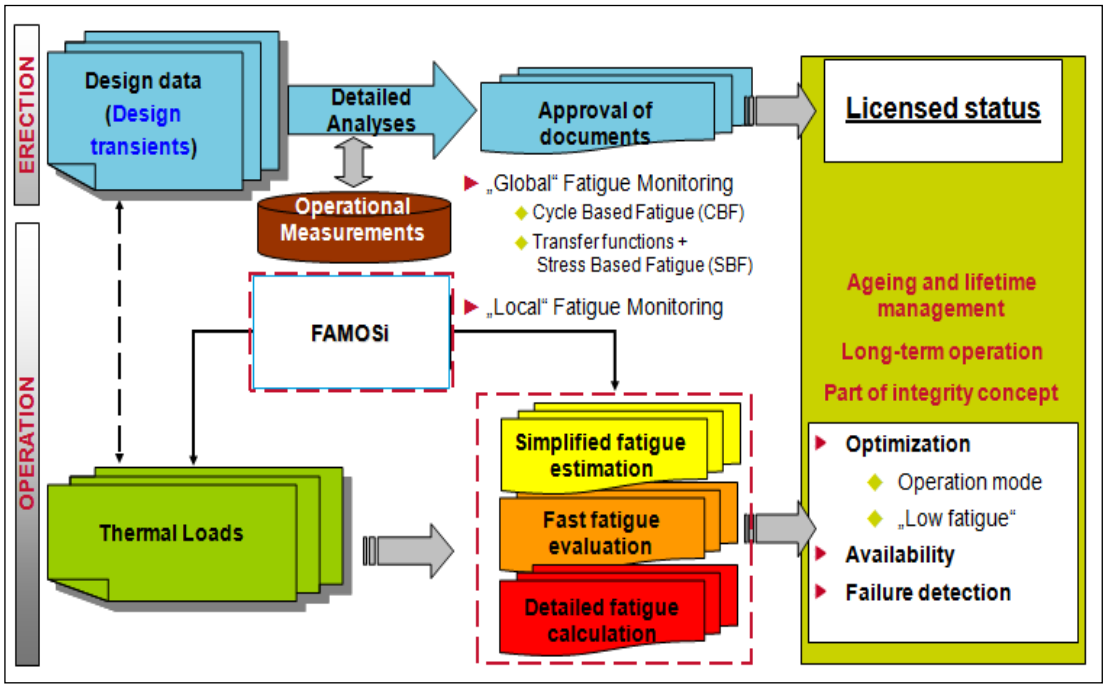


Fig-1.6: Modules of the AFC [5]

Regarding the aspects of new lifetime periods of nuclear power generation works (60 years of operation for new NPPs such as AREVA's EPR™ or generation III+ reactor) or due to lifetime extension projects (e.g. in the USA, Sweden or Switzerland) there is an increasing need of knowing the current state of the plant almost exactly in order to enable a qualified respective assessment. Moreover, thermal conditions and chemical composition of the fluid inside the piping system influence the allowable fatigue levels, which have come under extensive review due to the consideration of environmentally assisted fatigue (EAF), synonymous to the corrosive influence of the medium on the fatigue behavior. Therefore, for highly loaded components, some new and improved stress and fatigue evaluation methods, not overly conservative, are needed to meet the increasingly stringent allowable fatigue levels.

1.3 Fatigue Monitoring System (FAMOS)

FAMOS is termed as the heart of the AFC. The objectives of this module can be summarized as:

- to determine the fatigue status of the most highly stressed components
- to identify and optimize the operating modes which are unfavorable to fatigue e.g. valve leakages
- to improve the catalogue of transients used at the design phase
- to establish a basis for fatigue analysis based on realistic operating loads
- to use the results for lifetime management and lifetime extension.

It measures the thermal transient load data at fatigue relevant points of the pressure boundary of the primary and secondary circuits thus providing a realistic replica of the load history.[6] Thermocouples are being installed in various measurement sections including primary loops, surge line, spray lines, volume control system, feed-water system and some other relevant positions [Fig-1.5].

FAMOS is mainly focused on *local fatigue monitoring*, close to each fatigue relevant locations. Local fatigue monitoring is located at fatigue relevant locations at the outer surface of pipes and is based on additional temperature measurement by means of thermocouples.[5] Each measurement section consists of two (vertical) to seven (horizontal) thermocouples that are installed on a thin metal tape and a robust protection shell to prevent the thermocouples from being damaged for recording transient stratified thermal history. Both metal tape and protection shell are installed around the pipe under the piping insulation [shown in the figure-1.7, 1.8].

Fatigues are evaluated, analyzed and monitored by FAMOS in three stages that can be summarized as follows:

Stage-1: Simplified fatigue estimation

Simple estimations of fatigue relevance of real loads for components are based on thermal mechanical considerations (based on measured temperature). A basic decision about fatigue relevance (YES/NO) for the monitored position is made. In case of fatigue relevance (YES) based on some fatigue limit, a further evaluation is activated according to step 2.

Stage-2: Fast fatigue evaluation (FFE)

A code-conforming usage factor U is calculated in a highly automated way based on the simplified elasto-plastic fatigue analysis, computing inner side temperature distribution, simplified EAF etc. If $U \leq U_{admissible}$ the fatigue check will be successfully finished. If $U > U_{admissible}$ further analyses will be based on step 3. This thesis actually deals with this fast fatigue evaluation FFE, dedicated to improve computational methods and techniques.

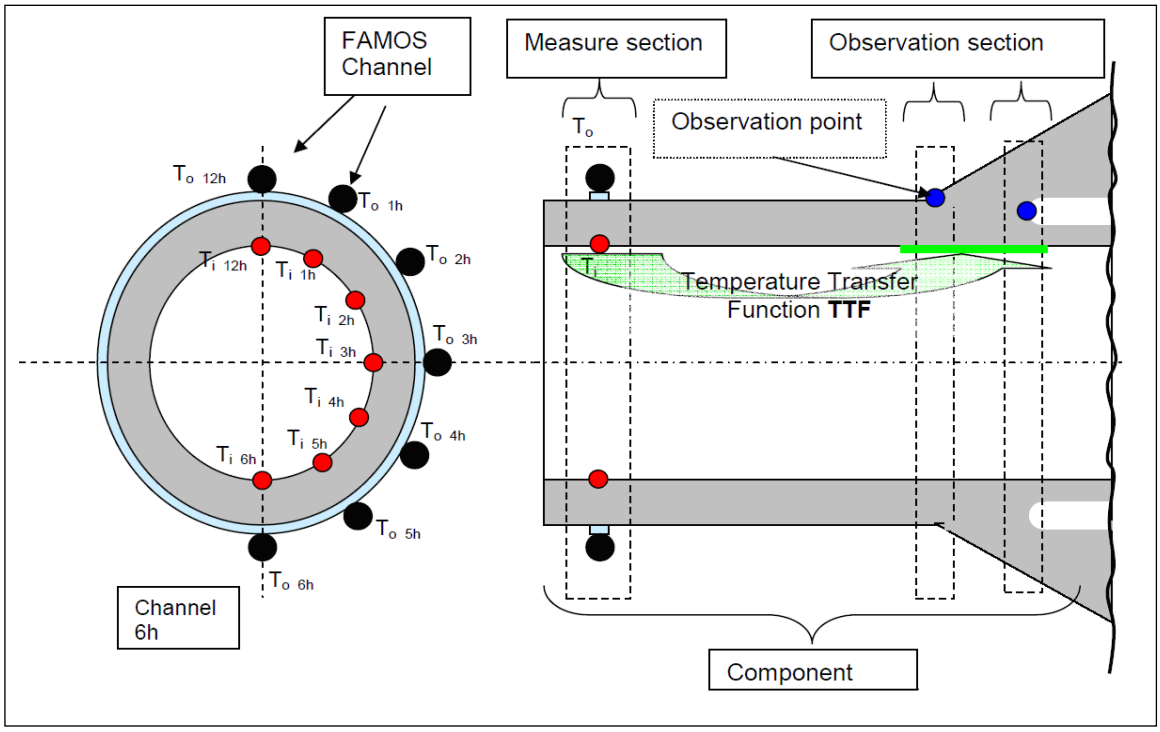


Fig-1.7: Thermo-couple's settings around the pipe near fatigue relevant locations [8]

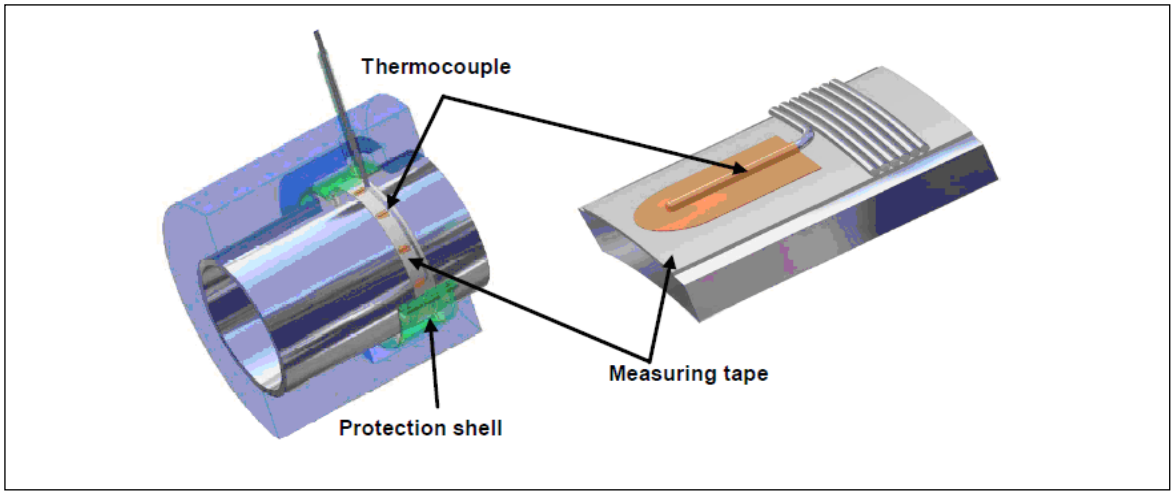


Fig-1.8: 3D view of Thermo-couple's settings [5]

Stage-3: Detailed fatigue calculation

The detailed fatigue calculation (DFC) is usually carried out after a certain time period of plant operation, ten years for instance. These analyses are often performed in the framework of the periodic safety inspection (PSI). Loading data of the operational period as well as anticipated loads of future operation are used as essential input parameters. Hence, usage factors are calculated for the current state of the plant and some prognoses are taken into account to get results until the end of life. The less conservative elasto-plastic (material model) fatigue analysis

method based on non-linear FE analyses will often be used for fatigue design. This is associated with an increased calculation effort. Computing times for complex geometries and numerous transients may be significant. In this step, there are scopes to consider the detailed EAF in the analysis process. [5]

The development of FAMOS was started in early 80's by KWU (Kraftwerk or Powerstation Union) then by Siemens and nowadays, it is continued by AREVA. Currently AFC uses a 1-D numerical approach in the second stage of FFE, known as unit transient method (UTM), which is in fact a simplified form of superposition principle, to calculate inside temperature.[3] Since AFC is in continuous improvement; it is necessary to explore more precise and rigorous techniques for analyzing inside temperature and stress for cycle counting. To improve analyzing thermal stratification phenomena is the main objective and motivation of this thesis.

1.4 Previous work and current contribution:

Our objective is to evaluate thermal stratification as correctly as possible. If we know somehow fluid temperature at different zones inside the pipe wall then possibly we could have possibly calculate wall temperature by using Newton's law of cooling. However, heat transfer coefficient or film coefficient h_c is different for different fluids which introduce some complexity in the equation. Alternatively, we can think the problem as an inverse one where inside wall temperature history or boundary condition is missing. A simple schematic of these two ideas is shown as per figure-1.9. Currently FAMOS is using a one dimensional algorithm based on Duhamel integral to solve inverse heat conduction problem and to know inside wall thermal conditions. The goal of this work is to generalize the algorithm for higher dimensions especially for 2-D pipe with different inner zones or strata or layers to accommodate with thermal stratification phenomena.

Before going to concentrate on generalizing algorithm it is better to have a look at the integral form of the Duhamel theorem. When surface temperature of a body, heat flux or the temperature of a surrounding fluid are a function of time, then temperature distribution inside the body can be determined by means an integral formula known as Duhamel integral. This integral derives from the superposition principle and can be applied to linear transient heat conduction problems when initial temperature of the body is uniform and equals T_o. In order to determine temperature field when surface temperature is time-variable, $T_o + f(t)$, it is necessary to determine $f(t)$ by solving a transient heat conduction problem when inner surface temperature T_{in}, undergoes a unit step-increase. Basically this theorem employs a "building block" solution which is used with the superposition principle to obtain the temperature at any point and time; and produces a convenient numerical approximation.[9][10]

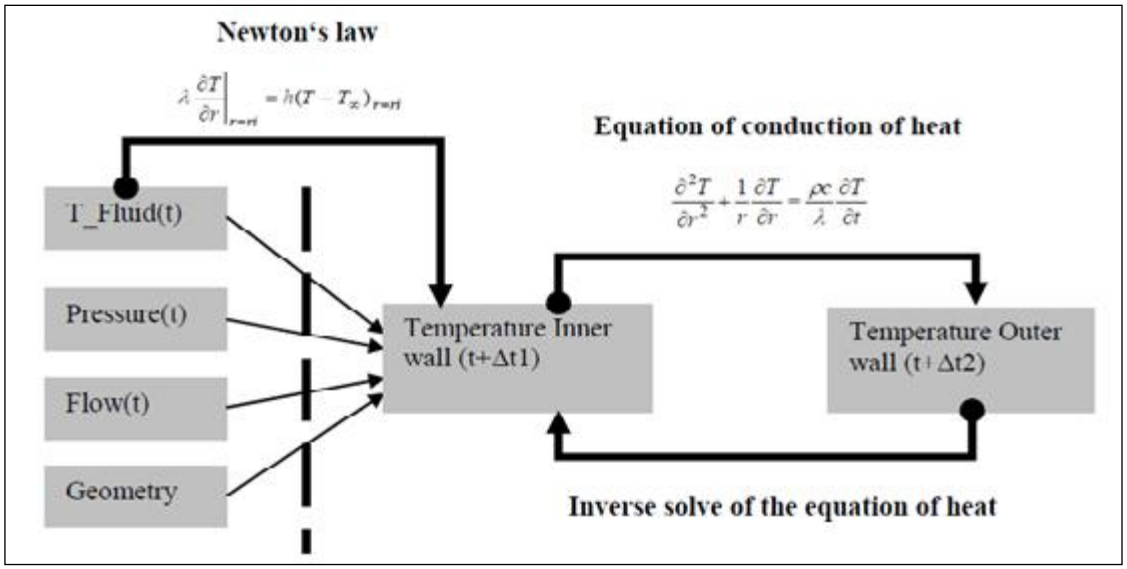


Fig. 1.9: Possible techniques of Inner wall temperature calculation [5]

The Duhamel's integral formula and its numerical form for 1-D can be represented as:

$$\boldsymbol{T(t) = T_0 + \int_0^t q(\lambda)\left[-\frac{\partial \emptyset(r,t-\lambda)}{\partial \lambda}\right] d\lambda} \quad \textbf{(2.1)}$$

$$\boldsymbol{T(t_M) = T_0 + \sum_{n=1}^{M} q_n \Delta \emptyset(t_{M-n})} \quad \textbf{(2.2)}$$

Where, $\lambda = t_i$, $i = \{0,1,2 \ldots\ldots M\}$ is time, t is the total time, $q(\lambda)$ is the heat flux at time λ and $\emptyset(t)$ is a temperature response function due to unit heat flux at inner surface. Though the above formula is based on heat flux, there is also temperature based formula as well which is being tested in the previous work; which can be represented as:

$$\boldsymbol{T_{out}\,(t_i) \;=\; \sum_{i=1}^{t_M} K_i . \emptyset(t_{M-i})} \quad \textbf{(2.3)}$$

Here K_i is the inside temperature at time t_i, and $\emptyset(t_i)$ is the response at time t_i due to some reference temperature change at the inside. Now, for simplicity, if we consider $T_{out}\,(t_i) = T_i$, and $\emptyset(t_i) = \emptyset_i$, then we can rewrite the above equation as:

$$\boldsymbol{T_1 = K_1 \emptyset_1}$$

$$\boldsymbol{T_1 = K_1 \emptyset_2 + K_2 \emptyset_1}$$

$$\boldsymbol{T_1 = K_1 \emptyset_3 + K_2 \emptyset_2 + K_3 \emptyset_1}$$ and so on.

This approach of solving 1-D inverse heat conduction problem has some basic limitations. *Firstly*, time step is considered strictly one second and hence the name of the algorithm is unit transient method (UTM), which sometimes may cause very small value of $\emptyset_1$, generally expected for less conducting wall. In each step division by such small value of $\emptyset_1$ might cause

divergence of the solution. *Secondly*, previous work showed that for sharp change of ***dT/dt*** creates some error in the solution and the error is comparatively high for bigger thickness. *Thirdly*, the most vital point is, it assumes uniform temperature or ***plug-flow*** inside the pipe which is very unlikely for thermal stratification problems. So, the algorithm is compromised for some error always even for small thickness assuming plug-flow condition. *Finally*, it assumes uniform initial temperature all over the geometry.

For generalizing this system of equations for higher dimensions, it is observed that it does not offer a straight forward flexibility, what means that there are different $\emptyset_i$in different equations with the same K. In other words, the equations do not represent the same superposition and scaling in all the equations, what is not favorable for formulating a multidimensional inverse heat conduction problem. As a result, a little modification is introduced in the formula maintaining perfect combination of superposition and scaling which has been done in this work. The basic idea is to decompose everything into steps, like time step Δ**t,** temperature rise or fall $\Delta \boldsymbol{T}$, difference in response $\Delta\emptyset$ etc. Before concentrating onto its algorithmic stages, probably it is worth to look into the basic mathematical aspects, clear generalized problem definition, problem type, its characteristic, limitations and existence of solution etc. These aspects will be discussed step by step in the coming chapters.

2. Problem Definition

2.1 Generalized heat conduction equation and simplification

For a control volume, general energy balance equation; or governing equation for solids, gases or liquids, which move (flow) at w velocity can be represented as:

$$\rho c_p \frac{DT}{Dt} = \nabla \cdot (\Lambda \nabla T) + \dot{q}_v + \frac{Dp}{Dt} + \Phi \qquad (2.1)$$

Where ρ is density of the medium, c_p is specific heat and Λ is generalized heat conductivity of the medium, $\dot{q}_v$ is volumetric heat source, p is pressure and Φ is dissipation function which is an irreversible power of internal friction forces, which influence the moving liquid particles and *D/Dt is* material time derivative what is actually

$$\frac{D}{Dt} = \frac{\partial}{\partial t} + w_x \frac{\partial}{\partial x} + w_y \frac{\partial}{\partial y} + w_z \frac{\partial}{\partial z}$$

Mathematically the dissipation function Φ is represented as:

$$\Phi = 2\mu \left[\left(\frac{\partial w_x}{\partial x}\right)^2 + \left(\frac{\partial w_y}{\partial y}\right)^2 + \left(\frac{\partial w_z}{\partial z}\right)^2 + \frac{1}{2}\left(\frac{\partial w_y}{\partial x} + \frac{\partial w_x}{\partial y}\right)^2 + \frac{1}{2}\left(\frac{\partial w_z}{\partial y} + \frac{\partial w_y}{\partial z}\right)^2 + \frac{1}{2}\left(\frac{\partial w_x}{\partial z} + \frac{\partial w_z}{\partial x}\right)^2 \right] \qquad (2.2)$$

Where μ stands for a dynamic viscosity.

With regards to the general governing equation, Thermo-physical properties ρ, C and *p* may be position or temperature dependent. A body can also be anisotropic, when the thermal conductivity is direction-dependent. Rate of energy generation per unit volume of heat sources $\dot{q}_v$ (or $\dot{Q}_v$) can be a function of position, temperature and time. Term *Dp/Dt* will be only taken into consideration in the cases of supersonic flows otherwise will be omitted since it is very small.[9]

Omitting *Dp/Dt and* replacing Λ with corresponding directional conductivities λ_{xx}, λ_{yy}, and λ_{zz} we can rewrite the generalized energy balance equation as:

$$\rho c_p \left(\frac{\partial T}{\partial t} + w_x \frac{\partial T}{\partial x} + w_y \frac{\partial T}{\partial y} + w_z \frac{\partial T}{\partial z}\right) = \frac{\partial}{\partial x}\left(\lambda_{xx} \frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(\lambda_{yy} \frac{\partial T}{\partial y}\right) + \frac{\partial}{\partial z}\left(\lambda_{zz} \frac{\partial T}{\partial z}\right) + \dot{q}_v + \Phi \qquad (2.3)$$

If a body is immovable, then $w_x = w_y = w_z = 0$ and $\Phi = 0$. Thermo-physical properties as usually can be temperature or position-dependent. If we accept such conditions, the above equation is simplified to a form:

$$\rho c_p \frac{\partial T}{\partial t} = \frac{\partial}{\partial x}\left(\lambda_{xx} \frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(\lambda_{yy} \frac{\partial T}{\partial y}\right) + \frac{\partial}{\partial z}\left(\lambda_{zz} \frac{\partial T}{\partial z}\right) + \dot{q}_v \qquad (2.4)$$

If thermo physical properties of a body ρ, C and λ's are temperature invariant and material is isotropic e.g. $\boldsymbol{\lambda_{xx} = \lambda_{yy} = \lambda_{zz} = k}$ then the problem becomes simplified second order linear PDE which can be represented in much simpler form as:

$$\boldsymbol{\rho c_p \frac{\partial T}{\partial t} = \nabla \cdot (k \nabla T) + \dot{q}_v} \qquad \textbf{(2.5)}$$

2.2 Initial and Boundary Conditions (BC)

Transient heat conduction problems are initial-boundary value problems for which it is required to assign appropriate initial and boundary conditions. *Initial conditions*, also called *Cauchy conditions*, are temperature values of a body Ω at its initial moment *t=0 s*, where Ω represents the whole domain.

$$\boldsymbol{T(x, y, z, 0) = T_0(x, y, z) \quad for\ (x, y, z) \in \Omega} \qquad \textbf{(2.6)}$$

2.2.1 First Kind Boundary Condition (Dirichlet Condition)

Temperature distribution on the edge of a body or boundary of the domain is assigned as follows:

$$\boldsymbol{T(x, y, z, t) = T_b(x, y, z, t) \quad for\ (x, y, z, t) \in S_D,\ t \in (0, t_f)} \qquad \textbf{(2.7)}$$

Where, $\boldsymbol{(x, y, z) = r}$ is a positional vector of a point located on the body's surface. If temperature of the body surface $\boldsymbol{T_b}$ *is* known from measurements taken, then the boundary conditions can be formulated as boundary conditions of the 1st kind. Other names of this type Boundary condition (BC) are essential or Dirichlet or prescribed value BC.

2.2.2 Second Kind Boundary Conditions (von Neumann Conditions)

Second kind boundary condition or heat flux boundary condition or Neumann condition for an anisotropic body is represented by the form:

$$\left(\lambda_{xx}\frac{\partial T}{\partial x}n_x + \lambda_{yy}\frac{\partial T}{\partial y}n_y + \lambda_{zz}\frac{\partial T}{\partial z}n_z\right)\Big|_A = \dot{q}(r_A, t)$$

$$\boldsymbol{for\ (x, y, z, t) \in S_N,\ t \in (0, t_f)} \qquad (2.8)$$

Where, $\boldsymbol{n_x = \cos(n, x), n_y = \cos(n, x)}$ and $\boldsymbol{n_z = \cos(n, x)}$ are directional cosines of a normal to the corresponding boundary or body surface. For isotropic bodies, the above condition becomes:

$$\left(\lambda\frac{\partial T}{\partial n}\right)\Big|_A = \dot{q}(r_A, t) \qquad (2.9)$$

f the wall is thermally insulated (adiabatic) then heat flux is zero then the Neumann condition would be:

$$\left(\frac{\partial T}{\partial n}\right)\Big|_A = 0$$

The boundary condition of 2nd kind is frequently set on the surface of radiated bodies, e.g. on the surface of boilers' radiant tubes. If the heat flux from a body surface is known from measurements taken, then the boundary condition of 2nd kind can be applied irrespectively of the type of heat transfer present on the body surface. Given condition is often set when solving steady-state and transient inverse heat conduction problems. If thermo physical properties of a body *c*, *p* and *k* are temperature invariant, then the inverse problem becomes linear, thereby easier to solve with boundary condition of 1st or 2nd kind is applied on the body surface.

2.2.3 Third Kind Boundary Conditions

The boundary condition of 3rd kind is also known as *Robin boundary condition* or *Newton law of cooling.* It is actually mixed type e.g. combination of Dirichlet and Neumann conditions. Its heat penetration coefficient, also called *heat transfer coefficient,* expresses the intensity of convective heat exchange. Coefficient $\boldsymbol{\alpha}$ (sometimes denoted by $\boldsymbol{h_c}$) is dependent on the type of heat exchange that occurs on a body's surface, the fluid type, and on the velocity and direction of the fluid flow with regard to body's surface. During boiling, condensation and natural convection, heat transfer coefficient or film coefficient h_c is also frequently a function of surface temperature or of the difference between surface temperatures *T* and T_c.

$$-(\Lambda \nabla \boldsymbol{T} \cdot \boldsymbol{n})|_{\boldsymbol{A}} = \boldsymbol{\alpha}(\boldsymbol{r_A}, \boldsymbol{t}, \boldsymbol{T_A})[\boldsymbol{T}(\boldsymbol{r_A}, \boldsymbol{t}) - \boldsymbol{T_c}] \qquad (2.10)$$

If a body is isotropic, then the above condition is simplified to a form:

$$-\boldsymbol{k}\frac{\partial \boldsymbol{T}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z},\boldsymbol{t})}{\partial \boldsymbol{n}} = \boldsymbol{h_c}\big(\boldsymbol{T}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z},\boldsymbol{t}) - \boldsymbol{T_c}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z},\boldsymbol{t})\big) \quad \boldsymbol{for}\ (\boldsymbol{x},\boldsymbol{y},\boldsymbol{z},\boldsymbol{t}) \in \boldsymbol{S_R},\ \boldsymbol{t} \in (\boldsymbol{0}, \boldsymbol{t_f}] \qquad \textbf{(2.11)}$$

In practice, the application of the 3rd kind boundary condition encounters difficulties with respect to the determination of spatial heat transfer coefficient changes and the medium's temperature at small flow velocity. If a liquid remains in a state of rest, then as a result of natural convection the medium moves alongside the solid's surface demonstrating significant temperature pulsations at the same time. Due to this reason, it is difficult to define the medium's temperature *Tc,* hence AREVA uses the inverse problem and calculate the inner temperature of the pipe without looking for Fluid temperature. It also should be added that spatial-temporal changes in the heat transfer coefficient on the surface of a solid can be determined when a conjugated heat transfer problem in a liquid and solid is solved using a computerized fluid mechanics program, which consists of different CFD methods. [9]

2.2.4 Other Boundary conditions

Boundary condition of forth kind might be for radiation, phase change or the boundary conditions at the point where two body surfaces meet. If a body surface is heated or cooled by radiation, then boundary condition is nonlinear. If phase-changes occur, for instance during the

process of melting or freezing, then a heat absorption or emission during the melting of a substance takes place on the phase boundary between a liquid and a solid. The boundary between the liquid and the solid phase is time-variable; this is why this kind of problems is called *a free and movable boundary.* [9]

2.3 Forward and Inverse Heat Conduction Problem

Heat conduction is only type of heat flow that occurs in non-transparent solids. In the cases of gases and fluids, heat conduction usually occurs in combination with other forms of conduction, such as convection and radiation. [9] Generally the integral form for heat transfer shows the total amount of energy which flows in or out the body. On the other hand, the differential form shows the local energy's fluxes through the material. [3] In the case of heat conduction problems the governing equations in the differential form and possible boundary and initial conditions can be represented as:

$$\rho c_p \frac{\partial T}{\partial t} = \nabla \cdot (k \nabla T) + \dot{q_v}\,, \quad (x,y,z) \in \Omega \subset R^3,\ t \in (0, t_f] \tag{2.12}$$

$$T(x,y,z,t) = T_b(x,y,z,t) \quad for\ (x,y,z,t) \in S_D,\ t \in (0, t_f] \tag{2.13}$$

$$-k \frac{\partial T(x,y,z,t)}{\partial n} = q_b(x,y,z,t) \quad for\ (x,y,z,t) \in S_N,\ t \in (0, t_f] \tag{2.14}$$

$$-k \frac{\partial T(x,y,z,t)}{\partial n} = h_c\big(T(x,y,z,t) - T_c(x,y,z,t)\big) \quad for\ (x,y,z,t) \in S_R,\ t \in (0, t_f] \tag{2.15}$$

$$T(x,y,z,0) = T_o(x,y,z,0) \quad for\ (x,y,z) \in \Omega \tag{2.16}$$

Where ρ denotes density of the solid, [kg/m3]; c is the constant-volume specific heat, [J/kg K]; T is temperature,[K]; k denotes thermal conductivity, [W/m K]; $\dot{q_v}$ or $\dot{Q}_v$ is the rate of heat generation per unit volume, [W/m3], frequently termed as source function; ∂/∂n means differentiation along the outward normal; h_c denotes the heat transfer coefficient, [W/m2 K]; T_b ,q_b and T_0 are given functions and Tc stands for environmental temperature, t_f final time. The boundary $\partial\Omega$ of the domain Ω is divided into three disjoint parts S_D, S_N, and S_R, denoted with subscripts D for Dirichlet, N for Neumann and R for Robin boundaries.

The equation (2) with conditions (3) to (6) describes an initial-boundary value problem for transient heat conduction. In the case of stationary problem the equation (2) becomes a Poisson equation or – when the source function $\dot{Q}_v$ is equal to zero – a Laplace equation.

In the classical heat-conduction equation (2), we have used the Fourier law of heat conduction. It was the first constitutive relation of heat flux and was proposed by the French mathematical physicist Joseph Fourier in 1807 based on experimentation and investigation. [11]

If the heat flux and/or temperature histories at the surface of a solid body are known as functions of time, then the field problem is of a direct type and generally considered as well posed and solvable, thus the temperature distribution can be found. However in many heat transfer situations, the surface heat flux and temperature histories must be determined from transient temperature measurements at one or more interior locations, what is considered as an inverse problem. Briefly speaking one might say the inverse problems are concerned with determining causes for a desired or an observed effect. [12] The inverse transient heat conduction problem is a class of problems in which some of the boundary conditions are unknown and are to be determined, based on the known or measured temperature variation with time at some accessible points. [13]

The concept of an inverse problem have gained widespread acceptance in modern applied mathematics, although it is unlikely that any rigorous formal definition of this concept exists. Most commonly, by inverse problem is meant a problem of determining one or more missing boundary conditions, various quantitative characteristics of a medium such as density, thermal conductivity, surface loading, shape of a solid body etc. by observation over physical fields in the medium or – in other words – a general framework that is used to convert observed measurements into information about a physical object or system that we are interested in. The fields may be of natural appearance or specially induced, stationary or depending on time. [12][14]

Within the class of inverse problems, it is the subclass of indirect measurement problems that characterize the nature of inverse problems that arise in applications. Usually measurements only record some indirect aspect of the phenomenon of interest. The inverse problems are difficult because they usually are extremely sensitive to measurement errors [12] especially much more difficult to solve analytically than the direct problem.[13] The difficulties are particularly pronounced as one tries to obtain the maximum of information from the input data.

A formal mathematical model of an inverse problem can be derived with relative ease. However, the process of solving the inverse problem is extremely difficult and the so-called exact solution practically does not exist. For instance, direct measurement of heat flux at the surface of a wall subjected to fire, at the outer surface of a re-entry vehicle or at the inside surface of a combustion chamber is extremely difficult. In such situations, the inverse method of analysis, using transient temperature measurements taken within the medium can be applied for the estimation of such quantities. Therefore, when solving an inverse problem the approximate methods like iterative procedures, regularization techniques, stochastic and system identification methods, methods based on searching an approximate solution in a subspace of the space of solutions (if the one is known), combined techniques or straight numerical methods etc. are used.[12]

2.4 Well-Posedness for Problems for Determining Solution (PDS)

Equations of mathematical physics are drawn from physical problems. In applications, their solutions always refer to particular solutions subjected to certain physical conditions. Such physical conditions are called the *conditions for determining solutions*, the *CDS* for short. The CDS is normally divided into initial conditions and boundary conditions. Finding solutions of equations of mathematical physics subjected to the CDS is called the *problem for determining solutions*, or the *PDS* for short.

The concept of well-posed or correctly posed problems was introduced in "Hadamard, 1923" [15]. Assume that a problem is defined as Au=g, where $u \in U$, $\boldsymbol{g} \in G$, U and G are metric spaces and **A** is an operator so that $\mathbf{A}U \in G$. In general $\boldsymbol{u}$ can be a vector that characterizes a model of phenomenon and $\boldsymbol{g}$ can be the observed attribute of the phenomenon. A well-posed problem must meet the following requirements:

(a) Existence: the solution must exist for any $\mathbf{g} \in \mathbf{G}$. A well-posed PDS must have solutions. In developing equations and their conditions for determining solution (CDS) from physical problems, we must normally make some idealized and simplifying assumptions. Many factors can lead to nonexistence of solutions. Examples are: (1) if the physically-dominant process or mechanisms are not taken into account by the equations, and (2) the CDS is too restrictive or too many. In deriving the PDS, attention should also be given to the surrounding of physical systems as well as to the fundamental physical laws in order to ensure the existence of solutions.[11]

(b) Uniqueness: the solution must be unique. In applications, a PDE is used for describing a unique physical relation. A well-posed PDS should thus have a unique solution. Many factors can contribute to non-uniqueness; a typical example is that the PDS does not have enough CDS.[11]

(c) Stability: the solution must be stable with respect to perturbation on the right hand side, i.e. the operator $\mathbf{A}^{-1}$ must be defined throughout the space G and be continuous. If one of the requirements is not fulfilled the problem is termed as an ill-posed. The CDS comes normally from experimental or field measurements with unavoidable errors. If the solution of a PDS is not stable with respect to the CDS such that a small error in obtaining the CDS can cause significant variations of the solution, then the PDS cannot be used to represent physical reality. A well-posed PDS thus must have a solution that is stable with respect to the CDS, so that the solution varies only a little for a small variation of data in the CDS. In other words, the solution of a well-posed PDS must depend continuously on its CDS. Since a non-homogeneous boundary condition can normally be homogenized by some function transformations, stability often refers to the stability of solutions with respect to the initial conditions. An analytical definition of stability can be made by introducing the size and norm of functions in some function space.

For ill-posed problems the inverse operator $\mathbf{A}^{-1}$ is not continuous in its domain $\mathbf{A}U \in G$ which means that the solution does not depend continuously on the input data $\boldsymbol{g} \in G$. [16][17][12] In general

we can say that the (usually approximate) solution of an ill-posed problem does not necessarily depend continuously on the measured data and the structure of the solution. Moreover, small measurement errors can be the source for unacceptable perturbations in the solution. The best example of the last statement is numerical differentiation of a solution of an inverse problem with noisy input data. Some more interesting remarks on the inverse and ill-posed problems can be found in [18].

If a PDS has a unique and stable solution, it is called a *well-posed PDS*. Otherwise, it is ill-posed. While a well-posed PDS is normally desirable from the point of view of applications, some physical problems do lead to an ill-posed PDS. Therefore the study of ill-posed PDS is also an important branch of partial differential equations. [11]

2.5 Classification of Inverse Heat Conduction Problems

Broadly speaking, heat conduction related inverse problems may be subdivided into the following categories: inverse conduction, inverse convection, inverse radiation and inverse phase change (melting or solidification) problems as well as all combination of them.[19] Here we have adopted classification based on the type of causal characteristics to be estimated:

1. **Boundary value determination inverse problems**: In this kind of inverse problem on a part of a boundary the condition is not known. Instead, in some internal points of the considered body some results of temperature measurements or anticipated values of temperature or heat flux are prescribed. The measured or anticipated values are called internal responses. They can be known on a line or surface inside the considered body or in a discrete set of points.

2. **Initial value determination inverse problems:** In this case an initial condition is not known, i.e. in the condition (2.16) the function T_0 is not known. In order to find the initial temperature distribution a temperature field in the whole considered domain for fixed t>0 has to be known

3. **Material properties determination inverse problems:** Material properties determination makes a wide class of inverse heat conduction problems. The coefficients can depend on spatial coordinates or on temperature. Sometimes dependence on time is considered. In addition to the coefficients also the thermal diffusivity, $a = k/\rho c$, [m/s2] is the one frequently being determined.

4. **Source determination inverse problems:** In the case of source determination, $\dot{Q}_v$ one can identify intensity of the source, its location or both. The problems are considered for steady state and for transient heat conduction. In many cases as an extra condition the

temperature data are given at chosen points of the domain Ω, usually as results of measurements. As an additional condition can be also adopted measured or anticipated temperature and heat flux on a part of the boundary.

5. **Shape determination inverse problems**: In such problems, in contrast to other types of inverse problems, the location and shape of the boundary of the domain of the problem under consideration is unknown. To compensate for this lack of information, more information is provided on the known part of the boundary. In particular, the boundary conditions are over specified on the known part, and the unknown part of the boundary is determined by the imposition of a specific boundary condition(s) on it. [12][9]

3. Solution Approaches

One of the difficulties of ill-posed problems is in defining what is meant by a "solution" because the solution most of the time does not satisfy general conditions of existence, uniqueness, and stability. [10] So difficulties associated with the implementation of inverse analysis should also be recognized. The main difficulty comes from the fact that inverse problems are ill-posed what is mentioned earlier, the solutions are very sensitive to changes in input data resulting from measurement and modelling errors, hence may not be unique. This essentially means that small errors in the data can lead to large errors in the solution. The analysis of ill-posed problems is a growing field in the mathematical and engineering literature during the last 50 years. Extensive reference list going back to the last century has been collected by Dinh Nho Hao and analyzed in his book "Methods for Inverse Heat Conduction Problems".[11][20] An excellent discussion of difficulties encountered in inverse analysis is well documented in the text [10] on inverse heat conduction. A variety of techniques for solving inverse heat conduction problems have been proposed in the literature [2l-26].

In spite of difficulties, there have been varied approaches to the inverse heat conduction problem. Many analytical and semi-analytical approaches have been developed for solving heat conduction problems. Explicit analytical solutions are limited to simple geometries, but are very efficient computationally and are of fundamental importance for investigating basic properties of inverse heat conduction problems. Analytical solutions of the inverse heat conduction problems are very important, because they provide closed form expressions for the heat flux in terms of temperature measurements, give considerable insight into the characteristics of inverse problems, and provide standards of comparison for approximate methods. [12] The use of the adjoin equation approach coupled to the conjugate gradient appears to be very powerful sometimes for solving inverse heat conduction problems. [27]

Many iterative methods for approximate solution of inverse problems are presented in monograph (Bakushinsky & Kokurin, 2004) and numerical methods for solving inverse problems of mathematical physics are presented in monograph (Samarski & Vabishchevich, 2007). In terms of methodology, these have included the exact solution technique [Burggraf (1964)], the inversion of Duhamel's integral [Stolz (1960), Beck (1968)], Laplace transformation techniques [Sparrow (1964), Imber (1972)], the control volume method [Taler (1996)], the use of Helmholtz equation or the theory of potentials method [Grysa (1989)], the finite difference method [Beck (1965, 1970, 1981, 1982), Blackwell (1981), Hensel (1984), Luo & Shih, 2005; Soti et al., (2007)], the finite element approaches [Hore (1977), Bass (1980)], Ling et al. [(2003, 2005)], boundary element method [(Białecki et al., 2006; Onyango et al., 2008)], the digital filtering method [Hills (1986), Hensel (1986)], the eigenvalue reduction method [Tandy (1986)], the group preserving scheme [Chang et al. (2005)], Tikhonov regularization method [Tikhonov

(1977)], Alifannov iterative regularization [Alifannov (1994)], the mollification method [Murio (1993)], the hyperbolic formulation method [Weber (1981)], the conjugate gradient method [Ozisik and Orlande (2000)] and the dynamic programming technique [Trujillo and Busby (1997)], the radial basis functions method [(Kołodziej et al., 2010)], the artificial bee colony method [(Hetmaniok et al., 2010)], the optimal dynamic filtration, [(Guzik & Styrylska, 2002)], the control volume approach [(Taler & Zima, 1999)] and the mesh less methods [(Sladek et al., 2006)] among many others etc. Detail references of these literatures can be found on [12] and [28].

Stolz[29], one of the earliest investigators of the subject, formulated the inverse problem by numerical inversion of an integral equation based on Duhamel's theorem. The equation is a Volterra integral equation of the first kind. Increased accuracy in the determination of the unknown boundary condition requires small time steps. However, if the time step is too small, the method in [29] is unstable. Beck [30] introduced an improved inverse procedure that allows smaller time steps to be used. The basic idea is to utilize interior temperature measurements at a few future time steps. Sparrow et al. [31] used the Laplace transform technique to calculate the unknown boundary condition. In the inverse Laplace transform procedure; there is still a need to invert a Volterra integral equation of the fist kind. Beck [32] introduced a set of criteria to judge the merits of an inverse technique. Two important criteria are that the method must yield accurate results for exact interior temperature data and that the method must be insensitive to measurement errors for the interior temperature. [13] Among various methods, numerical methods have been developed and used for thirty years. However, most of the methods used are more or less heuristic. Despite the great need there seems to be only a few computer programs available for solving practical problems in more than one dimension. [11]

Jarny- Ozisik presented an abstract three-dimensional formulation to solve inverse heat conduction as a general optimization problem by applying the adjoin equation approach coupled to the conjugate gradient algorithm. The formulation consists of the sensitivity problem, the adjoin problem and the gradient equations. A solution algorithm is presented for the estimation of the surface condition (i.e. heat flux or temperature), space dependent thermal conductivity and heat capacity from the knowledge of transient temperature recordings taken within the solid. In this approach, no a priori information is needed about the unknown function to be determined. It is shown that the problems involving a priori information about the unknown function become special cases of this general approach. When the direct problem is linear with the unknown function to be determined, then the functional to be minimized is quadratic convex, the solution is unique and the convergence of the sequence defined by the conjugate gradient method is guaranteed if some regularization is introduced, i.e. if the regularization parameter c is positive. [27] The conjugate gradient method (CG) as an iterative solution algorithm is itself a regularization method where the iteration index plays the role of the regularization parameter. Besides adjoin method, it is worth to discuss in brief the methodology of some techniques that can partially or completely be applied for solving an inverse problem.

3.1 Trefftz method

The method known as "Trefftz method" was firstly presented in 1926 for direct problems, (Trefftz, 1926) [33] where some functions are termed as Trefftz functions or T-functions. In the space of solutions of the considered equation they form a complete set of functions. In the case of any direct (or inverse) problem an approximate solution is assumed to have a form of a linear combination of functions that satisfy the governing partial linear differential equation (without sources). The unknown coefficients of the linear combination are then determined basing on approximate fulfillment the boundary, initial and other conditions (for instance prescribed at chosen points inside the considered body), finally having a form of a system of algebraic equations. [34]

T-functions usually are derived for differential equation in dimensionless form. The equation (2.12) with zero source term and constant material properties can be expressed in dimensionless form as follows:

$$\nabla^2 T(\xi,\tau) = \frac{\partial T(\xi,\tau)}{\partial \tau},\ (\xi,\tau) \in \Omega \times (0,\tau_f) \qquad (3.1)$$

Where, ξ stands for dimensionless spatial location and $\tau = k/\rho c$ denotes dimensionless time (Fourier number). In further consideration we will use notation $x = (x,y,z)$ and t for dimensionless coordinates. For dimensionless heat conduction equation in 1D the set of T-functions read

$$v_n(x,t) = \sum_{k=0}^{n/2} \frac{x^{n-2k}t^k}{(n-2k)!k!} \qquad n = 0,1,2,\ldots\ldots \qquad (3.2)$$

Where, *n/2 = floor*(*n/2*) stands for the greatest previous integer of *n/2*. T-functions in 2D are the products of proper T-functions for the 1D heat conduction equations:

$$V_m(x,y,t) = v_{n-k}(x,t)v_k(y,t) \quad n = 0,1,..;k = 0,1,\ldots..n; m = \frac{n(n+1)}{2}+\mathrm{k} \qquad (3.3)$$

The 3D T-functions are built in a similar way. Consider an inverse problem formulated in dimensionless coordinates as follows:

$$\nabla^2 T = \frac{\partial T}{\partial \tau} \qquad in\ \Omega \times (0,\tau_f] \qquad (3.3)$$

$$T = g_1 \qquad on\ S_D \times (0,\tau_f] \qquad (3.4)$$

$$\frac{\partial T}{\partial n} = g_2 \qquad on\ S_N \times (0,\tau_f] \qquad (3.5)$$

$$\frac{\partial T}{\partial n} + B_i T = B_i g_3 \quad on\ S_R \times (0,\tau_f] \qquad (4.6)$$

$$T = g_4 \qquad on\ S_{int} \times T_{int} \qquad (3.7)$$

$$T = h \qquad on\ \Omega\ for\ t = 0 \qquad (3.8)$$

where S_{int} stands for a set of points inside the considered region, $T_{int} \in (0,\tau_f)$ is a set of moments of time, the functions g_i, i=1,2,3,4 and *h* are of proper class of differentiability in the domains in which they are determined and $S_D \cup S_N \cup S_R = \partial\Omega, B_i = h_c i/k$ denotes the Biot number (dimensionless heat transfer coefficient) and *l* stands for characteristic length. The sets

S_{int} and T_{int} can be continuous (in the case of anticipated or smoothed or described by continuous functions input data) or discrete. Assume that g_1 in not known and g_4 describes results of measurements on $S_{int} \times T_{int}$. An approximate solution of the problem is expressed as a linear combination of the T-functions

$$\boldsymbol{T \approx u \sum_{k=1}^{K} \alpha_k \theta_k} \qquad \textbf{(3.9)}$$

with θ_k standing for T-functions. The objective functional can be written down as

$$\begin{aligned} \boldsymbol{I(u)} = & \int_{S_N \times (0,\tau_f)} \left(\frac{\partial u}{\partial n} - g_2\right)^2 dSdt \\ & + \int_{S_R \times (0,\tau_f)} \left(\frac{\partial u}{\partial n} + B_i u - B_i g_3\right)^2 dSdt \\ & + \int_{S_{int} \times T_{int}} (u - g_4)^2 dSdt + \int_{\Omega} (u - h)^2 d\Omega \end{aligned} \qquad \textbf{(3.10)}$$

In the contrary to the formula (15), the integral containing residuals of the governing equation fulfilling, $\iint_{\Omega \times (0,\tau_f)} ((\nabla^2 - \partial/\partial t)u)^2 d\Omega dt$ does not appear here because u, as a linear combination of T-functions, satisfies the equation (3.5). Minimization of the functional ***I(u)*** (being in fact a function of K unknown coefficients $\alpha_1 \ldots . \alpha_k$ leads to a system of K algebraic equations for the unknowns. The solution of this system leads to an approximate solution, (3.9), of the considered problem. Hence, for $(\boldsymbol{x}, t) \subset S_D \times (0, \tau_f)$ one obtains approximate form of the functions g_1.

It is worth to mention that approximate solution of the considered problem can also be obtained in the case when, for instance, the function *h* is unknown. In the formula (3.9) the last term is then omitted, but the minimization of the functional ***I(u)*** can be done. The final result has physical meaning, because the approximate solution (3.9) consists of functions satisfying the governing partial differential equation. The greater the number of T-functions in (3.9), the better the approximation of the solutions takes place. However, with increasing *K*, conditioning of the algebraic system of equation that results from minimization of ***I(u)*** can become worse. Therefore, the set S_{int} has to be chosen very carefully. Since the system of algebraic equations for the whole domain may be ill-conditioned, a finite element method with the T-functions as base functions is often used to solve the problem.

3.2 Fundamental solution method

The fundamental solution method, like the Trefftz method, is useful to approximate the solution of multidimensional inverse problems under arbitrary geometry. The method uses the fundamental solution of the corresponding heat equation to generate a basis for approximating the solution of the problem. The solvability of the linear system depends on the non-singularity

of the matrix *A*, which is still an open research problem. Fundamental solution method belongs to the family of Trefftz method. Both methods, frequently lead to ill-conditioned system of algebraic equation. To solve the system of equations, different techniques are used. One of the most powerful among them, namely Tikhonov regularization technique, is briefly presented in the further parts of this chapter. A number of regularization methods have been developed for solving this kind of ill-conditioning problem. [35][36] Most of time, this regularization method is very useful in many cases.

4.3 Tikhonov regularization method

Tikhonov introduced what he called the regularization method to reduce the sensitivity of ill-posed problems to measurement errors.[10] This is perhaps the most common and well known of regularization schemes.[38] Instead of looking directly for a solution for an ill-posed problem $A\tilde{\lambda} = \tilde{b}$ we consider a minimum of a functional

$$J[\tilde{\lambda}] = \left\|A\tilde{\lambda} - \tilde{b}\right\|^2 + \alpha^2\left\|\tilde{\lambda} - \tilde{\lambda}_0\right\|^2 \qquad (3.11)$$

with $\tilde{\lambda}_0$ being a known vector, $\|\cdot\|$ denotes the Euclidean norm, and α^2 is called the regularization parameter. The necessary condition of minimum of the functional (3.11) leads to the following system of equation:

$$A^T(A\tilde{\lambda} - \tilde{b}) + \alpha^2(\tilde{\lambda} - \tilde{\lambda}_0) = 0 \qquad (3.12)$$

Hence

$$\tilde{\lambda} = (A^TA + \alpha^2 I)^{-1}(A^T\tilde{b} + \alpha^2\tilde{\lambda}_0)$$

Taking into account (34) after transformation one obtains the following form of the functional *J*:

$$\begin{aligned} J[\tilde{\lambda}] &= \left\|W\Sigma W^T\tilde{\lambda} - WW^T\tilde{b}\right\|^2 + \alpha^2\left\|VV^T(\tilde{\lambda} - \tilde{\lambda}_0)\right\|^2 \\ &= \|W(\Sigma y - c)\|^2 + \alpha^2\|V(y - y_0)\|^2 \\ &= \|\Sigma y - c\|^2 + \alpha^2\|y - y_0\|^2 = J[y] \end{aligned} \qquad (3.13)$$

Where, $y = V^T\tilde{\lambda}$, $y_0 = V^T\tilde{\lambda}_0$, $c = W^T\tilde{b}$ and the use has been made from the properties $WW^T = V^TV = I_N$. Minimization of the functional $J[y]$ leads to the following vector equation:

$$\Sigma^T(\Sigma y - c) + \alpha^2(y - y_0) = 0 \text{ or } (\Sigma^T\Sigma y + \alpha^2 y) = (\Sigma^T c + \alpha^2 y_0) \qquad (3.14)$$

Hence

$$y_i = \frac{\sigma_i}{\sigma_i^2+\alpha^2}c_i + \frac{\alpha^2}{\sigma_i^2+\alpha^2}y_{0i}, \quad i = 1, 2, \ldots, N \text{ or} \qquad (3.15)$$

$$\tilde{\lambda} = \sum_{i=1}^{N}\left(\frac{\sigma_i}{\sigma_i^2+\alpha^2}W_i^T\tilde{b}v_i + \frac{\alpha^2}{\sigma_i^2+\alpha^2}\tilde{\lambda}_0\right) \qquad (3.16)$$

If $\tilde{\lambda}_0 = 0$ the Tikhonov regularized solution for equation $A\tilde{\lambda} = \tilde{b}$ based on singular value decomposition of the $N\times N$ matrix *A* can be expressed as

$$\tilde{\lambda}_\alpha = \sum_{i=1}^{N} \frac{\sigma_i}{\sigma_i^2 + \alpha^2} W_i^T \tilde{b} v_i \qquad (3.17)$$

The determination of a suitable value of the regularization parameter α^2 is crucial and is still under intensive research. Recently the L-curve criterion is frequently used to choose a good regularization parameter. Define a curve L by

$$L = \left\{ \left(log\left(\left\| \tilde{\lambda}_\alpha \right\|^2 \right), log\left(\left\| A\tilde{\lambda}_\alpha - \tilde{b} \right\|^2 \right) \right) \right\} \qquad (3.17)$$

A suitable regularization parameter α^2 is the one near the "corner" of the L-curve. [35][36]

3.4 The Levenberg-Marquardt method

The Levenberg-Marquardt method, originally devised for application to nonlinear parameter estimation problems, has also been successfully applied to the solution of linear ill-conditioned problems. The detail can be found in [12] and [19]

3.5 Kalman filter method

Inverse problems can be regarded as a case of system identification problems. System identification has enjoyed outstanding attention as a research subject. Among a variety of methods successfully applied to them, the Kalman filter, [16][38][39] is particularly suitable for some inverse problems. The Kalman filter is an efficient recursive filter that estimates the state of a dynamic system from a series of incomplete and noisy measurements. The technique is simple and efficient, takes explicit measurement uncertainty incrementally (recursively), and can also take into account *a priori* information, if any.

3.5 FEM method

The FEM leads to promising results when T-functions are used as shape functions. Application of the T-functions as base functions of FEM to solving the inverse heat conduction problem was reported in [34]. Even the condition of temperature continuity in nodes may be weakened. Three different versions of the FEM with T-functions (FEMT) are considered in solving inverse heat conduction problems: (a) FEMT with the condition of continuity of temperature in the common nodes of elements, (b) no temperature continuity at any point between elements and (c) nodeless FEMT.

3.6 Principle of Superposition

A partial differential equation (PDE) is said to be *linear* if it is linear in the unknown function and all its derivatives. An equation which is not linear is called a *nonlinear* equation. A nonlinear equation is said to be *quasi-linear* if it is linear in all highest-ordered derivatives of the unknown function. When there is any constant or any function of independent variables exists in the PDE then the equation is said to be homogeneous, the definition of homogeneity is only for linear PDE.

Linear equations have several remarkable properties that are very useful in finding their solutions. One such property is the **principle of superposition**. In physics and systems theory, the **superposition principle**, also known as **superposition property**, states that, for all linear systems, the net response at a given place and time caused by two or more stimuli is the sum of the responses which would have been caused by each stimulus individually. So, if input *A* produces response *X* and input *B* produces response *Y* then input (*A* + *B*) produces response (*X* + *Y*).

Mathematically, a function to be linear must satisfy two requirements, additivity (sometimes implied by the word superposition) and homogeneity of degree 1 (scalar multiplication). For any scalar "a", two requirements are defined as:

$$F(x_1 + x_2 + \ldots..) = F(x_1) + F(x_2) + \ldots\ldots \quad \textit{Superposition} \quad (3.18)$$

$$F(ax) = a\,F(x) \quad \textit{Homogeneity} \quad (3.19)$$

Combined two requirements can be presented as: $F(ax_1 + bx_2 + \ldots..) = aF(x_1) + bF(x_2) + \ldots\ldots$, right side of this equation is sometimes called a linear combination of functions, $F(x)$.

The importance of linear systems is that they are easier to analyze mathematically; there is a large body of mathematical techniques, frequency domain linear transform methods such as Fourier, Laplace transforms, and linear operator theory, that is applicable. Because physical systems are generally only approximately linear, the superposition principle is only an approximation of the true physical behavior.

The superposition principle applies to *any* linear system, including algebraic equations, linear differential equations, and systems of equations of those forms. The stimuli and responses could be numbers, functions, vectors, vector fields, time-varying signals, or any other object which satisfies certain axioms. Note that when vectors or vector fields are involved, a superposition is interpreted as a vector sum.

By writing a very general stimulus (in a linear system) as the superposition of stimuli of a specific, simple form, often the response becomes easier to compute. For example, in Fourier analysis, the stimulus is written as the superposition of infinitely many sinusoids. Due to the superposition principle, each of these sinusoids can be analyzed separately, and its individual response can be computed. (The response is itself a sinusoid, with the same frequency as the stimulus, but generally a different amplitude and phase.) According to the superposition principle, the response to the original stimulus is the sum (or integral) of all the individual sinusoidal responses. [40]

Mathematically, it is a superposition to express the solution in a form of series. Note that the PDS indeed satisfies the conditions for applying the principle of superposition. The most general homogeneous or non-homogeneous second-order linear partial differential equation in *n* independent variables may respectively be written in the form:

$$Lu = 0 \qquad \text{and} \qquad Lu = f$$

With the context of linear PDE the following properties can be readily shown and are collectively called the *principle of superposition*, which is important in finding solutions of linear equations.

Property 1. A linear combination of two solutions of a homogeneous equation is also a solution of the equation. That is $L(c_1 u_1+c_2 u_2) = 0$, if $Lu_1 = 0$ and $Lu_2 = 0$. Here c_1 and c_2 are arbitrary constants.

Property 2. Let a sequence of functions $\{u_i\}$, $i = 1,2, \cdots$ be solutions of a homogeneous equation $Lu = 0$, and $u = \sum_{i=1}^{\infty} c_i u_i$, where c_i ($i = 1,2, \cdots$) are constants be uniformly convergent and twice differentiable with respect to the independent variables $x_1, x_2, \cdots, x_n$ term by term in a domain. Then u is also a solution of the equation, that is:

$$Lu = L(\sum_{i=1}^{\infty} c_i u_i) = 0 \qquad \textbf{(3.20)}$$

Property 3. $\boldsymbol{u = u_1 + u_2}$ is a solution of a nonhomogeneous equation if $\boldsymbol{u_1}$ and $\boldsymbol{u_2}$ are the solutions of the homogeneous and the nonhomogeneous equations, respectively. Therefore, we have

$$Lu = L(\boldsymbol{u_1 + u_2}) = f, \text{ if } L\boldsymbol{u_1} = 0 \text{ and } L\boldsymbol{u_2} = f. \qquad \textbf{(3.21)}$$

It should be remarked that the supplementary conditions must be linear as well in order to apply this principle.[11]

4. Algorithm

As discussed above it is the principle of superposition by which we can approximate temperature response from given temperature or heat flux history. Now we will attempt to represent it by mathematical formula for direct and inverse heat conduction problem.

4.1 Algorithm for Forward Heat conduction from known temperature history

As shown the generalized problem definition before, here the problem consists of governing equation, Dirichlet, Neumann boundary conditions and initial condition as shown below. For simplicity, we are considering outer wall boundary condition adiabatic, although it is mixed in practice. Since previous work considered accessible side as default boundary (adiabatic), we will continue with the same throughout the work. However, the algorithm is generalized irrespective of boundary conditions.

$$\rho c_p \frac{\partial T}{\partial t} = \nabla \cdot (k \nabla T) + \dot{q}_v \,, \quad (x, y, z) \in \Omega \subset R^3, \; t \in (0, t_f] \qquad (4.1)$$

$$T(x, y, z, t) = T_b(x, y, z, t) \quad for \; (x, y, z, t) \in S_D, \; t \in (0, t_f] \qquad (4.2)$$

$$\frac{\partial T(x,y,z,t)}{\partial n} = 0 \quad for \; (x, y, z, t) \in S_N, \; t \in (0, t_f] \qquad (4.3)$$

$$T(x, y, x, 0) = T_o(x, y, x, t) \quad for \; (x, y, z) \in \Omega$$

To represent the algorithm, first we have to subdivide the whole domain Ω into P (where P ϵ N) number of sub domains Ω_n. Sub-domains boundary will be denoted by $\partial\Omega_n$, n= {1,2,3,….P} . Let ***T(r,0)*** is the initial temperature and ***T(r, t) = T(x,y,z, t)*** is the temperature at any point in the domain at any time ***t***, then we can write:

$$T(r,t) = T(x,y,z,t) = T_o(r,0) + \Delta T(r,t) \qquad \ldots\ldots\ldots (4.4)$$

Where, $\Delta T(r, t)$ is the temperature change in ***t*** time. If we can calculate this $\Delta T(r, t)$ correctly then we can calculate *T(r, **t**)* at any point at any time ***t*** from the known temperature boundary condition or known temperature history.

If we discretize the whole time domain *[0,t_f]* into m number of step sizes Δt e.g. $t_f = m * \Delta t$, $m \in N$, $t_1 = \Delta t$, $t_2 = \Delta t$…. etc. are (global) time steps, then we can rewrite the above equation as:

⇨ $T(r,t) = T_o(r,0) + \sum_{i=1}^{m} \Delta T(r, t_i) = T_o(r,0) + \Delta T(r, t_1) + \Delta T(r, t_2) + \ldots\ldots\ldots + \Delta T(r, t_m)$

Or, in simpler form:

⇨ $T(r,t_1) = T_o(r,0) + \Delta T(r, t_1)$

⇨ $T(r, t_2) = T_o(r,0) + \Delta T(r, t_1) + \Delta T(r, t_2) = T(r, t_1) + \Delta T(r, t_2)$

⇨ $T(r, t_3) = T_o(r,0) + \Delta T(r, t_1) + \Delta T(r, t_2) + \Delta T(r, t_3) = T(r, t_2) + \Delta T(r, t_3)$

……

……..

⇨ $T(r,t) = T_o(r,0) + \sum_{i=1}^{m} \Delta T(r, t_i) = T(r,t_{m-1}) + \Delta T(r, t_m)$ ……. (4.5)

Here $\Delta T(r, t_i)$ are temperature changes at i-th global time step at point r.

We will adopt the above formula considering P number of sub domains. To know temperature change at any point in i-th time step $\Delta T(r, t_i)$, we have to know the influence of each sub domain at that point in i-th time step. So we can write:

$$\Delta T(r, t_1) = \Delta T_1^r(r, t_1) + \Delta T_2^r(r, t_1) + \Delta T_3^r(r, t_1) + \ldots\ldots + \Delta T_P^r(r, t_1) \qquad (4.6)$$

Where $\Delta T_i^r(r, t_1)$, $i = \{1,2,\ldots P\}$ are individual responses at point $r = (x,y,z)$ at first time step t_1 for each sub-domain's input temperature change $\Delta T(\partial\Omega_i, t_1)$ at the Dirichlet boundaries.

It is worth mentioning that, for a defined domain with constant thermo physical properties, any change in the input temperature per time step will cause some response to all point of the domain upto certain time until the response gets steady. For consistency in notation, it is necessary to introduce local and global time steps. Let's say, this steady state time is $t_s = \Delta t^* m'$, where $m' \in N$. Now we can clearly express that $t_1, t_2 \ldots t_m$ etc. are global time steps with respect to the time domain whereas $\lambda_1, \lambda_2, \lambda_3 \ldots \ldots\ldots \lambda_{m'}$ are local time steps corresponding to a particular input temperature change in some global time step t_i . Please note that local and global time step size should be equal. Steady local time t_s can be expressed:

$$t_s = (\Delta t)_1 + (\Delta t)_2 + \ldots\ldots + (\Delta t)_{m'} = \lambda_1 + \lambda_2 + \lambda_3 + \ldots\ldots + \lambda_{m'}$$

From now on, $\Delta T_i^r(r, t_1)$ will be denoted more clearly as $\Delta T_i^r(r, \lambda_j\, t_1)$ which would indicate the response at point r at local time step λ_j caused by $\Delta T(\partial\Omega_i, t_1)$, where $\Delta T(\partial\Omega_i, t_1)$ is the known temperature change at first global time step at the i-th sub-domain's boundary $\partial\Omega_i$. So the clear cut distinction in notation is that $\Delta T(\partial\Omega_i, t_1)$ is the input temperature change at 1st time step in i-th sub-domain's boundary, whereas $\Delta T_i^r(r, \lambda_j\, t_1)$ is the i-th temperature response at the λ_j local time step at point r caused by $\Delta T(\partial\Omega_i, t_1)$.

So the equation (6.6) becomes:

$$\Delta T(r, t_1) = \sum_{i=1}^{P} \Delta T_i^r(r, \lambda_1, t_1)$$

For second global time step, calculating $\Delta T(r, t_2)$, we can decompose responses into two parts:

$\Delta T(r, t_2)$ = Responses at 2nd global time step caused by $\Delta T\ (\partial\Omega_i, t_1)$
+ Responses at 2nd global time step caused by $\Delta T\ (\partial\Omega_i, t_2)$

⇨ $$\Delta T(r, t_2) = \sum_{i=1}^{P} \Delta T_i^r\,(r, \lambda_2, t_1) + \sum_{i=1}^{P} \Delta T_i^r\,(r, \lambda_1, t_2)$$

Introducing global and local time step into the equations, we can generalize the equations 6.6 more comprehensible way:

$$\Delta T(r, t_1) = \sum_{i=1}^{P} \Delta T_i^r\,(r, \lambda_1, t_1) = \sum_{i=1}^{P} \Delta T_i^r\,(r, t_1, t_1)$$
$$\Delta T(r, t_2) = \sum_{i=1}^{P} \Delta T_i^r\,(r, \lambda_2, t_1) + \sum_{i=1}^{P} \Delta T_i^r\,(r, \lambda_1, t_2)$$
$$\Delta T\,(r, t_3) = \sum_{i=1}^{P} \Delta T_i^r\,(r, \lambda_3, t_1) + \sum_{i=1}^{P} \Delta T_i^r\,(r, \lambda_2, t_2) + \sum_{i=1}^{P} \Delta T_i^r\,(r, \lambda_1, t_3)$$

……….

…………

$$\Delta T(r, t_m) = \sum_{i=1}^{P} \Delta T_i^r\,(r, \lambda_m, t_1) + \sum_{i=1}^{P} \Delta T_i^r\,(r, \lambda_{m-1}, t_2) + \sum_{i=1}^{P} \Delta T_i^r\,(r, \lambda_{m-2}, t_3) + \ldots\ldots\ldots + \sum_{i=1}^{P} \Delta T_i^r\,(r, \lambda_1, t_m)$$

⇨ $$\Delta T(r, t_m) = \sum_{i=1}^{P} \sum_{j=1}^{m} \Delta T_i^r\,(r, \lambda_{m-j+1}, t_j) = \sum_{j=1}^{m} \sum_{i=1}^{P} \Delta T_i^r\,(r, \lambda_{m-j+1}, t_j) \qquad \textit{(4.7)}$$

If we want to sum all $\Delta T\ (r, t_j)$ terms then the formula will look like**:**

⇨ $$\Delta T(r, t_1) + \Delta T(r, t_2) + \Delta T\,(r, t_3) + \ldots\ldots + \Delta T(r, t_m) = \left\{ \sum_{i=1}^{P} \Delta T_i^r\,(r, \lambda_1, t_1) + \sum_{i=1}^{P} \Delta T_i^r\,(r, \lambda_2, t_1) + \cdots \ldots + \sum_{i=1}^{P} \Delta T_i^r\,(r, \lambda_m, t_1) \right\} + \left\{ \sum_{i=1}^{P} \Delta T_i^r\,(r, \lambda_1, t_2) + \sum_{i=1}^{P} \Delta T_i^r\,(r, \lambda_2, t_2) + \cdots \ldots + \sum_{i=1}^{P} \Delta T_i^r\,(r, \lambda_{m-1}, t_2) \right\} + \left\{ \sum_{i=1}^{P} \Delta T_i^r\,(r, \lambda_1, t_3) + \sum_{i=1}^{P} \Delta T_i^r\,(r, \lambda_2, t_3) \ldots \ldots + \sum_{i=1}^{P} \Delta T_i^r\,(r, \lambda_{m-2}, t_3) \right\} + \ldots\ldots\ldots + \left\{ \sum_{i=1}^{P} \Delta T_i^r\,(r, \lambda_1, t_m) \right\}$$

⇨ $$\sum_{i=1}^{m} \Delta T(r, t_i) = \sum_{i=1}^{P} \sum_{j=1}^{m} \Delta T_i^r \left(r, \lambda_j, t_1\right) + \sum_{i=1}^{P} \sum_{j=1}^{m-1} \Delta T_i^r \left(r, \lambda_j, t_2\right) + \ldots\ldots + \sum_{i=1}^{P} \sum_{j=1}^{1} \Delta T_i^r \left(r, \lambda_j, t_m\right)$$

⇨ $$\sum_{i=1}^{m} \Delta T(r, t_i) = \Delta T(r, t) = \sum_{i=1}^{P} \sum_{j=1}^{m} \sum_{k=1}^{m-j+1} \Delta T_i^r \left(r, \lambda_j, t_k\right)$$

Now we can put to the original equation (6.4) to get $T(r,t)$

⇨ $$T(r,t) = T(r,0) + \Delta T(r, t) = T(r,0) + \sum_{i=1}^{m} \Delta T(r, t_i)$$
$$= T(r,0) + \sum_{i=1}^{P} \sum_{j=1}^{m} \sum_{k=1}^{m-j+1} \Delta T_i^r \left(r, \lambda_j, t_k\right)$$

⇨ $$\boldsymbol{T(r,t) = T(r,0) + \sum_{i=1}^{P}\sum_{i=1}^{m}\sum_{k-1}^{m-j+1}\Delta T_i^r\left(r,\lambda_i,t_k\right)} \quad (\boldsymbol{4}.8)$$

This is the generalized form of the forward algorithm at point r and time t. The most important thing is still missing, how we could determine the second term of the above equation. To get that, let's consider $\Phi_i^r(\boldsymbol{r},t_s)$ is a response function for some reference temperature change ΔT^{ref} for a time step in i-th sub-domain. Thus, considering all sub-domains we will have P number of reference temperature response functions. Now we can easily calculate the second term of the equation (6.8) by scaling (comparing) with reference temperature response function$\Phi_i^r(\boldsymbol{r},\mathrm{t_s})$ e.g. for each step of the algorithm implementation we should know the corresponding reference response $\Delta\Phi_i^r(\mathrm{r},\lambda_\mathrm{j})$ then we will scale the real case with this reference response. This will look like:

$$\sum_{i=1}^{P}\sum_{j=1}^{m}\sum_{k=1}^{m-j+1}\Delta\boldsymbol{T}_i^r\left(\boldsymbol{r},\boldsymbol{\lambda}_j,\boldsymbol{t}_k\right) = \sum_{i=1}^{P}\sum_{j=1}^{m}\sum_{k=1}^{m-j+1}\boldsymbol{C}_i(\Delta\boldsymbol{T}(\boldsymbol{\partial\Omega}_i,\boldsymbol{t}_k))\,.\,\Delta\Phi_i^r(\boldsymbol{r},\boldsymbol{\lambda}_j)$$

Where $\boldsymbol{C}_i(\Delta\boldsymbol{T}(\boldsymbol{\partial\Omega}_i,\boldsymbol{t}_k))$ is the scaling factor that depends on the temperature change in i-th sub-domain during k-th time step $\Delta\boldsymbol{T}(\boldsymbol{\partial\Omega}_i,\boldsymbol{t}_k)$. So the equation 6.8 becomes:

$$\boldsymbol{T(r,t) = T(r,0) + \sum_{i=1}^{P}\sum_{i=1}^{m}\sum_{k-1}^{m-j+1}\boldsymbol{C}_i(\Delta\boldsymbol{T}(\boldsymbol{\partial\Omega}_i,\boldsymbol{t}_k))\,.\,\Delta\Phi_i^r(\boldsymbol{r},\boldsymbol{\lambda}_i)} \quad (\boldsymbol{4}.9)$$

The next task would be to define reference temperature change in a step ΔT^{ref} and its response functions $\Phi_i^r(\boldsymbol{r},t_s)$ either analytically (if possible) or numerically.

4.2 Defining $\Delta\boldsymbol{T}^{ref}$ and Evaluating $\Phi_i^r(\boldsymbol{r},\boldsymbol{t_s})$

The purpose of defining a reference temperature difference $\Delta\boldsymbol{T}^{ref}$ as per fixed time step is to maintain homogeneity or *scaling*. This means we are looking for such a $\Delta\boldsymbol{T}^{ref}$ and its response $\boldsymbol{\Phi}_i^r(\boldsymbol{r},\boldsymbol{t_s})$ so that we can scale any temperature change per time step at the inaccessible boundary and hence we can determine the responses as well by comparing (scaling) with reference responses $\boldsymbol{\Phi}_i^r(\boldsymbol{r},\boldsymbol{t_s})$. We can simply decompose prescribed temperature history possibly in many ways; of course, discrete counterparts should be scalable with the reference temperature difference $\Delta\boldsymbol{T}^{ref}$. In this regard, two types of such $\Delta\boldsymbol{T}^{ref}$ have been discussed in previous works, one is ***slope type*** and another one is ***impulse type***. Both are shown in the respective following figures.

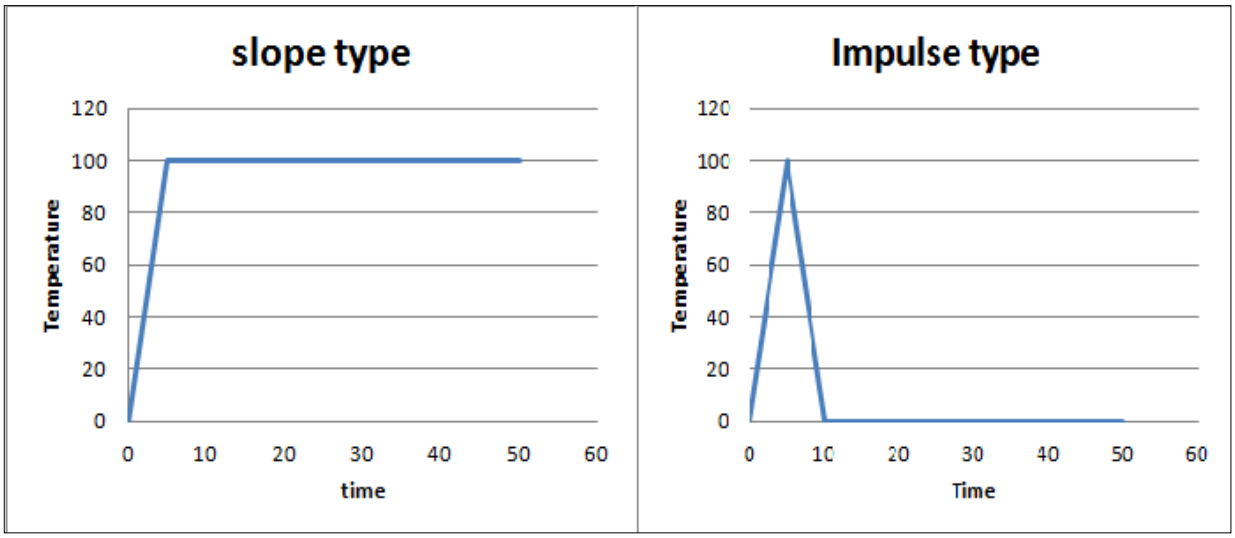


Fig-4.1: Slope and impulse type reference temperature change

What will be the response of slope type at the outer or accessible boundary for a 1D case is also shown in the figure- 6.2. Linearizing and decomposition of a temperature profile according to impulse type is also given (figure-6.3). It can be characterized as upward (Λ) or downward (V) impulse type. For scaling any one can be taken as reference, nevertheless, upward one would be a better choice in terms of sign consideration.

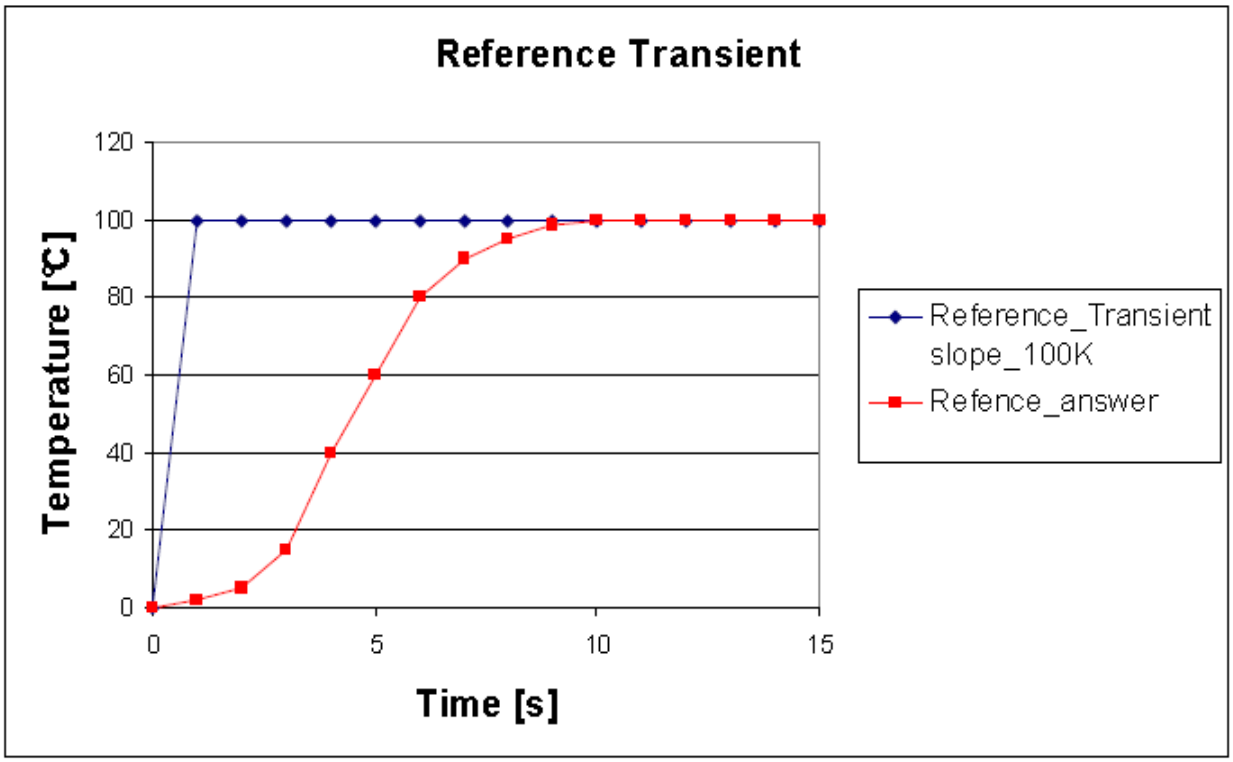


Fig.4.2: Slope type reference temperature profile and its typical response for a 2-D pipe[42]

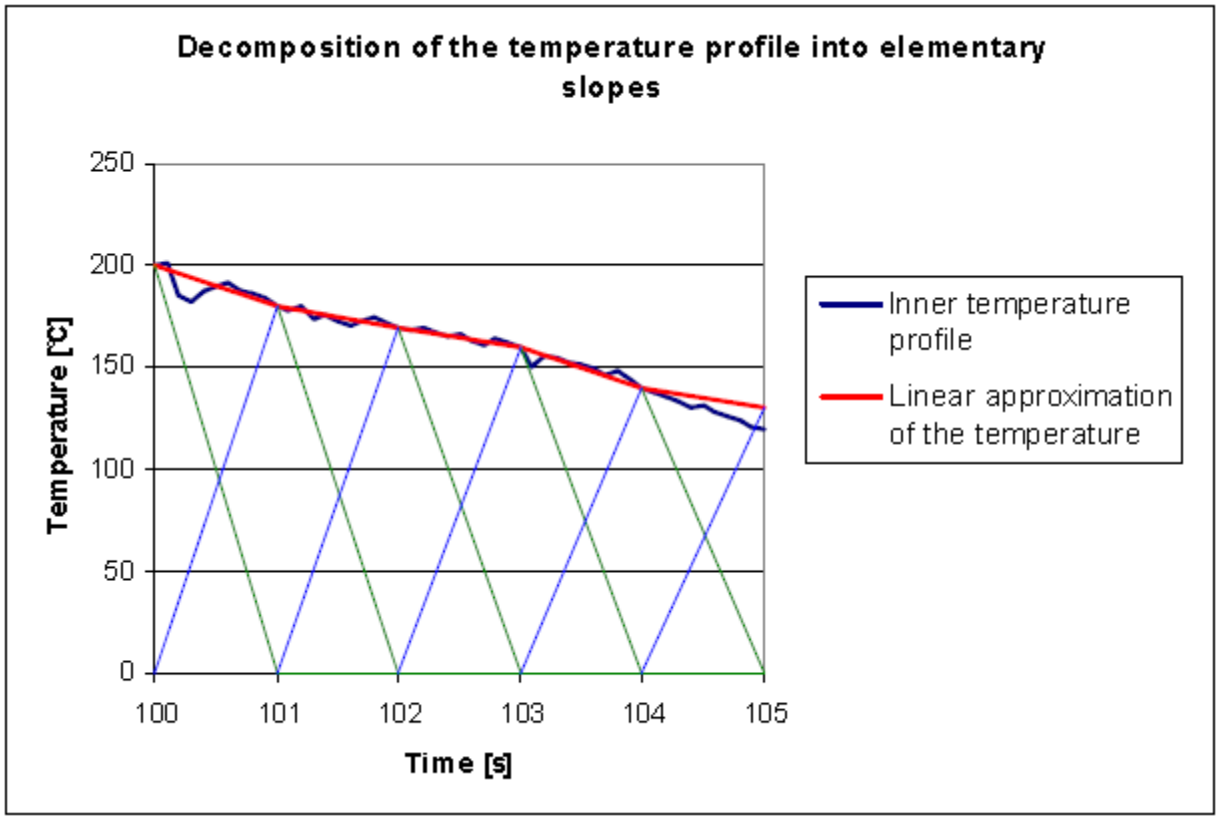


Fig.4.3: Decomposition of impulse type reference temperature[42]

To check the scalability, different inputs have been tested in the previous work in both type of slop and impulse, both confirmed that they are scalable. In order to review the proportionality of both inputs, ANSYS's simulations at 50°C, 100°C and 200°C with different wall thickness were run. Figure 6.4 shows that the outer temperatures are the same and they follow linear proportionality relation in both cases.

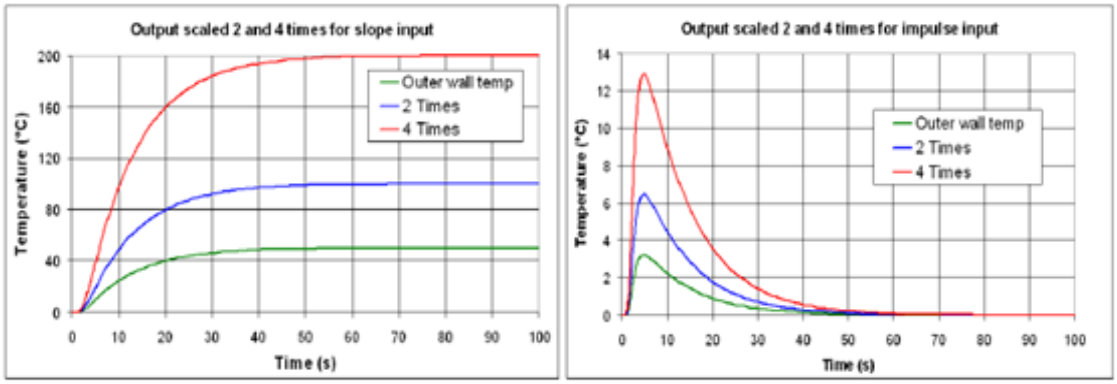


Fig.-4.4: Showing scalability in slope and impulse type ΔT^{ref} [3]

From the figure it is clear that impulse type ΔT^{ref} used two sharp slope changes while slope type changes once. Sharp slope changing in the impulse type is not favorable for bigger time step as it has a tendency to reduce the accuracy what is also shown in the previous work. Most

importantly, slope type ΔT^{ref} ensures that it can be accommodated very easily with the principle of superposition while impulse type does not provide such flexibility. Moreover, the slope type offers more consistency to the algorithm and flexibility in programming, because it is a smooth function. The response of any temperature change per time step can be scaled either increasing or decreasing profile, not with both tendencies like for impulse case. In addition, the coupling of the temperature module with other modules, for instance stress or fatigue calculation module, is easier with the slope input than the impulse input.[3][43]

It is worth to mention that we have to apply the defined $\Delta \boldsymbol{T^{ref}}$ at each sub-domin separately and we will store the responses for each sub-domain. During the time of applying $\Delta \boldsymbol{T^{ref}}$ in a sub-domain it is necessary to keep zero temperature over the other neighboring sub-domains boundary surfaces.

4.2.1 FHCP: Algorithm Summary and Block Diagram

Once we have known thermo-physical properties ρ, c, k, defined domain Ω, and the temperature history at the boundary or Dirichlet boundary condition $\boldsymbol{T(r,t)|_{\partial\Omega}}$ then we can proceed for implementing forward algorithm to know temperature history at any point in the domain especially at the accessible boundary surfaces. Step by step representation of the complete algorithm is given below.

Step-1: Choose a suitable time ***step*** $\Delta \boldsymbol{t}$, and reference transient temperature $\Delta \boldsymbol{T^{ref}}$

Step-2: Subdivide the domain into P number of sub-domains $\boldsymbol{\Omega_n,\ n=\{1,2,\ldots..P\},\ P\epsilon\ N}$

Step-3: Apply $\Delta \boldsymbol{T^{ref}}$ on each of $\boldsymbol{\Omega_n}$ and compute the responses. Store the response history into a matrix as per defined $\Delta \boldsymbol{t}$ of concerned points where the temperature to be calculated. Thus get $\boldsymbol{P}$ number of matrices $\boldsymbol{R_1^{m*n}, R_2^{m*n}, \ldots\ldots, R_P^{m*n}}$ for all sub-domains. Each matrix will contain $\boldsymbol{m*n}$ entries, where **m** is the total number of global time steps and $\boldsymbol{n}$ is the number of concerned points.

Step-4: Assemble the matrices into a single matrix where number of rows will be $\boldsymbol{m}$ and number of columns will be $\boldsymbol{n*n}$. In other words, if individual response matrices are $\boldsymbol{R_1^{m*n}, R_2^{m*n}\ldots. R_P^{m*n}}$ then assembled matrix will be $\boldsymbol{R^{m*nn} = R_1^{m*n} \cup R_2^{m*n} \cup \ldots\ldots \cup R_P^{m*n}}$

Step-5: Linearize $\boldsymbol{T(r,t)|_{\partial\Omega}}$ for all sub-domain $\boldsymbol{\Omega_n}$ according to $\Delta \boldsymbol{t}$ and store into another matrix $\boldsymbol{U_{in}^{m*n}}$ that will be treated as input temperature matrix for forward problem.

Step-6: Compute temperature change $\Delta T(r,t_i)$ any point r and at t_i time step utilizing U_{in}^{m*n} and R^{m*nn} with the help of equation 6.9

Step-7: Compute needed temperature at point r, $T(r, t_i) = T(r, t_{i-1}) + \Delta T(r, t_i)$

Step-8: Compute temperature for all required points

Let's the reference temperature ΔT^{ref} and its response function $\Phi_i^r(x, y, z, t_s)$ is computed numerically or by any simulation environment (like ANSYS), then the above steps can be represented as a simple block diagram as shown below:

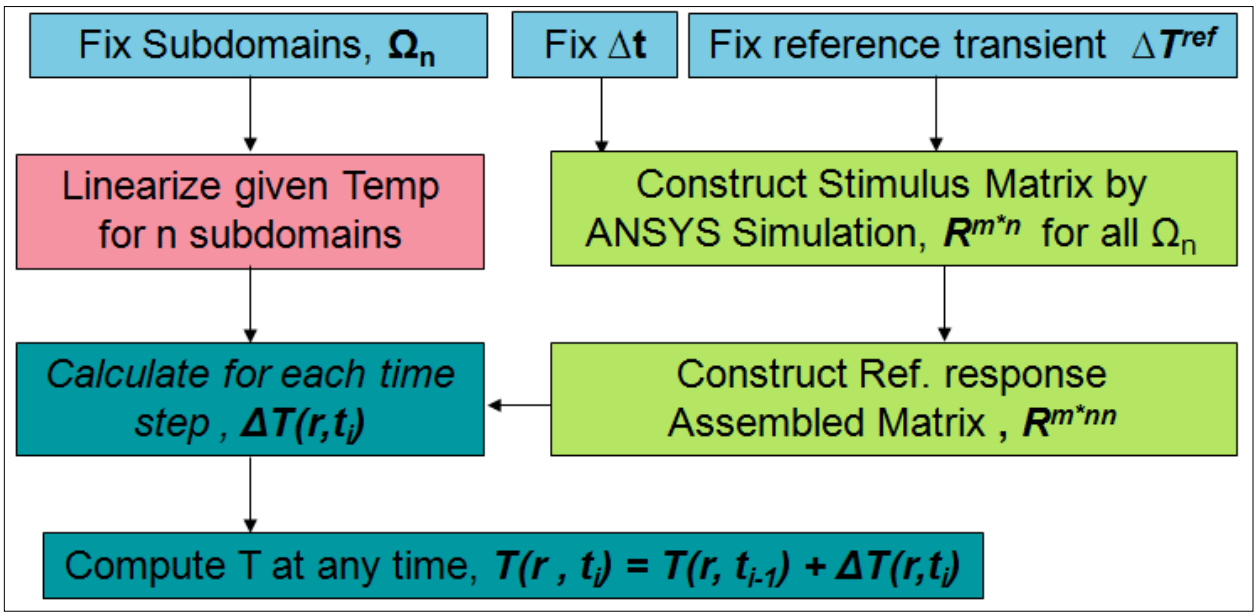


Figure 4.5: Block diagram of the forward algorithm

4.2.2 FHCP: Graphical Representation in 1D Case

Let us assume fluid of uniform temperature is flowing through the pipe, domain Ω is small block from pipe wall with small width and height, hence the heat conduction problem can be represented as one dimensional problem as shown in the figure 6.5. Simplified problem definition with prescribed value or temperature history at the Dirichlet boundary ($\partial\Omega_D = S_D$) in one boundary surface and normal gradient of the variable or Neumann boundary condition in other two surfaces are given by $T(x,t)=T_b(x,t)$ and $\frac{\partial T(x,t)}{\partial n} = 0$ respectively, where $T_b(x,t)$ is temperature distribution during t as shown in the following figure 6.6 and $\frac{\partial T(x,t)}{\partial n} = 0$ is used to indicate the insulated condition of two walls. As mentioned before, number 4 surface is with default boundary (adiabatic). Let the final time is $t_f = 6$ sec and the initial temperature at all point

on the domain is $T_o(x) = 100$. From four boundary surfaces, 1 is active and 2, 3 and 4 are insulated ($\partial\Omega_N = S_N$). Heat flux through the insulated wall is zero e.g $\frac{\partial T(x,t)}{\partial n} = 0$ at three insulated surfaces S_N. So the simplified problem definition can be given by the set of four equations as:

$$\rho c_p \frac{\partial T}{\partial t} = \nabla \cdot (k \nabla T), \quad (x) \in \Omega \subset R, \quad t \in (0, t_f]$$

$$T(x,t) = T_b(x,t) \quad for \ (x,t) \in S_D, \quad t \in (0, t_f]$$

$$-k \frac{\partial T(x,t)}{\partial n} = 0 \quad for \ (x,t) \in S_N, \quad t \in (0, t_f]$$

$$T(x,0) = T_o(x) \quad for \ (x) \in \Omega$$

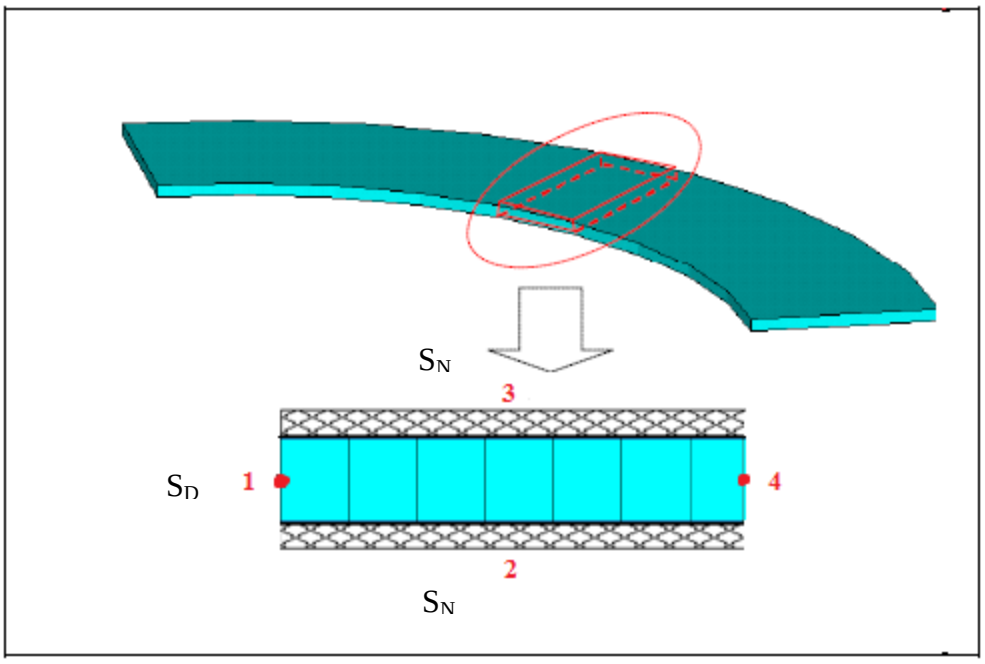


Fig.-4.6: Converting a heat flow problem through the pipe wall into 1 dimensional [43]

First task is to linearize $T_b(x,t)$ according to time step size. Let say time stem $\Delta t = 1$ sec. So we can represent accordingly as $[T_b(x,t)] = [100, 50, 70, 30, 30, 30, 30]^T$ is also shown in the graph. Next we will decompose the linearized $T_b(x,t)$ in each time step according to slope type reference temperature change from the initial temperature 100^o. Let's check whether this decomposition maintains superposition. To test this, we can select any time, let at t=3, ΔT's are decomposed temperature profile.

$$T_b(x,3) = 100 + \Delta T_1(x) + \Delta T_2(x) + \Delta T_3(x) = 100 + (50-100) + (120-100) + (60-100) = 30$$

This calculation shows that the sum of decomposed temperature at time t=3 is 30, on the other hand, from the given profile of $T_b(x,t)$ we see that $T_b(x,3)= 30$. So this approach fully maintains the additivity or superposition. Decomposed profiles are also shown in the following graph 6.7.

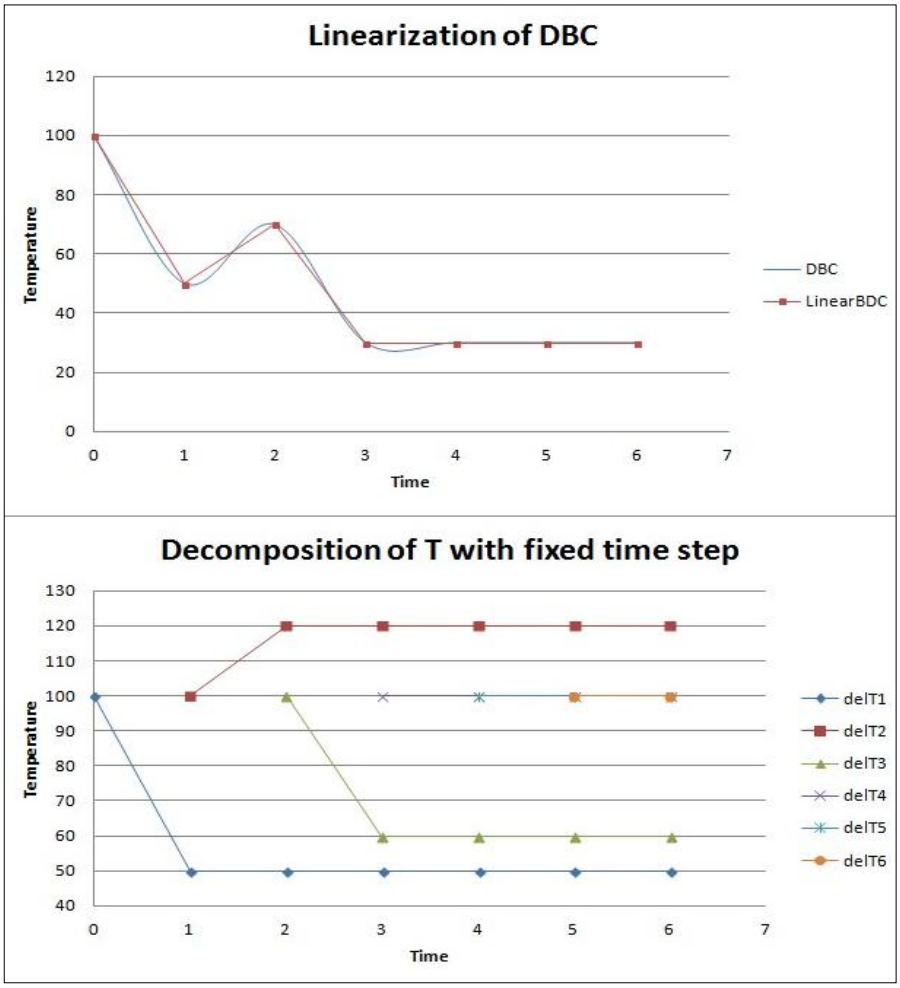
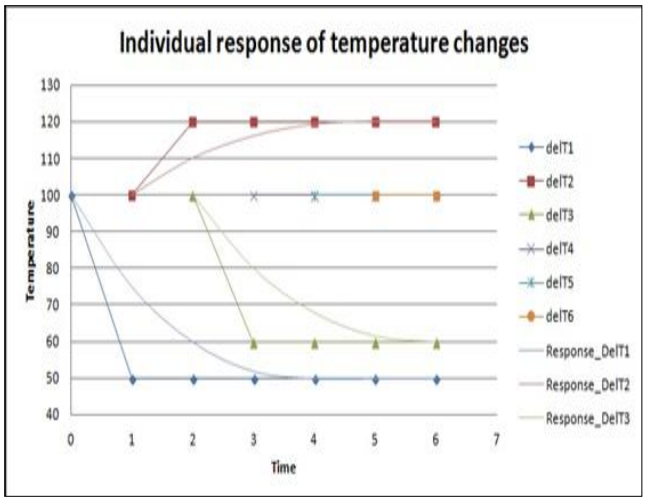


Fig. 4.7, 4.8: given temperature profile at the Dirichlet boundary (4.7) and its decomposed version (4.8)

Our next task is to calculate responses at the outer surface for each decomposed temperature by scaling with standard reference response $\boldsymbol{\Phi}^r(\boldsymbol{r},\boldsymbol{t_s})$ **.** Let the typical scaled response profiles of ΔT after scaling at each time step are like the response curves as shown in the graph 4.8. Now again if we start super-positioning of responses from initial temperature 100^o at each time steps then we could end up with the cumulative response of the given $T_b(x,t)$. Total response (sum of individual responses) is also shown in the graph after superimposing [Figure 4.9 and 4.10]. For example at time t = 3 sec, temperature, $T(x, 3) = T(x,0) + \Delta T_1(x,3) + \Delta T_2(x,3) + \Delta T_3(x,3) = 100+(52-100) + (116-100) + (80-100) = 100 – 48 +16-20 = 48$, which is evident from the figure 4.10.



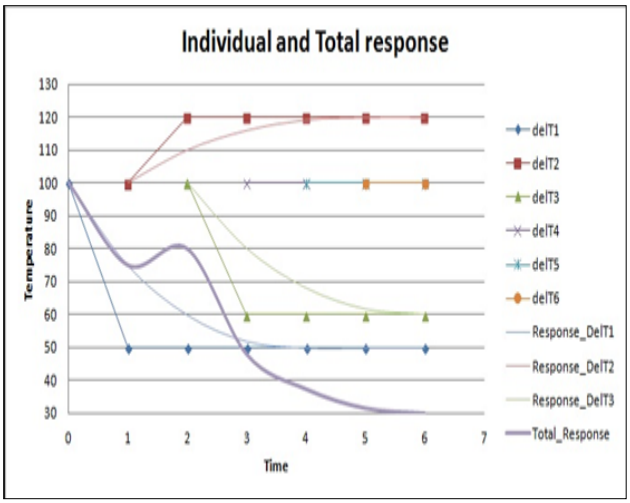


Fig. 4.9, 4.10: Typical responses of each decomposed temperature change (4.9) and cumulative response (4.10)

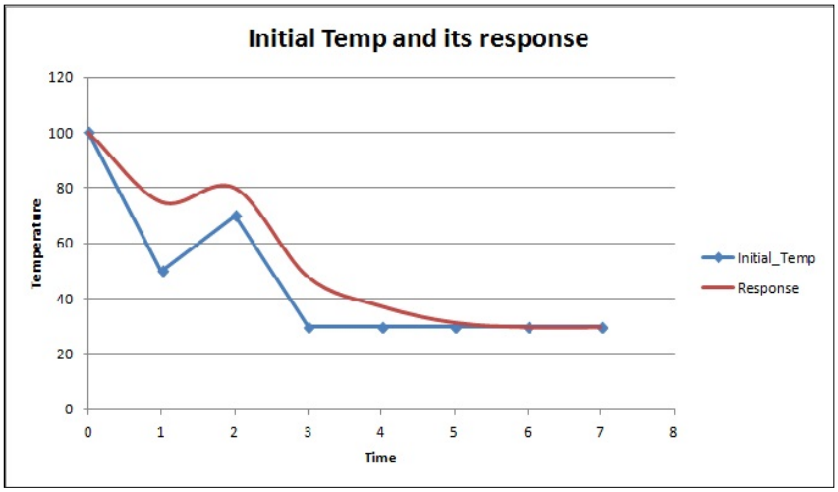


Fig. 4.11: Dirichlet temperature history at the inner wall and its response at the outer wall

4.3 Inverse Heat conduction algorithm from known temperature history

In the inverse case, for simplicity, we will consider only outer (accessible) and inner (inaccessible) walls. So for an inverse heat conduction algorithm with known temperature history in the accessible part of the domain, the main task is to form a system of linear equation and solving it in each time step. To do this, at first we will calculate temperature change in the outside or accessible part at each global time and e.g. $[\Delta \boldsymbol{T}(\boldsymbol{r_i}, \boldsymbol{t_k})]$**,** where k = {1, 2 …. m} and i = {1,2,3….. P}. Then we will separate the net response at each time step $[\Delta \boldsymbol{T^r}(\boldsymbol{r_i}, \boldsymbol{\lambda_1}, \boldsymbol{t_k})]$ from the calculated $[\Delta \boldsymbol{T}(\boldsymbol{r_i}, \boldsymbol{t_k})]$**.** To separate the net response we have to apply forward algorithm in each global time step and store in a matrix. Our objective is to find $\Delta \boldsymbol{T}(\boldsymbol{\partial \Omega_i}, \boldsymbol{t_k})$ in the inaccessible boundary by knowing it's net response in the accessible part or accessible boundary at first local time step. By knowing $\Delta \boldsymbol{T}(\boldsymbol{\partial \Omega_i}, \boldsymbol{t_k})$ at each time step we could build temperature

history in the inaccessible sub-domain's boundary. To comprehend the whole algorithm we will proceed step by step.

4.3.1 Computing temperatures at inaccessible boundary after first global time step

First we will concentrate on the first global time step to know what happened at the inaccessible boundary. Like forward algorithm, we will consider P points from sub-domains boundaries $\partial\Omega_i$, and center point in each $\partial\Omega_i$. In the previous section we have seen that a direct or forward heat conduction problem (FCHP) can be summarized for a particular point ***(x,y,z)*** or ***r*** on the domain by:

$$T(x,y,z,t) = T_o(x,y,z,0) + \sum_{i=1}^{P}\sum_{j=1}^{m}\sum_{k=1}^{m-j+1} C_i(\Delta T(\partial\Omega_i, t_k)) \,.\, \Delta\Phi_i^r(x,y,z,\lambda_j)$$

$$\Rightarrow T(r,t) = T(r,0) + \sum_{i=1}^{P}\sum_{j=1}^{m}\sum_{k=1}^{m-j+1} C_i(\Delta T(\partial\Omega_i, t_k)) \,.\, \Delta\Phi_i^r(r,\lambda_j) \qquad (4.9)$$

Considering ***r*** points in the accessible boundary and considering only first global time step, m =1, j=1 and m-j+1=1, using this values we can rewrite the equation as:

$$T(r, t_1) = T(r, t_0) + \sum_{i=1}^{P} C_i(\Delta T(\partial\Omega_i, t_1))\,\Delta\Phi_i^r(r,\lambda_1)$$

$$\Rightarrow T(r, t_1) = T(r, t_0) + C_1(\Delta T(\partial\Omega_1, t_1)).\Delta\Phi_1^r(r,\lambda_1) + C_2(\Delta T(\partial\Omega_2, t_1)).\Delta\Phi_2^r(r,\lambda_1) + \ldots\ldots + C_P(\Delta T(\partial\Omega_P, t_1))\,\Delta\Phi_P^r(r,\lambda_1)$$

Here scaling factor $C_i(\Delta T(\partial\Omega_i, t_1))$ is actually a factor of $\frac{\Delta T(\partial\Omega_i, t_1)}{\Delta T_{ref}}$ for slope type ΔT^{ref}, replacing this factor and, for ease of notation, putting $\Delta T(\partial\Omega_i, t_1)$ = temperature change at first global time step and at i-th sub-domain = $\Delta T_{1,i}$, we get the above equation:

$$\Rightarrow T(r, t_1) - T(r, t_0) = \Delta T_{1,1} * \frac{\Delta\Phi_1^r(r,\lambda_1)}{\Delta T_{ref}} + \Delta T_{1,2} * \frac{\Delta\Phi_2^r(r,\lambda_1)}{\Delta T_{ref}} + \ldots\ldots + \Delta T_{1,P} * \frac{\Delta\Phi_P^r(r,\lambda_1)}{\Delta T_{ref}}$$

Considering known values $T(r, t_1) - T(r, t_0) = \Delta T(r, t_1)$ we get the equation as:

$$\Rightarrow \Delta T(r, t_1) = \Delta T_{1,1} * \frac{\Delta\Phi_1^r(r,\lambda_1)}{\Delta T_{ref}} + \Delta T_{1,2} * \frac{\Delta\Phi_2^r(r,\lambda_1)}{\Delta T_{ref}} + \ldots\ldots + \Delta T_{1,P} * \frac{\Delta\Phi_P^r(r,\lambda_1)}{\Delta T_{ref}} \qquad (4.10)$$

This equation is the basis of inverse heat conduction algorithm for known temperature history. We can rewrite the equation for P number of different points as:

$$\Rightarrow \Delta T(r_1, t_1) = \Delta T_{1,1} * \frac{\Delta\Phi_1^r(r_1, \lambda_1)}{\Delta T_{ref}} + \Delta T_{1,2} * \frac{\Delta\Phi_2^r(r_1, \lambda_1)}{\Delta T_{ref}} + \ldots\ldots + \Delta T_{1,P} * \frac{\Delta\Phi_P^r(r_1, \lambda_1)}{\Delta T_{ref}}$$

$$\Rightarrow \Delta T(r_2, t_1) = \Delta T_{1,1} * \frac{\Delta\Phi_1^r(r_2, \lambda_1)}{\Delta T_{ref}} + \Delta T_{1,2} * \frac{\Delta\Phi_2^r(r_2, \lambda_1)}{\Delta T_{ref}} + \ldots\ldots + \Delta T_{1,P} * \frac{\Delta\Phi_P^r(r_2, \lambda_1)}{\Delta T_{ref}}$$

$$\Rightarrow \Delta T(r_3, \quad t_1) = \Delta T_{1,1} * \frac{\Delta\Phi_1^r(r_3, \quad \lambda_1)}{\Delta T_{ref}} + \Delta T_{1,2} * \frac{\Delta\Phi_2^r(r_3, \quad \lambda_1)}{\Delta T_{ref}} + \ldots\ldots + \Delta T_{1,P} * \frac{\Delta\Phi_P^r(r_3, \quad \lambda_1)}{\Delta T_{ref}}$$

……

……

……

$$\Rightarrow \Delta T(r_P, \quad t_1) = \Delta T_{1,1} * \frac{\Delta\Phi_1^r(r_P, \quad \lambda_1)}{\Delta T_{ref}} + \Delta T_{1,2} * \frac{\Delta\Phi_2^r(r_P, \quad \lambda_1)}{\Delta T_{ref}} + \ldots\ldots + \Delta T_{1,P} * \frac{\Delta\Phi_P^r(r_P, \quad \lambda_1)}{\Delta T_{ref}}$$

Now if want to represent the above system of equations as matrix-vector form, the system will look like:

$$\begin{bmatrix} \Delta T(r_1,1) \\ \Delta T(r_2,1) \\ \Delta T(r_3,1) \\ \vdots \\ \Delta T(r_P,1) \end{bmatrix} = \frac{1}{\Delta T_{ref}} \begin{bmatrix} \Delta\Phi_1^r(r_1,\lambda_1) & \Delta\Phi_2^r(r_1,\lambda_1) & \Delta\Phi_3^r(r_1,\lambda_1) & \ldots & \ldots & \Delta\Phi_P^r(r_1,\lambda_1) \\ \Delta\Phi_1^r(r_2,\lambda_1) & \Delta\Phi_2^r(r_2,\lambda_1) & \Delta\Phi_3^r(r_2,\lambda_1) & \ldots & \ldots & \Delta\Phi_P^r(r_2,\lambda_1) \\ \Delta\Phi_1^r(r_3,\lambda_1) & \Delta\Phi_2^r(r_3,\lambda_1) & \Delta\Phi_3^r(r_3,\lambda_1) & \ldots & \ldots & \Delta\Phi_P^r(r_3,\lambda_1) \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ \Delta\Phi_1^r(r_P,\lambda_1) & \Delta\Phi_2^r(r_P,\lambda_1) & \Delta\Phi_3^r(r_P,\lambda_1) & \ldots & \ldots & \Delta\Phi_P^r(r_P,\lambda_1) \end{bmatrix} \begin{bmatrix} \Delta T_{1,1} \\ \Delta T_{1,2} \\ \Delta T_{1,3} \\ \ldots \\ \ldots \\ \Delta T_{1,P} \end{bmatrix} \quad (4.11)$$

In the above linear system of equation only unknown vector is $[\Delta T_{1,1} \quad \Delta T_{1,2} \quad \ldots \quad \ldots \quad \Delta T_{1,P}]^T$ or $[\Delta T_{1,i}]$, depending on the conditioning of the operator matrix, say $A_{ø}$, we can choose a suitable solver and hence can solve this linear system of equation to find unknown temperature $[\Delta T_{1,i}]$.

4.3.2 Computing unknown temperature changes at second time step

At second global time step t_2, we can easily calculate the temperature difference at accessible points, $\Delta T(r, t_2)$ from the known temperature history. To compute required $\Delta T_{2,i}$ *or* $\Delta T_{2,i}(r, t_2)$, e.g. temperature differences for P number of points or temperature difference vector in the inaccessible part at second time step, we need to compute the net response for second time step $\Delta T^r(r, [\Delta T_{2,i}], t_2)$ caused by $[\Delta T_{2,i}]$. Please note that third bracket *"[]"* is used here to indicate a vector. Now we can subdivide the cumulative response at second time step as:

Temperature change at any point r = response caused by $[\Delta T_{2,i}]$ + response caused by $[\Delta T_{1,i}]$

$$\Rightarrow \quad \Delta T(r, t_2) = \Delta T^r(r, [\Delta T_{2,i}], t_2) + \Delta T^r(r, [\Delta T_{1,i}], t_2)$$

$$\Rightarrow \quad \Delta T(r, t_2) = \Delta T^r(r, [\Delta T_{2,i}], \lambda_1) + \Delta T^r(r, [\Delta T_{1,i}], \lambda_2)$$

$$\Rightarrow \quad \Delta T^r(r, [\Delta T_{2,i}], t_2) = \Delta T(r, t_2) - \Delta T^r(r, [\Delta T_{1,i}], \lambda_2) \quad (4.12)$$

To compute $\Delta T^r(r, [\Delta T_{1,i}], \lambda_2)$, we need to know temperature change vector at first time step in the inaccessible part e,g. $[\Delta T_{1,i}]$, which is already computed in the previous section 6.2.1. Since

$[\Delta \boldsymbol{T}_{1,i}]$ is already computed; now we could also compute its response up to steady state time $\boldsymbol{t_s}$. Computing the responses of $[\Delta \boldsymbol{T}_{1,i}]$ is basically a direct or forward heat conduction problem. We already discussed the algorithm in the section 6.1. So basically we would compute responses in the second time step $\Delta \boldsymbol{T}^{r}(\boldsymbol{r}, [\Delta \boldsymbol{T}_{2,i}], \boldsymbol{\lambda_2})$ due to temperature change in the first time step $[\Delta \boldsymbol{T}_{1,i}]$ by applying forward algorithm. Once we are done with this computation, we could easily compute the net response $\Delta \boldsymbol{T}^{r}(\boldsymbol{r}, [\Delta \boldsymbol{T}_{2,i}], \boldsymbol{\lambda_2})$ caused by$[(\Delta \mathbf{T}_{2,i})]$. To get the unknown vector in the inaccessible part at second global time step e.g. $[(\Delta \mathbf{T}_{2,i})]$, we will have to calculate the whole net response vector$[\Delta \mathbf{T}^{r}(\mathbf{r}, [\Delta \mathbf{T}_{2,i}], \boldsymbol{\lambda_2})]$. Putting this vector $[\Delta \mathbf{T}^{r}(\mathbf{r}, [\Delta \mathbf{T}_{2,i}], \boldsymbol{\lambda_2})]$ in the right side of the matrix vector equation we could solve the system of linear equation to get $[\Delta T_{2,i}]$. So, the Matrix vector equation will look like:

$$\begin{bmatrix} \Delta \boldsymbol{T}(\boldsymbol{r_1}, \boldsymbol{t_2}) - \Delta \boldsymbol{T}^{r}(\boldsymbol{r_1}, [\Delta \boldsymbol{T}_{1,i}], \boldsymbol{\lambda_2}) \\ \Delta \boldsymbol{T}(\boldsymbol{r_2}, \boldsymbol{t_2}) - \Delta \boldsymbol{T}^{r}(\boldsymbol{r_2}, [\Delta \boldsymbol{T}_{1,i}], \boldsymbol{\lambda_2}) \\ \Delta \boldsymbol{T}(\boldsymbol{r_3}, \boldsymbol{t_2}) - \Delta \boldsymbol{T}^{r}(\boldsymbol{r_3}, [\Delta \boldsymbol{T}_{1,i}], \boldsymbol{\lambda_2}) \\ \vdots \\ \Delta \boldsymbol{T}(\boldsymbol{r_P}, \boldsymbol{t_2}) - \Delta \boldsymbol{T}^{r}(\boldsymbol{r_P}, [\Delta \boldsymbol{T}_{1,i}], \boldsymbol{\lambda_2}) \end{bmatrix} =$$

$$\frac{1}{\Delta \boldsymbol{T}_{ref}} \begin{bmatrix} \Delta \Phi_1^r(\boldsymbol{r_1}, \boldsymbol{\lambda_1}) & \Delta \Phi_2^r(\boldsymbol{r_1}, \boldsymbol{\lambda_1}) & \Delta \Phi_3^r(\boldsymbol{r_1}, \boldsymbol{\lambda_1}) & \dots & \dots & \Delta \Phi_P^r(\boldsymbol{r_1}, \boldsymbol{\lambda_1}) \\ \Delta \Phi_1^r(\boldsymbol{r_2}, \boldsymbol{\lambda_1}) & \Delta \Phi_2^r(\boldsymbol{r_2}, \boldsymbol{\lambda_1}) & \Delta \Phi_3^r(\boldsymbol{r_2}, \boldsymbol{\lambda_1}) & \dots & \dots & \Delta \Phi_P^r(\boldsymbol{r_2}, \boldsymbol{\lambda_1}) \\ \Delta \Phi_1^r(\boldsymbol{r_3}, \boldsymbol{\lambda_1}) & \Delta \Phi_2^r(\boldsymbol{r_3}, \boldsymbol{\lambda_1}) & \Delta \Phi_3^r(\boldsymbol{r_3}, \boldsymbol{\lambda_1}) & \dots & \dots & \Delta \Phi_P^r(\boldsymbol{r_3}, \boldsymbol{\lambda_1}) \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \Delta \Phi_1^r(\boldsymbol{r_P}, \boldsymbol{\lambda_1}) & \Delta \Phi_2^r(\boldsymbol{r_P}, \boldsymbol{\lambda_1}) & \Delta \Phi_3^r(\boldsymbol{r_P}, \boldsymbol{\lambda_1}) & \dots & \dots & \Delta \Phi_P^r(\boldsymbol{r_P}, \boldsymbol{\lambda_1}) \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{T}_{2,1} \\ \Delta \boldsymbol{T}_{2,2} \\ \Delta \boldsymbol{T}_{2,3} \\ \dots \\ \dots \\ \Delta \boldsymbol{T}_{2,P} \end{bmatrix}$$

OR,

$$\begin{bmatrix} \Delta \boldsymbol{T}^{r}(\boldsymbol{r_1}, [\Delta \boldsymbol{T}_{2,i}], \boldsymbol{t_2}) \\ \Delta \boldsymbol{T}^{r}(\boldsymbol{r_2}, [\Delta \boldsymbol{T}_{2,i}], \boldsymbol{t_2}) \\ \Delta \boldsymbol{T}^{r}(\boldsymbol{r_3}, [\Delta \boldsymbol{T}_{2,i}], \boldsymbol{t_2}) \\ \vdots \\ \Delta \boldsymbol{T}^{r}(\boldsymbol{r_P}, [\Delta \boldsymbol{T}_{2,i}], \boldsymbol{t_2}) \end{bmatrix}$$

$$= \frac{1}{\Delta \boldsymbol{T}_{ref}} \begin{bmatrix} \Delta \Phi_1^r(\boldsymbol{r_1}, \boldsymbol{\lambda_1}) & \Delta \Phi_2^r(\boldsymbol{r_1}, \boldsymbol{\lambda_1}) & \Delta \Phi_3^r(\boldsymbol{r_1}, \boldsymbol{\lambda_1}) & \dots & \dots & \Delta \Phi_P^r(\boldsymbol{r_1}, \boldsymbol{\lambda_1}) \\ \Delta \Phi_1^r(\boldsymbol{r_2}, \boldsymbol{\lambda_1}) & \Delta \Phi_2^r(\boldsymbol{r_2}, \boldsymbol{\lambda_1}) & \Delta \Phi_3^r(\boldsymbol{r_2}, \boldsymbol{\lambda_1}) & \dots & \dots & \Delta \Phi_P^r(\boldsymbol{r_2}, \boldsymbol{\lambda_1}) \\ \Delta \Phi_1^r(\boldsymbol{r_3}, \boldsymbol{\lambda_1}) & \Delta \Phi_2^r(\boldsymbol{r_3}, \boldsymbol{\lambda_1}) & \Delta \Phi_3^r(\boldsymbol{r_3}, \boldsymbol{\lambda_1}) & \dots & \dots & \Delta \Phi_P^r(\boldsymbol{r_3}, \boldsymbol{\lambda_1}) \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \Delta \Phi_1^r(\boldsymbol{r_P}, \boldsymbol{\lambda_1}) & \Delta \Phi_2^r(\boldsymbol{r_P}, \boldsymbol{\lambda_1}) & \Delta \Phi_3^r(\boldsymbol{r_P}, \boldsymbol{\lambda_1}) & \dots & \dots & \Delta \Phi_P^r(\boldsymbol{r_P}, \boldsymbol{\lambda_1}) \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{T}_{2,1} \\ \Delta \boldsymbol{T}_{2,2} \\ \Delta \boldsymbol{T}_{2,3} \\ \dots \\ \dots \\ \Delta \boldsymbol{T}_{2,P} \end{bmatrix}$$

$$\Rightarrow \quad [\Delta \boldsymbol{T}^{r}(\boldsymbol{r}, [\Delta \boldsymbol{T}_{2,i}], \boldsymbol{t_2})] = \boldsymbol{[A_\Phi]}. [\Delta \boldsymbol{T}_{2,i}] \qquad \textbf{\textit{(4.13)}}$$

Which can be compared with linear system of equation **b = Ax,** or, **Ax=b.** Applying a suitable linear system solver again we can solve the system and get $[\Delta \boldsymbol{T}_{2,i}]$.

4.3.3 Generalizing the equation for any time step

For any time steps we can generalize the net response formula based on second time step equation. We have just seen the net response vector for second time step is:

$$\begin{bmatrix} \Delta \boldsymbol{T}^r(\boldsymbol{r}_1, [\Delta \boldsymbol{T}_{2,i}], \boldsymbol{t}_2) \\ \Delta \boldsymbol{T}^r(\boldsymbol{r}_2, [\Delta \boldsymbol{T}_{2,i}], \boldsymbol{t}_2) \\ \Delta \boldsymbol{T}^r(\boldsymbol{r}_3, [\Delta \boldsymbol{T}_{2,i}], \boldsymbol{t}_2) \\ \vdots \\ \Delta \boldsymbol{T}^r(\boldsymbol{r}_P, [\Delta \boldsymbol{T}_{2,i}], \boldsymbol{t}_2) \end{bmatrix} = \begin{bmatrix} \Delta \boldsymbol{T}(\boldsymbol{r}_1, \boldsymbol{t}_2) - \Delta \boldsymbol{T}^r(\boldsymbol{r}_1, [\Delta \boldsymbol{T}_{1,i}], \boldsymbol{\lambda}_2) \\ \Delta \boldsymbol{T}(\boldsymbol{r}_2, \boldsymbol{t}_2) - \Delta \boldsymbol{T}^r(\boldsymbol{r}_2, [\Delta \boldsymbol{T}_{1,i}], \boldsymbol{\lambda}_2) \\ \Delta \boldsymbol{T}(\boldsymbol{r}_3, \boldsymbol{t}_2) - \Delta \boldsymbol{T}^r(\boldsymbol{r}_3, [\Delta \boldsymbol{T}_{1,i}], \boldsymbol{\lambda}_2) \\ \vdots \\ \Delta \boldsymbol{T}(\boldsymbol{r}_P, \boldsymbol{t}_2) - \Delta \boldsymbol{T}^r(\boldsymbol{r}_P, [\Delta \boldsymbol{T}_{1,i}], \boldsymbol{\lambda}_2) \end{bmatrix}$$

$$\Rightarrow \quad [\Delta \boldsymbol{T}^r(\boldsymbol{r}, [\Delta \boldsymbol{T}_{2,i}], \boldsymbol{t}_2)] = [\Delta \boldsymbol{T}(\boldsymbol{r}, \boldsymbol{t}_2)] - [\Delta \boldsymbol{T}^r([\Delta \boldsymbol{T}_{1,i}], \boldsymbol{\lambda}_2)] \qquad (4.14)$$

Likewise, the net response vector for third time step will look like:

$$\Rightarrow \quad [\Delta \boldsymbol{T}^r(\boldsymbol{r}, [\Delta \boldsymbol{T}_{3,i}]), \boldsymbol{t}_3] = [\Delta \boldsymbol{T}(\boldsymbol{r}, \boldsymbol{t}_3)] - [\Delta \boldsymbol{T}^r(\boldsymbol{r}, [\Delta \boldsymbol{T}_{1,i}], \boldsymbol{\lambda}_3)] - [\Delta \boldsymbol{T}^r([\boldsymbol{r}, \Delta \boldsymbol{T}_{2,i}], \boldsymbol{\lambda}_2)]$$

For forth time step it will be:

$$\Rightarrow \quad [\Delta \boldsymbol{T}^r(\boldsymbol{r}, [\Delta \boldsymbol{T}_{4,i}], \boldsymbol{t}_4)] = [\Delta \boldsymbol{T}(\boldsymbol{r}, \boldsymbol{t}_4)] - [\Delta \boldsymbol{T}^r(\boldsymbol{r}, [\Delta \boldsymbol{T}_{1,i}], \boldsymbol{\lambda}_4)] - [\Delta \boldsymbol{T}^r(\boldsymbol{r}, [\Delta \boldsymbol{T}_{2,i}], \boldsymbol{\lambda}_3)] - [\Delta \boldsymbol{T}^r(\boldsymbol{r}, [\Delta \boldsymbol{T}_{3,i}], \boldsymbol{\lambda}_2)]$$

Similarly for any time step t_k , k = {1, 2, 3….. $t_f/\Delta t$} the above equation can be written as generalized form:

$$\Rightarrow \quad [\Delta \boldsymbol{T}^r(\boldsymbol{r}, [\Delta \boldsymbol{T}_{k,i}], \boldsymbol{t}_k)] = [\Delta \boldsymbol{T}(\boldsymbol{r}, \boldsymbol{t}_k)] - [\Delta \boldsymbol{T}^r(\boldsymbol{r}, [\Delta \boldsymbol{T}_{1,i}], \boldsymbol{\lambda}_k)] - [\Delta \boldsymbol{T}^r(\boldsymbol{r}, [\Delta \boldsymbol{T}_{2,i}], \boldsymbol{\lambda}_{k-1})] - \ldots\ldots - [\Delta \boldsymbol{T}^r([\Delta \boldsymbol{T}_{k-1,i}], \boldsymbol{\lambda}_2)]$$

$$\Rightarrow \quad [\Delta \boldsymbol{T}^r(\boldsymbol{r}, [\Delta \boldsymbol{T}_{k,i}], \boldsymbol{t}_k)] = [\Delta \boldsymbol{T}(\boldsymbol{r}, \boldsymbol{\lambda}_k)] - \sum_{j=1}^{k-1} [\Delta \boldsymbol{T}^r(\boldsymbol{r}, [\Delta \boldsymbol{T}_{j,i}], \boldsymbol{\lambda}_{k-j+1})] \qquad (4.15)$$

Thus inverse heat conduction algorithm for any time step $\boldsymbol{t_k}$ can be generally represented as:

$$\frac{1}{\Delta T_{ref}}\begin{bmatrix} \Delta\Phi_1^r(r_1,\lambda_1) & \Delta\Phi_2^r(r_1,\lambda_1) & \dots & \dots & \dots & \Delta\Phi_P^r(r_1,\lambda_1) \\ \Delta\Phi_1^r(r_2,\lambda_1) & \Delta\Phi_2^r(r_2,\lambda_1) & \dots & \dots & \dots & \Delta\Phi_P^r(r_2,\lambda_1) \\ \Delta\Phi_1^r(r_3,\lambda_1) & \Delta\Phi_2^r(r_3,\lambda_1) & \dots & \dots & \dots & \Delta\Phi_P^r(r_3,\lambda_1) \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \Delta\Phi_1^r(r_P,\lambda_1) & \Delta\Phi_2^r(r_P,\lambda_1) & \dots & \dots & \dots & \Delta\Phi_P^r(r_P,\lambda_1) \end{bmatrix}\begin{bmatrix} \Delta T_{k,1} \\ \Delta T_{k,2} \\ \Delta T_{k,3} \\ \dots \\ \dots \\ \Delta T_{k,P} \end{bmatrix} = \begin{bmatrix} \Delta T^r(r_1,[\Delta T_{k,i}],\ t_k) \\ \Delta T^r(r_2,[\Delta T_{k,i}],\ t_k) \\ \Delta T^r(r_3,[\Delta T_{k,i}],\ t_k) \\ \vdots \\ \Delta T^r(r_P,[\Delta T_{k,i}],\ t_k) \end{bmatrix}$$

…….. ***(4.16)***

4.3.4 IHCP: Algorithm summary and Flow Diagram

Once we have the transient temperature history at the accessible boundary $T(r_{out},t)|_{\partial\Omega}$, sub-domains, time step Δt, reference transient ΔT^{ref} and its response matrix R^{m*nn} (for all sub-domains) then we can proceed for implementing the inverse algorithm to know temperature history at any point in the inside or inaccessible boundaries. Step by step we can represent the complete algorithm as:

Step-1: Calculate temperature differences vector $[\Delta T(r_i, t_1)]$ at first global time step from the known temperature history $T(r_{out},t)|_{\partial\Omega}$

Step-2: Construct the matrix $A_{\emptyset}^{n*n}$ picking the first row of R^{m*nn}. Let the elements of R^{m*nn} are $R_{i,j}$, then the constructed matrix $A_{\emptyset}^{n*n}$ will look like:

$$A_{\emptyset}^{n*n} = \begin{bmatrix} R_{1,1} & R_{1,2} & R_{1,3} & .. & .. & R_{1,n} \\ R_{1,n+1} & R_{1,n+2} & R_{1,n+2} & .. & .. & R_{1,2n} \\ \vdots & \vdots & \vdots & :: & :: & :: \\ .. & .. & .. & .. & .. & .. \\ R_{1,(n-1)n+1} & R_{1,(n-1)n+2} & R_{1,(n-1)n+3} & .. & .. & R_{1,nn} \end{bmatrix} \quad (4.17)$$

Step-3: Construct the system of linear equation with the operator matrix $A_{\emptyset}^{n*n}$, unknown vector $[\Delta T_{k,i}]$ for corresponding global time step k = {t_1, t_2, …. T_m} and with net response vector $[\Delta T^r(r,[\Delta T_{k,i}],\ t_k)]$. For the first global time step, $[\Delta T^r(r,[\Delta T_{k,i}],\ t_k)] = [\Delta T(r, t_1)]$

Step-4: Solve the system with a suitable solver and find the unknown temperature changes $[\Delta T_{k,i}]$ in the inaccessible part.

Step-5: Apply forward algorithm with just computed $[\Delta T_{k,i}]$ as known values, find its responses at the accessible points till steady state time t_s is reached and store the values in a matrix $B^{m'\times P}$. At each time step we will have such a matrix.

Step-6: Compute net response vector for next step[$\Delta \boldsymbol{T}^{r}(\boldsymbol{r}, [\Delta \boldsymbol{T}_{k+1,i}], \boldsymbol{t}_{k+1})$] with the help of all B matrices and go to step-3. If the current global time step is t_m, go to step-7.

Step-7: Compute required temperature history for all required points,

$$[\boldsymbol{T}(\boldsymbol{r}_{in}, \boldsymbol{t}_{k})] = [\boldsymbol{T}(\boldsymbol{r}_{in}, \boldsymbol{t}_{k-1})] + [\boldsymbol{\Delta T}_{k,i}]$$

The above steps can be represented as a simple block diagram as well:

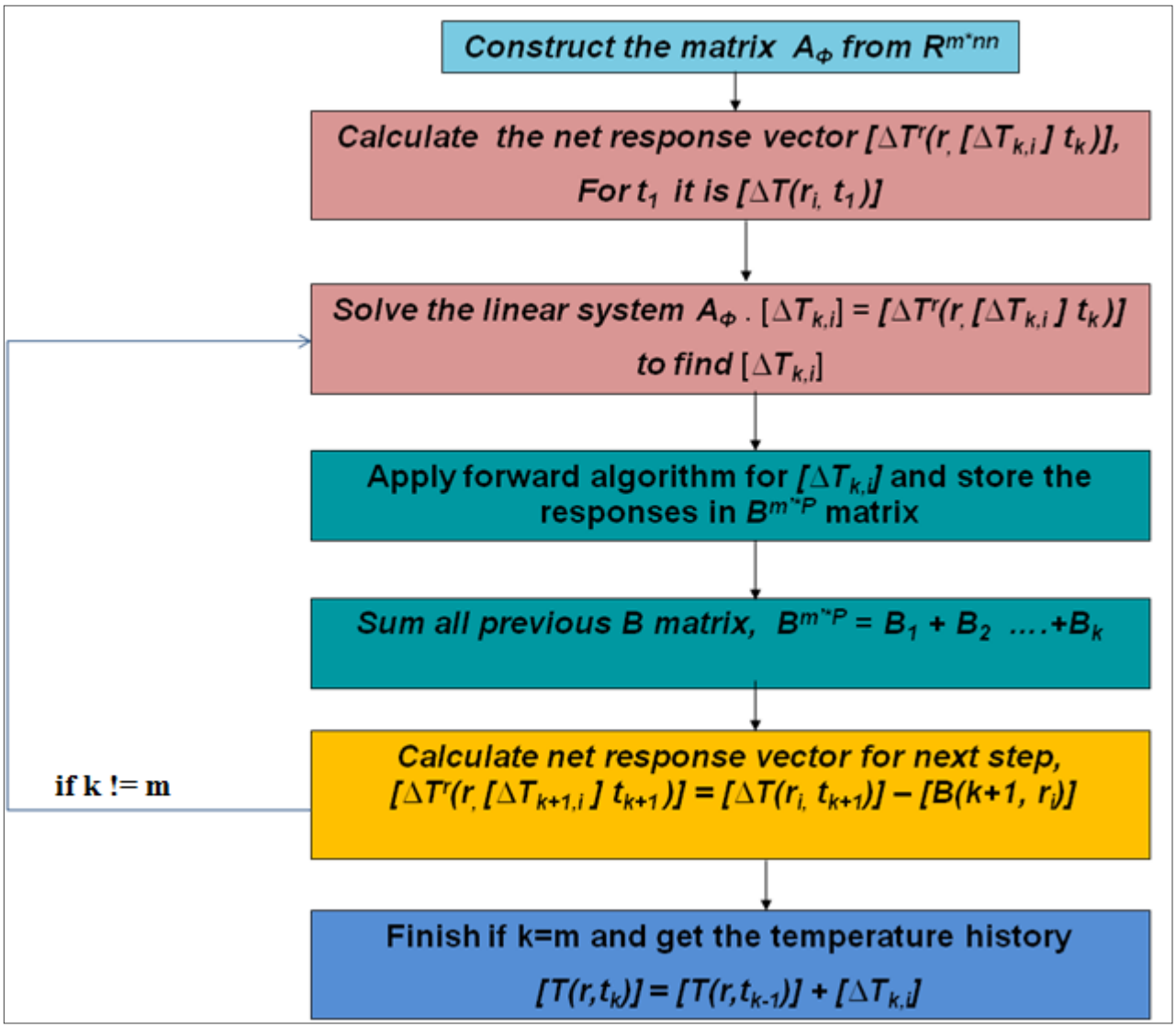


Figure 4.12: Block diagram of Inverse algorithm

4.3.5 IHCP: Graphical Representation in 1D Case

This part is to illustrate inverse heat conduction algorithm in a simple one dimensional case. 1D problem definition is the same as discussed for forward heat conduction algorithm. The measured/supplied temperature difference $[\Delta \boldsymbol{T}(\boldsymbol{r}_i, \boldsymbol{t}_1)]$ in the outside wall or accessible boundary

will be compared with the temperature difference of the reference case. Outside temperature history is shown in the figure 4.13.

If we read the graph 4.13, we could read that $Tout\ (0) = 40°C$, $Tout\ (1) = 40.8°C$, It means that $\Delta Tout\ (t_1) = 0.8$. Reference type is being considered here the same as presented in the example of 1D forward algorithm e.g. slope type reference. For the reference data as shown in the figure-4.14, we can calculate the temperature difference between the two first consecutive times at the outer wall as: $To\ \_\ ref\ (0) = 0°C$, $To\ \_\ ref\ (1) = 2°$, It means: $\Delta To\ \_\ ref\ (t_1) = 2°$

The comparison between $\Delta Tout\ (t_1)$ and $\Delta To\ \_\ ref\ (\lambda_1)$ gives the inner temperature at the time ***t=1*** as per the simple formula below: $\Delta T_{in}(t_1) = \frac{\Delta Tout\ (t_1) * \Delta Tin\ \_\ ref\ (\lambda_1)}{\Delta To\ \_\ ref\ (\lambda 1)} = \frac{0.8*100}{2} = 40$

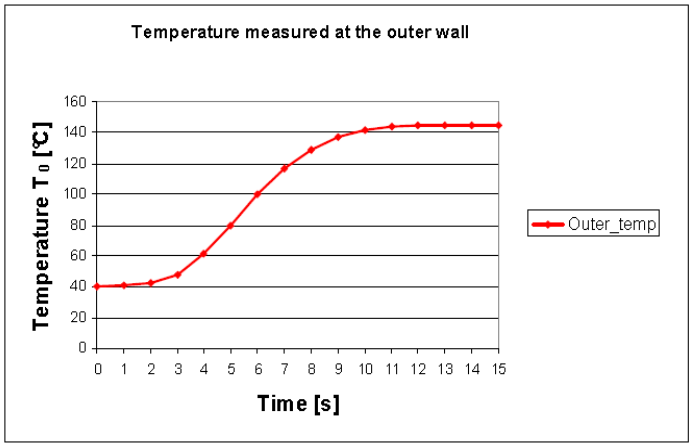


Fig- 4.13: Given outside temperature history [42]

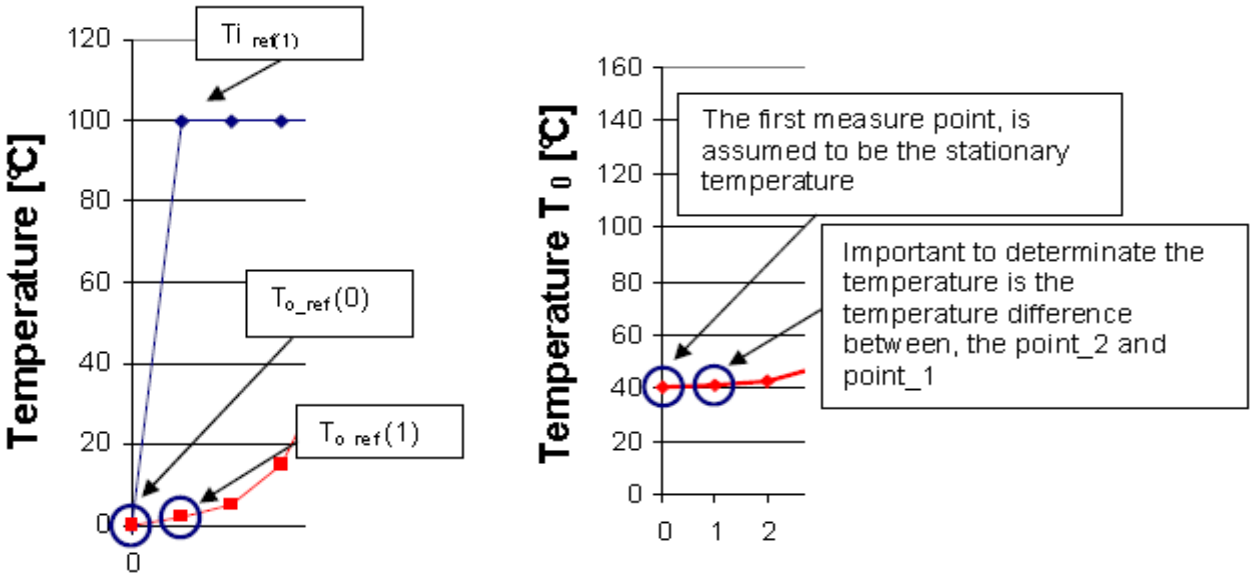


Fig. 4.14, 4.15: Showing ΔT^{ref} and its response, 4.15 is in magnified view [42]

This means that the temperature variation at the inner side between the time t=0 and t=1 is 40°C. There will be some definite contribution of this 40 degree change on the outer wall temperature history, that's why we need to know the influence of this 40 degree what can be computed by applying a forward algorithm. This contribution is also shown by the orange colored line in the figure 4.16.

From the reference data we found that for 100^o change, it results 5^o change in the first two time steps. So by comparison we can say 40^o will cause 2^o change in the outside e.g at t=2. From the known temperature history we see, the outside temperature $T_{out}(t=2) = 42.6$, initial temperature is 40^o, the contribution of $\Delta T_{in}(t_1)$ is 2^o, so the net response at second global time step $\Delta T_{in}^r(t_2) = T_{out}(t=2) - 2 - 40 = 42.6-42 = 0.6$. This 0.6 degree is caused by $\Delta T_{in}(t_2)$. So, we can find the inner wall temperature change in the second global time step is $0.6*100/2 = 30^o$. So $T_{in}(0) = 40$, $T_{in}(1) = 40+40 = 80$ and $T_{in}(2) = 80+30 = 110^o$. For third global time step t_3, again we need to know the contribution of 40^o and 30^o, then we could separate the net response $\Delta T_{in}^r(t_3)$ caused by inner wall temperature change $\Delta T_{in}(t_3)$.

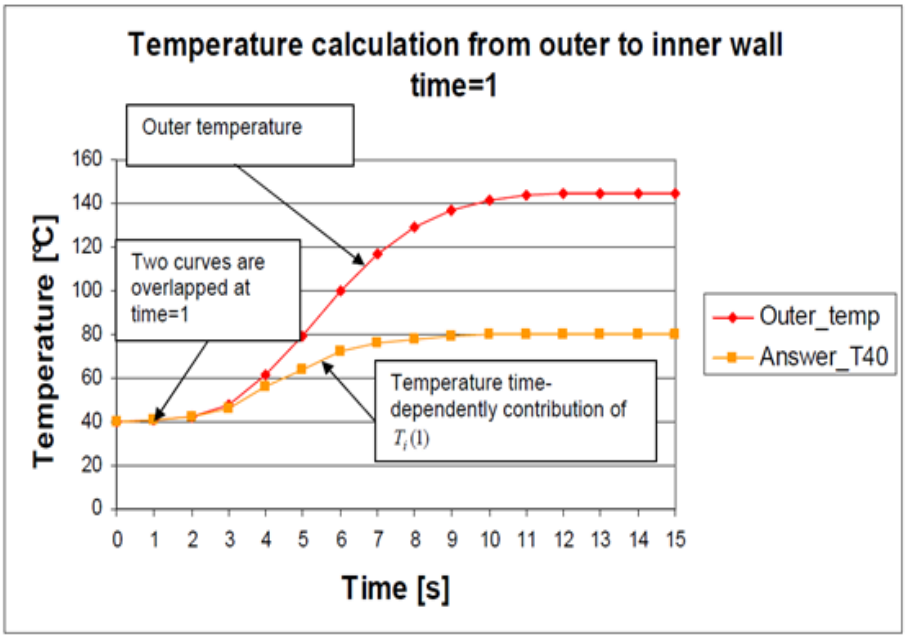


Figure 4.16: Outer wall temperature history and the contribution of $\Delta T_{in}(t_1)$ [42]

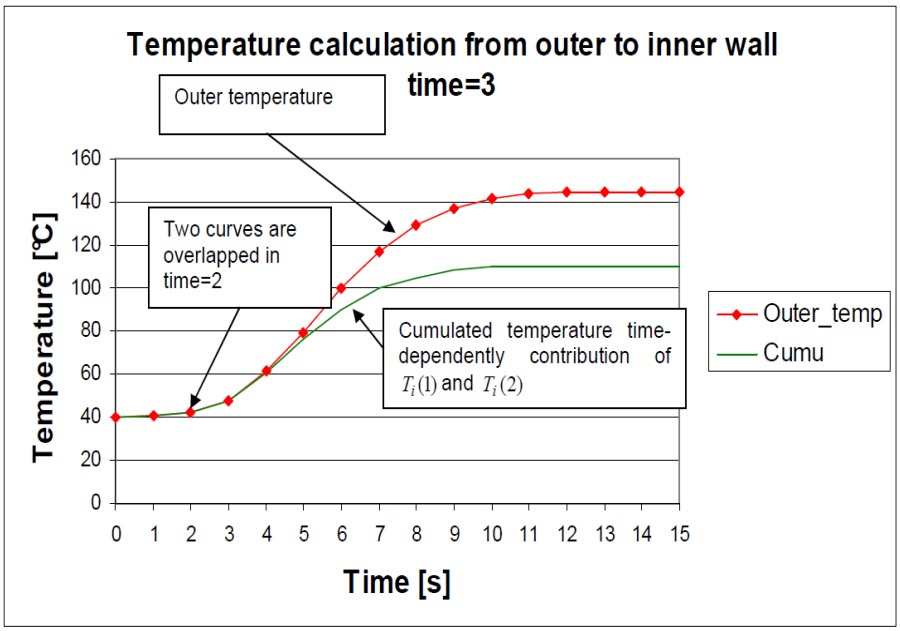


Figure 4.17: Cumulative contribution of $\Delta T_{in}(t_1)$ and $\Delta T_{in}(t_2)$ to the outer wall temperature history [42]

In this way we can proceed to compute the complete inner history step by step. The combined contribution of 40^o and 30^o is computed and shown in the next plot 4.17. It is necessary to reiterate here that, at each step we are calculating temperature change at inaccessible part by a ratio comparison formula $\frac{\Delta T_{in}(t_i)}{\Delta T_{out}(t_i)} = \frac{\Delta Tin\_ref\ (\lambda_1)}{\Delta Tout\_ref\ (\lambda 1)}$, this formula will be changed into a system of linear equation for higher dimension like 2D or 3D.

4.4 Forward/Inverse Heat conduction algorithm from heat flux history

If the parameter of interest in the inaccessible part is heat flux q instead of temperature T, we can use the same principle of superposition and thus the similar algorithm to find unknown heat flux in the accessible boundary. The idea is, change in heat flux $\Delta q(\partial\Omega_i, t_k)$ at the inner boundary surface in each time step will result temperature increase or decrease to the outer or accessible boundary surfaces. So if the temperature history of a body is known in the accessible boundary, then we could compute heat flux in the inaccessible boundary surfaces. Instead of ΔT^{ref} and its responses we will have to select some suitable Δq^{ref} and its responses $\Delta\Phi_i^r(r, \lambda_j)$. Apart from this reference heat flux change, everything will be the same. Thus we can prove that for forward case the equation (6.9) will be like:

$$T(r,t) = T(r,0) + \sum_{i=1}^{P}\sum_{j=1}^{m}\sum_{k=1}^{m-j+1} C_i(\Delta q(\partial\Omega_i, t_k)) \,.\, \Delta\Phi_i^r(r, \lambda_j) \qquad (4.18)$$

Keeping the same meaning of notations used so far we can write the system of equation for inverse case in each global time step to be solved as:

$$\begin{bmatrix} \Delta\Phi_1^r(\boldsymbol{r_1},\boldsymbol{\lambda_1}) & \Delta\Phi_2^r(\boldsymbol{r_1},\boldsymbol{\lambda_1}) & \dots & \dots & \dots & \Delta\Phi_P^r(\boldsymbol{r_1},\boldsymbol{\lambda_1}) \\ \Delta\Phi_1^r(\boldsymbol{r_2},\boldsymbol{\lambda_1}) & \Delta\Phi_2^r(\boldsymbol{r_2},\boldsymbol{\lambda_1}) & \dots & \dots & \dots & \Delta\Phi_P^r(\boldsymbol{r_2},\boldsymbol{\lambda_1}) \\ \Delta\Phi_1^r(\boldsymbol{r_3},\boldsymbol{\lambda_1}) & \Delta\Phi_2^r(\boldsymbol{r_3},\boldsymbol{\lambda_1}) & \dots & \dots & \dots & \Delta\Phi_P^r(\boldsymbol{r_3},\boldsymbol{\lambda_1}) \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \Delta\Phi_1^r(\boldsymbol{r_P},\boldsymbol{\lambda_1}) & \Delta\Phi_2^r(\boldsymbol{r_P},\boldsymbol{\lambda_1}) & \dots & \dots & \dots & \Delta\Phi_P^r(\boldsymbol{r_P},\boldsymbol{\lambda_1}) \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{q_{k,1}} \\ \Delta\boldsymbol{q_{k,2}} \\ \Delta\boldsymbol{q_{k,3}} \\ \dots \\ \Delta\boldsymbol{q_{k,P}} \end{bmatrix} = \frac{\mathbf{1}}{\Delta\boldsymbol{T_{ref}}} \begin{bmatrix} \Delta\boldsymbol{T^r}(\boldsymbol{r_1},[\Delta\boldsymbol{q_{k,i}}],\boldsymbol{t_k}) \\ \Delta\boldsymbol{T^r}(\boldsymbol{r_2},[\Delta\boldsymbol{q_{k,i}}],\boldsymbol{t_k}) \\ \Delta\boldsymbol{T^r}(\boldsymbol{r_3},[\Delta\boldsymbol{q_{k,i}}],\boldsymbol{t_k}) \\ \dots \\ \Delta\boldsymbol{T^r}(\boldsymbol{r_P},[\Delta\boldsymbol{q_{k,i}}],\boldsymbol{t_k}) \end{bmatrix}$$

(4.19)

The same principle can be adopted for mixed type or any other possible boundary condition as well, as long as the governing equation and BC's are linear.

5. ANSYS Simulations for 2D Pipe

To test the algorithm discussed above for 2D pipe, we have to fix reference temperature function and its responses to all subdomains e.g. the ΔT^{ref} should be applied to each innner zones individually and the responses for each case should be recorded. It is already discussed that slope type ΔT^{ref} is a perfect choice to maintain the superposition principle on which the algorithm is based on. Since knowing the responses of ΔT^{ref} is a direct or forward head conduction problem, we can use any simulation software package. After doing all the necessary simulations, we will have to assemble the data into a matrix for further use**.** Our all implementation attempts are limited for 2D pipes of different diameter and thickness. Well known simulations environment ANSYS [Mechanical APDL' (ANSYS Parametric Design Language)] has been used to know responses of ΔT^{ref}. Some typical simulations, taken as reference case, have also been performed to test forward and inverse algorithm pair. Before starting APDL's finite element analysis, we can recapitulate the problem difinition here again with considering only essential and natural boundary condition. For all simulations we will only consider essential boundary condition at inner side and adiabatic (default) one at outer wall.

$$\rho c \frac{\partial T}{\partial t} = \nabla \bullet (k \nabla T), \qquad (x,y) \in \Omega \subset R^2, \Omega = (r_{in}, r_{out}] \times \theta, \ t \in (0, t_f] \qquad (7.1)$$

$$T(x,y,t) = T_b(x,y,t) \ \text{ or } \ T(r_{in}, \theta, t) = T_b(r_{in}, \theta, t), \qquad \text{for } (x,y,t) \in S_D = \partial\Omega_1 \qquad (7.2)$$

$$\frac{\partial T(x,y,t)}{\partial n} = 0 \ or \ \frac{\partial T(r,\theta,t)}{\partial n} = 0, \qquad for \ (x,y,t) \in S_N = \partial\Omega \backslash \partial\Omega_1 \ \ t \in (0, t_f] \qquad (7.3)$$

$$T(x,y,0) = T_0(x,y,0) \ \text{ or } \ T(r,\theta,0) = T_0(r,\theta,0), \qquad \text{for } (x,y) \in \Omega \text{ or } (r,\theta) \in \Omega \qquad (7.3)$$

5.1 ANSYS simulation with diffetrent r_{in}, thickness, ρ, c, k

We already mentioned that we need to apply reference temperature in each subdomain keeping zero degree temperature to other boundary surface of the domain. For 2D pipe, since the left half of the domain is symmetric to the right half, so we will consider only half of the domain. Two representative ANSYS codes/commands will be included in the appendix. Preprocessing and solution sections of all simulations attempted were command based whereas post processing was with graphical user interface (GUI) as GUI provides a great flexibility from different aspects in the interactive mode. In real case, heat transfer occurs from fluid to solid inner wall of the pipe, for which we need to know fluid temperature, film coefficient and heat flux. Since we do not know the fluid temperature, so for simplicity of simulations we will condiser ΔT^{ref} directly on the surface boundary, e.g. instead of considering mixed boundary condition, we will adopt DBC only. Subdomainns have been devided based on uniform circumferential distance, not based on radial distances. Outer wall corresponding points can be considered the locations of thermocouples or the middle points of two thermocouples. We used middle points for most of the cases. In this regard, another important point is zone numbering, which is done from upper to

lower side of the 2D pipe. By two ANSYS commands we can do that. First we can directly apply heat by D (degree of freedom) command or we can use SF (surface force) command with very high film coefficient that will ensures that fluid temperature is the wall temperature. In our problem we tried both approaches in different sumulations. It should be noted that ANSYS environment uses heat conduction and convection together e.g energy transport with convection and diffusin equations which is applicable for solid and liquid as well. Thus the linear problem will be simulated as nonlinear one by using load step and substep concept what will be discussed broadly later. Convecction added equation can be represendted as:

$$\rho c \frac{DT}{Dt} = \nabla \bullet (k \nabla T) \qquad (7.4)$$

$$\Rightarrow \rho c \left\{ \frac{\partial T}{\partial t} + w_x \frac{\partial T}{\partial x} + w_y \frac{\partial T}{\partial y} + w_z \frac{\partial T}{\partial z} \right\} = \nabla \bullet (k \nabla T) = \frac{\partial}{\partial x}\left(k_x \frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(k_y \frac{\partial T}{\partial y}\right) + \frac{\partial}{\partial z}\left(k_z \frac{\partial T}{\partial z}\right)$$

Where D/Dt is with as usual meaning, material time derivative, generally used to describe convection phenomena.

5.1.1 Simulation with six sub-domains (six inner zones or layers) with 10mm thickness

First, **Thermophysical parameters:** Used thermophysical parameters for this simulation are :

Parameter	Unit	Value at room temp.
Inner Dia.	mm	180
Thickness	mm	10
Film coefficient	$W/m^2.k$	10000
Specific Heat	J/g.K	15
Density	kg/m^3	7930
Heat conductivity	W/m.K	47
Material Type	Austenitic Steel	

Model and parameters are so chosen that it closely relates some real piping facilities in the nuclear power plant components.

Second, Element type: ANSYS environment provides some prescribed elements type for a particular type of problem. For example, in 2D thermal and thermo-structural problems, it provides two elements types, namely PLANE55 and PLANE77. Both are from Serendipity family of elements, uses nodes at vertices and at the element boundary. PLANE55 is a 4-points based linear (bilinear) rectangular elemnt while PLANE77 is 8-points based quadratic (biquadratic) element as shown below. So basically both elements approximate the unknown variable by their corresponding polynomials which are also given.

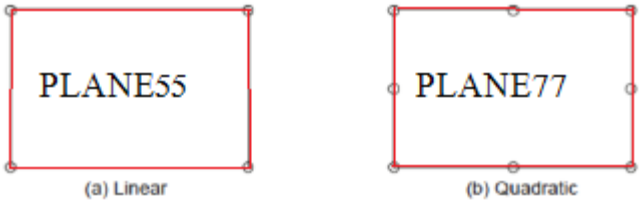


Figure 5.1: Two different elements for thermostructural analysis

Polynomial for PLANE55: $T_h(x,y) = C_1 + C_2x + C_3y + C_4xy$ *(7.5)*

For PLANE55: $T_h(x,y) = C_1 + C_2x + C_3y + C_4xy + C_5x^2 + C_6y^2 + C_7 x^2 y + C_8 xy^2$ *(7.6)*

All constants C's are nodal value dependent. To check the element preference, two different elements have been applied for a fixed problem, it seems 8-pt based elements ensures little bit better accuracy [figure 5.2] and that's why this element has been used in most of the simulations.

Third, Model: In the real case seven thermocouples are installed around the half of the pipe in each fatigue relevant location. A typical subsectioning of the domain, corresponding inner zones stratification and thermocouple locations are shown the figure 5.2. Discretized ANSYS model is also shown in the figure 5.3.

Forth, Time control: Although steady state time t_s, in most cases, is less than 100 seconds for small diameter, so it is sufficient to choose 200 sec. for this simulation. So our reference temperature function ΔT^{ref} will look like:

$$\Delta T^{ref} = \begin{cases} 100t, & for\ 0 \le t \le 1 \\ 100, & for\ 1 \le t \le 200 \end{cases}$$

Load step and substep selections are partly automatic partly manual, it is forced to generate results at each time step e.g at each second to conform with predefined 1 second time step. The obvious reason is to avoid interpolation in time stepping, since interpolation error, even very small, may cause big noise and fluctuation in the inverse problem.

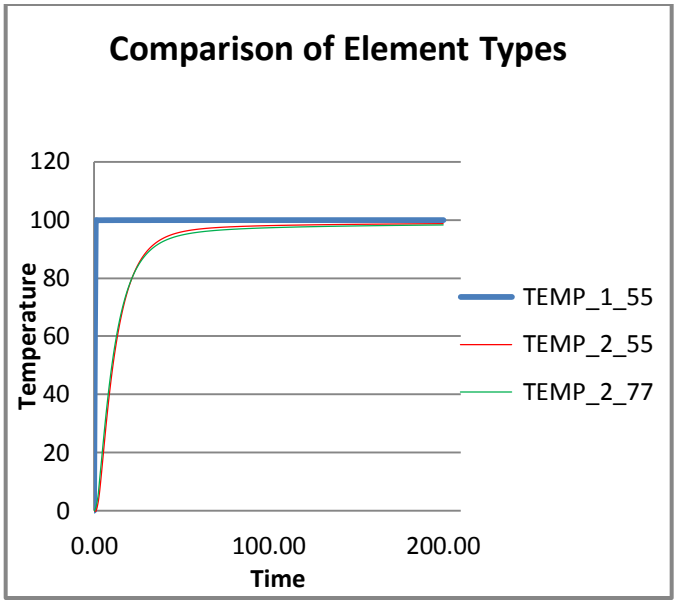


Figure 5.2: coparing two element types

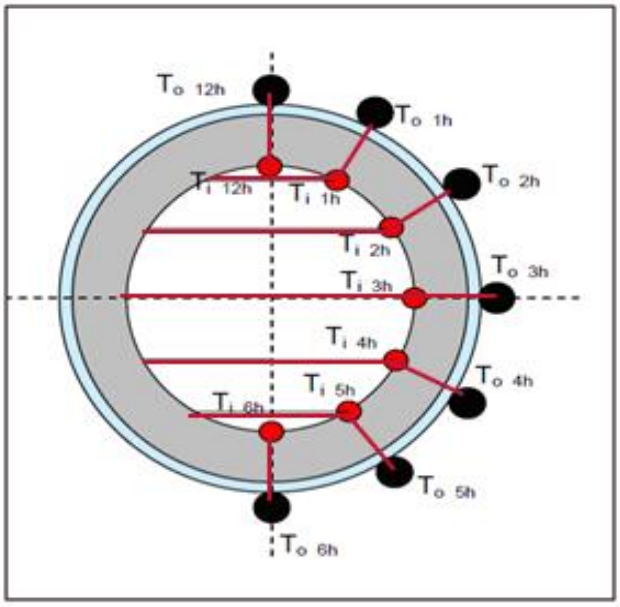


Figure-5.3: Thermocouple locations and subsections to the inner wall

Fifth, Computational grids and solver: Total number of grid points for most simulation is 1152, number of biquadratice element is 188 and the corresponding global matrix is as usually sparse, as for most traditional finite element discretization with structured grid. Regarding solver; sparse, Preconditioned conjugate gradient (PCG) and Incomplete Cholesky conjugate gradient (ICCG) solvers cover 95% of all ANSYS applications. Sparse solver is default in most cases for robustness and efficiency which is mostly sparse direct while it could be iterative as well. The direct one is based on a direct elimination (Cholesky decomposition, LU for unsymmetrical matrices and LDL for symmetrical) of equations. The problem of direct solver is creating lower triangular matrix is memory consuming. That's why ANSYS automatically uses hard drive memory. Detail solver info can be found by the command **BCSOPTION.**

In a nonlinear solution in ANSYS, there are three distinct levels: (1) Load step, (2) Substeps and (3) Equilibrium Iterations. The number of load steps is specified by the user. Different load steps must be used if the loading on the structure changes abruptly. The use of load steps also becomes necessary if the response of the structure at specific points in time is desired. A solution within each load step is obtained by applying the load incrementally in substeps. Within each substep, several equilibrium iterations are performed until convergence is accomplished, after which ANSYS proceeds to the next substeps. As the number of substeps used increases, the accuracy of the solution improves. However this also means that more computational time is being used. ANSYS offers the automatic time stepping feature to optimize the task of obtaining a solution with acceptable accuracy in a reasonable amount to time. The automatic time stepping feature decides on the number and size of substeps within load steps. When using automatic time

stepping, if a solution fails to converge within a substep, the bisection method is activated, which restarts the solution from the last converged substep. [ANSYS tutorial]

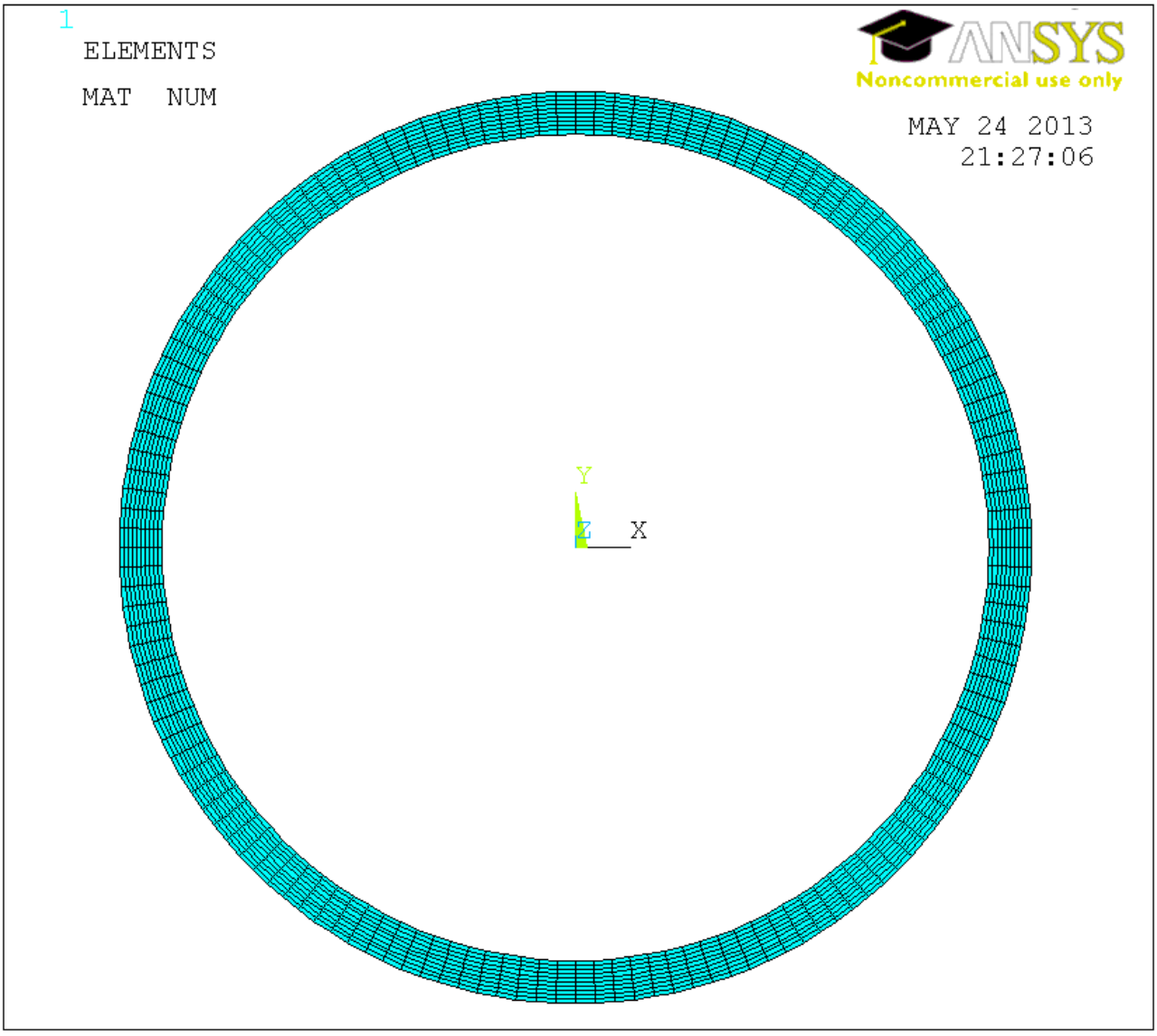


Figure 5.4: Ansys model

Sixth, Response functions: Responses at outer wall due to reference temperature applied individually on each boundary surfaces are shown by figure 5.5, 5.6 and 5.7. Every response function ultimately heads toward steady state after some time. Depending on the wall thickness and diameter it might be 50, 100, 1000 or even more. Figure 5.5 indicates that due to 100 degree temperature rise in the inner surface in zone 1, temperature increase at steady state at the correspong outer surface is about 98.7%, gradually it decreases toward the other outer surfaces. At the mid point of the nearest surfaces, increase is 1.3% . While figure 5.6 indicates, 97.4% rise

is its corresponding outer surfaces and 1.3% rise happened at the mid points of two neighboring outer surfaces. Since the domain is axisymmetric, each 90 degree section of the domain shows symmetry to the nearest 90 degree section. Hence, response function of ΔT^{ref} on $\partial\Omega_1$ will show the similar tendency for response function ΔT^{ref} on $\partial\Omega_6$, only the locations will be flipped, that means 1.3% rise will be observed at the fifth outer zones for later case. Same thing will happen for other zone-pairs like 2, 5 and 3, 4. That's why; the ΔT^{ref} and its responses are shown only for first three. In the plot's legend; reference temperature profile, inner wall and outer wall temperature profiles has been indicated by 'UT','InT' and 'OutT' respectively. L11 indicate the response function or outer wall temperature profile at outer layer-1 due to ΔT^{ref} on inner surface -1, thus the response at outer surface-2 due to ΔT^{ref} in the inner surface-1 is denoted by L12 and so on. The response profile is similar to error function which is frequeantly appeared in heat conduction problems.

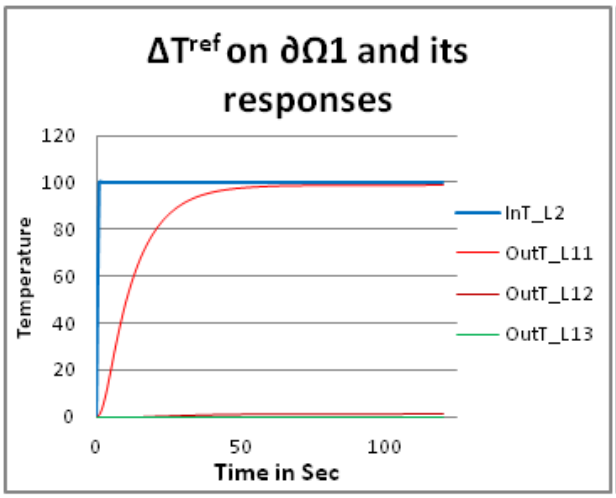


Figure 5.5: ΔT^{ref} at inner surface-2 and its responses to outer sufaces

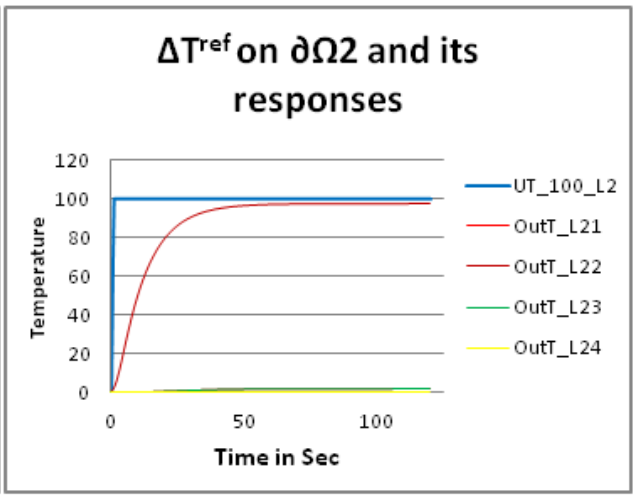


Figure 5.6: ΔT^{ref} at inner surface-2 and its responses to outer sufaces

Finally, **Mixing zone:** To know the thermal behaviour accurately it is necessary to know the temperature mixing zone. When we apply some temperature at inner surface it mainly affects its nearest neighboring zones. To separate different temeprature at different zones it is necessary to know the affected zone of mixed temperatures, influenced by two or more zones input condition. A picture from the visual representation of the first simulation is shwon in the figure 5.8, which shows that the most singnificant mixing zone $[\Delta T^{ref}$ *on* $\partial\Omega_1]$ exists in an angular distance of 18-20 degree. In fact the influenced zone distance is about 30^o in each side which we have seen from the graphs 5.5 to 5.7, in which $18\text{-}20^o$ area from both neighboring zones is highly affected.

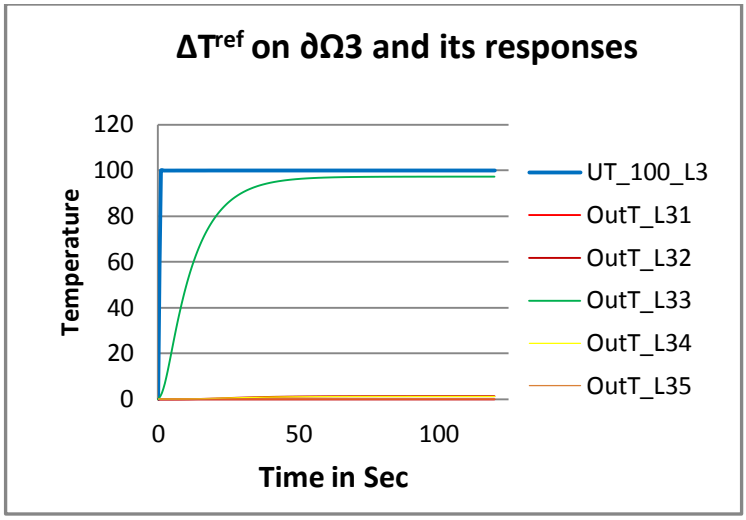


Figure 5.7: ΔT^{ref} at inner surface-3 and its responses to outer sufaces

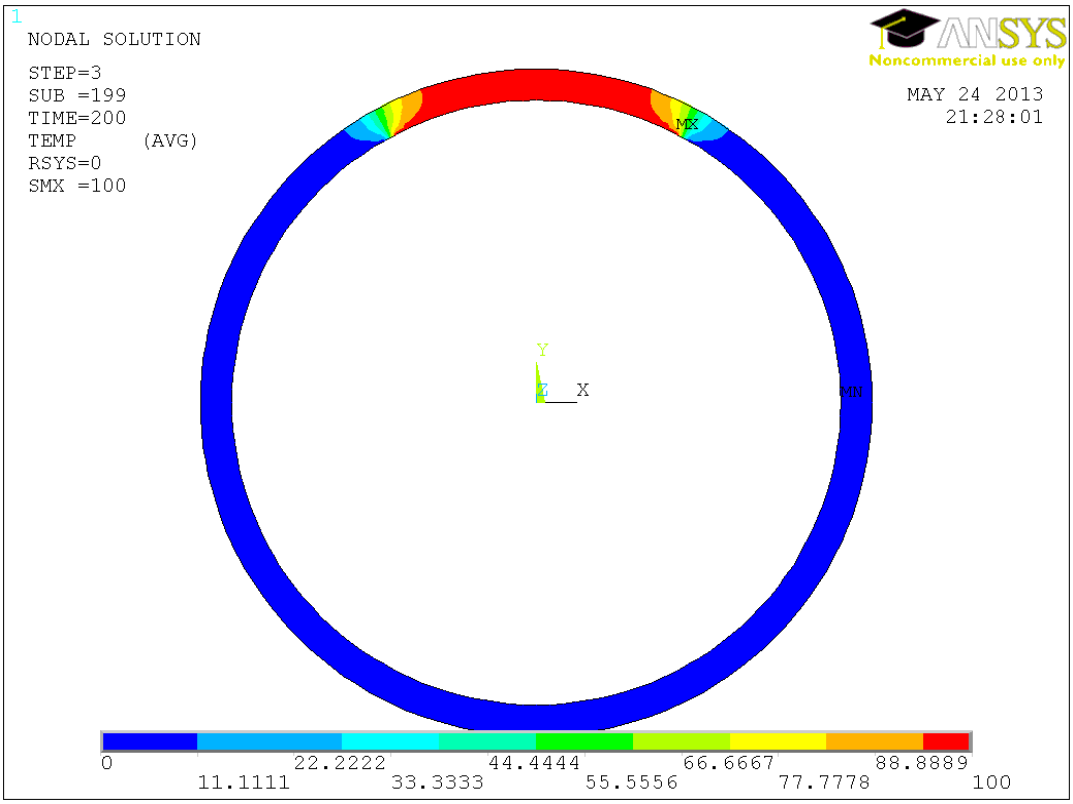


Figure 5.8: ΔT^{ref} on $\partial\Omega_1$, Temperature profile at time t=196

5.1.2 Simulation with six zones, D_{in} =406.4 mm, S=19mm thickness

This simulation is to know what happens with higher thickness and diameter of a pipe from a practical case. We have seen that there is a highly concentrated mixing zone in between two layers where strong intermixing of temperatrue occurs ; in this regard, length of this strong mixing zone varies depending on the thickness and inner radius. If the thickness below 20mm

then the mixing zone occurs in the range of 18 to 22 degree angular ditance. Thus if a subdomains angular distance more than 22 degree then we will miss the most part of the mixed zone and so, we will not get the accurate picture of the thermal stratification. For the given thickness and diameter, though diameter is bigger than last one, we are getting 98.7^o and only 0.6^o at the mid point of the nearest zone as shown in the figure 5.8. Moreover, for $\Delta \boldsymbol{T}^{ref}$ applied on $\partial\boldsymbol{\Omega_1}$ corresponding temperatures are 99.3 and 0.6^o respectively [shown in the fig-5.9]. Following two figures indicated that we are missing the most significant part of mixing zone. If we use these data into the inverse algorithm, it is expected to be nearly similar to one dimensional case with very small differences.

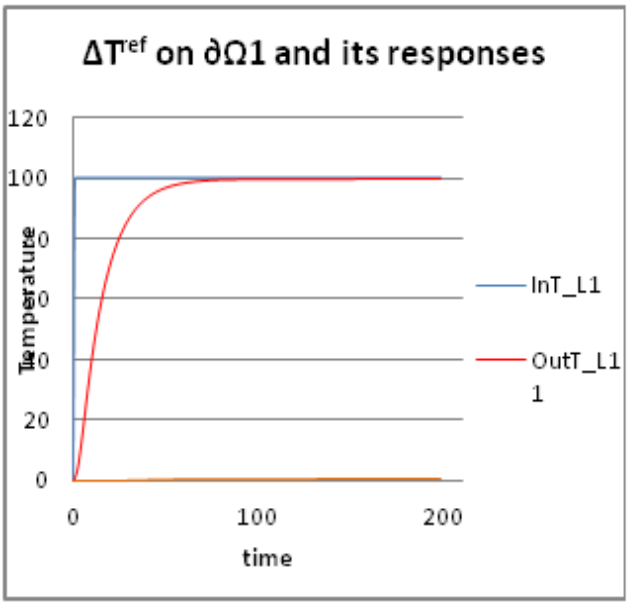


Figure 5.9: ΔT^{ref} at inner surface-2 and its responses

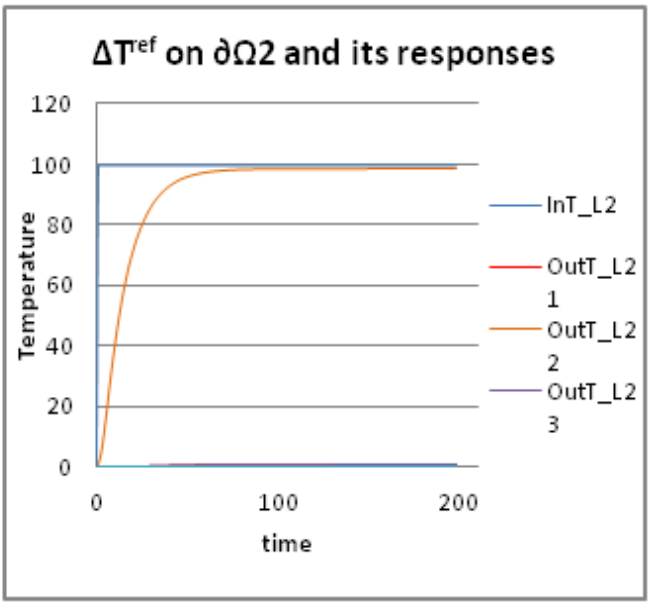


Figure 5.10: ΔT^{ref} at inner surface-1 and its responses

5.1.3 Simulation with 18 sub-domains and 10mm thickness

Now we want to see the responses when we consider smaller subdomains, for this reason several simulations has been done with 18 zones keeping the typical thickness of 10mm. Thermophysical properties and element type are kept same like the first one. Applying $\Delta \boldsymbol{T}^{ref}$ on the inner surface number-1 results temperature distribution of 81.88^o on its outer part and 16.7^o and 1.3^o to the neighboring outer surfaces. Simulation number-2 or $\Delta \boldsymbol{T}^{ref}$ on the second zone shows the distribution of 66.4^o on its own outer surface, 16.3 ^{o} on both of its neighboring outer surfaces and 1.2 ^{o} on next surfaces. While third simulation shows the distribution as 66.3^o, 15.5^o and 1.3^o. Remaining simulations follows the same pattern of simulation-3 and so those are not explained in detail. Corresponding figures of first three simulations are given in the figure 5.11, 5.12 and 5.13.

Outer surface temperature profile has been denoted by 'RT' with the same meaning of previously used 'OutT'.

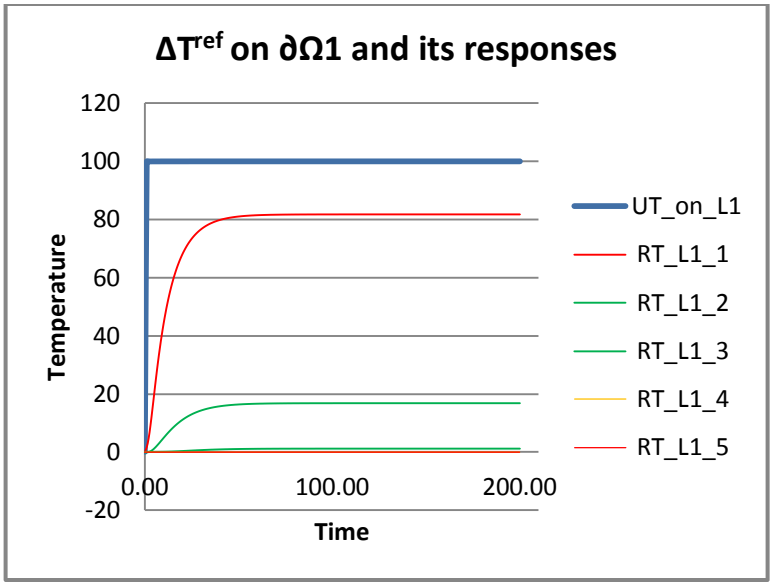


Figure 5.11: ΔT^{ref} at inner surface-1 and its responses

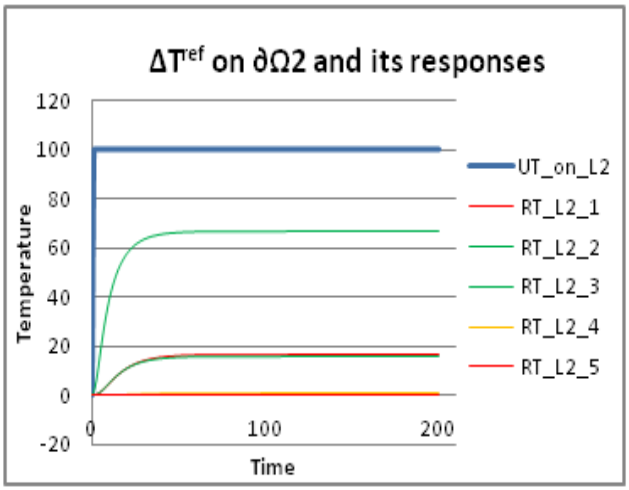


Figure 5.12: ΔT^{ref} at inner surface-2 and its responses

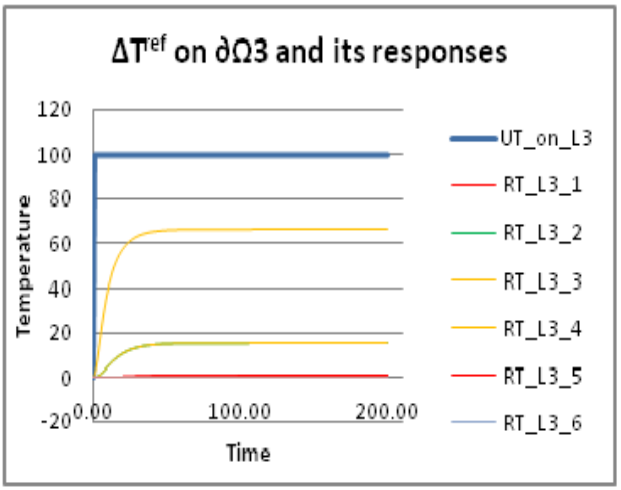


Figure 5.13: ΔT^{ref} at inner surface-3 and its responses

5.2 Simulation with 30mm thickness, D_{in}=160, 12 zones

As part of testing with bigger thickness, a simulation with 30mm thickness has also been attempted with 12 subdomains. Only one simulation applying reference temperature on 1st zone has been done to get the temperature distribution on different zone.

Discretized domain or ANSYS model and steady state condition have been presented by the two following figures 5.14 and 5.15. It shows at steady state the temperature on its 1st zone's outer surface is 56.18^o whereas 25.28, 7.4, 2.03, 0.56, 0.15 degree on subsequent zones. Complete response profiles and steady state response are shown in the figure 5.16 and 5.17.

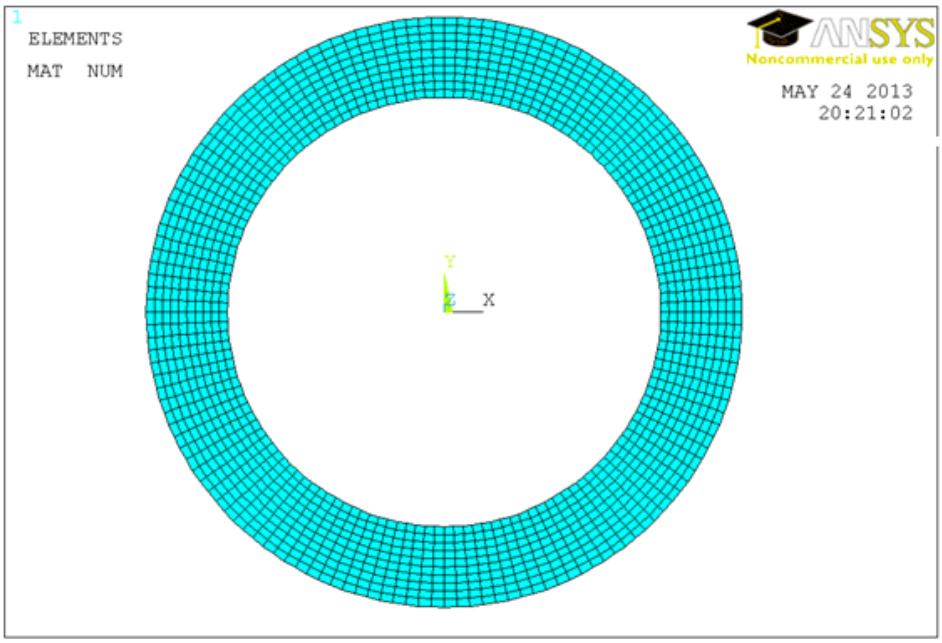


Figure 3.14: Ansys model of 30mm thick pipe

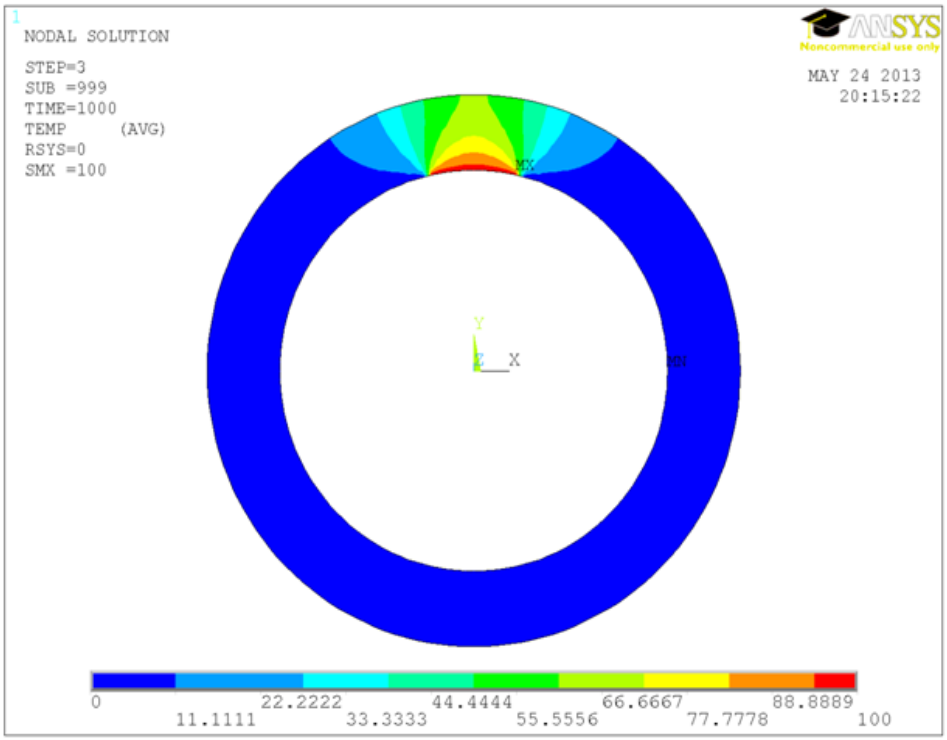


Figure 5.15: ΔT^{ref} Responses at steady state for 30mm thick pipe

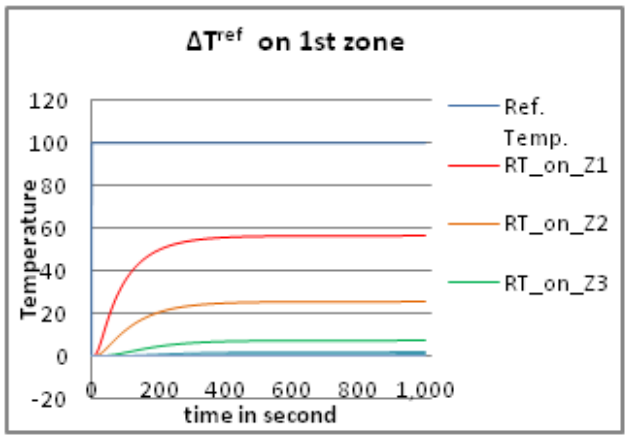


Figure 5.16: ΔT^{ref} on the 1st zone and its responses

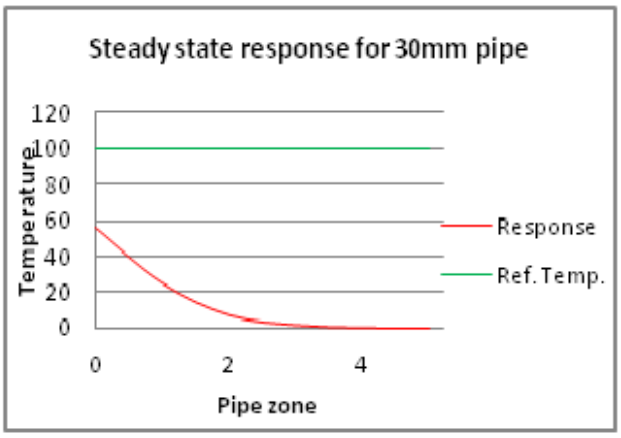


Figure 5.17: Steady state response of ΔT^{ref} applied on the 1st zone and its responses to other zoens

5.3 Simulation with 40mm thickness, D_{in}=160, 12 zones

One simulation for 40mm thick pipe has also been performed which shows that even after 1000 second the reponses of 100 degree referecnce temperature do not get steady. This unsteady behaviour is shown in the figure 5.18.

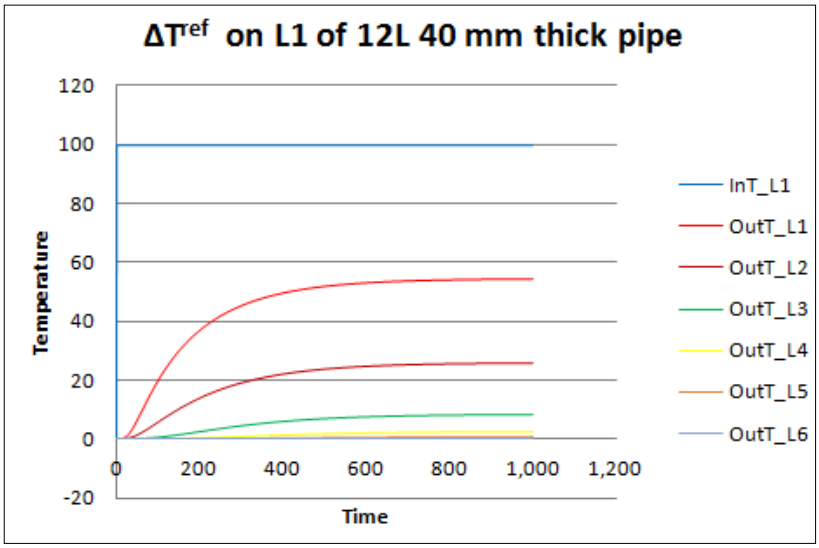


Figure 5.18: ΔT^{ref} on the 1st zone and its responses for 40 mm thick pipe

5.4 Simulation Summary

The first and foremost point of the various simualtions is the length of mixing zone. We have seen in the picture taken from postprocessed visual information that there is a fixed lenth of mixing zone depending on the thicknesss of the pipe, in whcih we can see that in some area in the vicinity of two layers boundaries the mixing is strong and dominating. To get reasonably accurate stratification behaviour, we need to concentrate on this mixing zone. If the arc length of a layer is bigger than the length of strong mixing zone we will miss the the actual stratification pattern. So, it is necessary to fix a layer‘s length less than that of mixing zone. For 90 mm inner radius and 10mm thick pipe this length is in between 18^o to 22^o which means we need at least 10 zones or layers to get reasonable accuracy. In the same way we can conclude that for 30 mm at least 12; and for 40mm at least 15 zones are necessary.

The second important thing is the length of steady state time which is important for controlling total computations in the algorithm. The selection of this duration is user choice, say for example, in the first simulation a user can take 50 to 90 seconds. Of course, longer steady state time ensures better accuracy. For 19mm thickness, the lower bound of steady state time is around 90 sec whereas for 30mm it is near 600 s. However, 40 mm thick pipe showed that even in 1000 s it does not get steady. These steady state time are not supposed to be strictly simialar for same thickness, but it depends on other thermal properties as well.

6. Implementation, Test and Results

This chapter will discuss the algorithm implementation related issues and test results of forward and inverse heat conduction algorithms for different diameter and thickness ranging form small number of layers or zones to comparatively higher number of zones. All tests are based on Scilab and MATLAB* program snippets attached in the appendix.

6.1 Implemention in SciLAB and MATLAB

Firstly, **About SciLAB**: Scilab is a freely distributed open source scientific software package for numerical mathematics and Scientific visualization mainly used in system control and signal processing applications developed by French based research institutes INRIA and ENPC, and now it is owned by the Scilab Consortium. This high-level, numerically oriented programming language provides an interpreted programming environment with syntax and functions which have high similarity with MATLAB; it is designed to work mainly with matrices and vectors as data type. It is capable of interactive calculations as well as automation of computations through programming. By using matrix-based computation, dynamic typing, and automatic memory management, many numerical problems can be expressed in a reduced number of code lines, as compared to similar solutions using traditional languages, such as Fortran, C, or C++. This allows users to rapidly construct models and to perform tests for a range of mathematical problems. Updated version of this software can be used for signal processing, statistical analysis, image enhancement, fluid dynamics simulations, and numerical optimization of explicit and implicit dynamics as well.[41]

Secondly, **Implementation in Scilab** : Since previous works at AREVA were mainly based on Scilab environment, that's why all preliminary programming and algorithm testing tasks of this thesis were also based on it. For up to 6 zones of the domain with 500 second simulation time it worked fine in Scilab. However, the program crashed for implementing the same size inverse problem causing 'stack-size overflow' indicating the environment is not greatly optimized in memory handling. It should be noted that for both forward and inverse algorithm, we need to store responses at each global time step into a matrix. For the complete problem this intermediate response matrix is a 3D matrix, which requires substantial memory spaces. For 6 zones, 500 second problem, both for forward and inverse cases, the size of that matrix is 6^2*500^2 *8 ≈ 72MB, which exceeds default stack size 76 Mb. To accommodate with larger data size, an additional function is necessary in order to set the maximum possible stack size [stacksize('max')]. In forward case, we can effectively utilize memory size by a little code optimization techniques e.g. by copying the data at each step into resultant temperature history matrix and reusing the space of that variable; but in inverse case it is possible with some complex code optimization techniques.

*Tested with academic version of the University of Erlangen-Nuremberg

In our implementation, no optimization techniques have been attempted. On the other hand, maximum possible stack size is also limited to some hundred Mb in Scilab, which eventually limits the problem size. Besides, the speed in Scilab is almost one-third as MATLAB. Considering all of these difficulties, we can say that we need another efficient and robust environment without Scilab, and hence we had to shift to MATLAB for comparatively larger problem size.

Thirdly, **MATLAB Implementation**: The main reason for re-implementation forward and inverse problem in MATLAB is mainly to overcome memory storage issue for a single variable. It is already mentioned that the intermediate variable is a 3-D matrix which requires huge data depending on the problem size. MATLAB can handle a variable with maximum size of 1 GB. With this functionality in MATLAB, we can simulate a 6 zones problem for 30 minutes.

Forthly, **Computational time**: Problems size with 18 zones and 500 second for forward case took around 15 minutes in Scilab, while Matlab computes the same problem by less than 6 minutes.

Fiftly, **Main block of the Implementation:** The challenging part is forward algorithm is to implement with a single block which needs four nested looping with an if-else control structure. In inverse case, the main block contains two sub blocks, one for calculating right side b vector and other one is for implementing forward response part which results a combined complex looping and control structure. Source condes both for Matlab and Scilab are included in the appendix.

Finally, **Flow chart**: The way the inverse algorithm is implemented is shown in a complete flow chart as given in the figure 6.1.

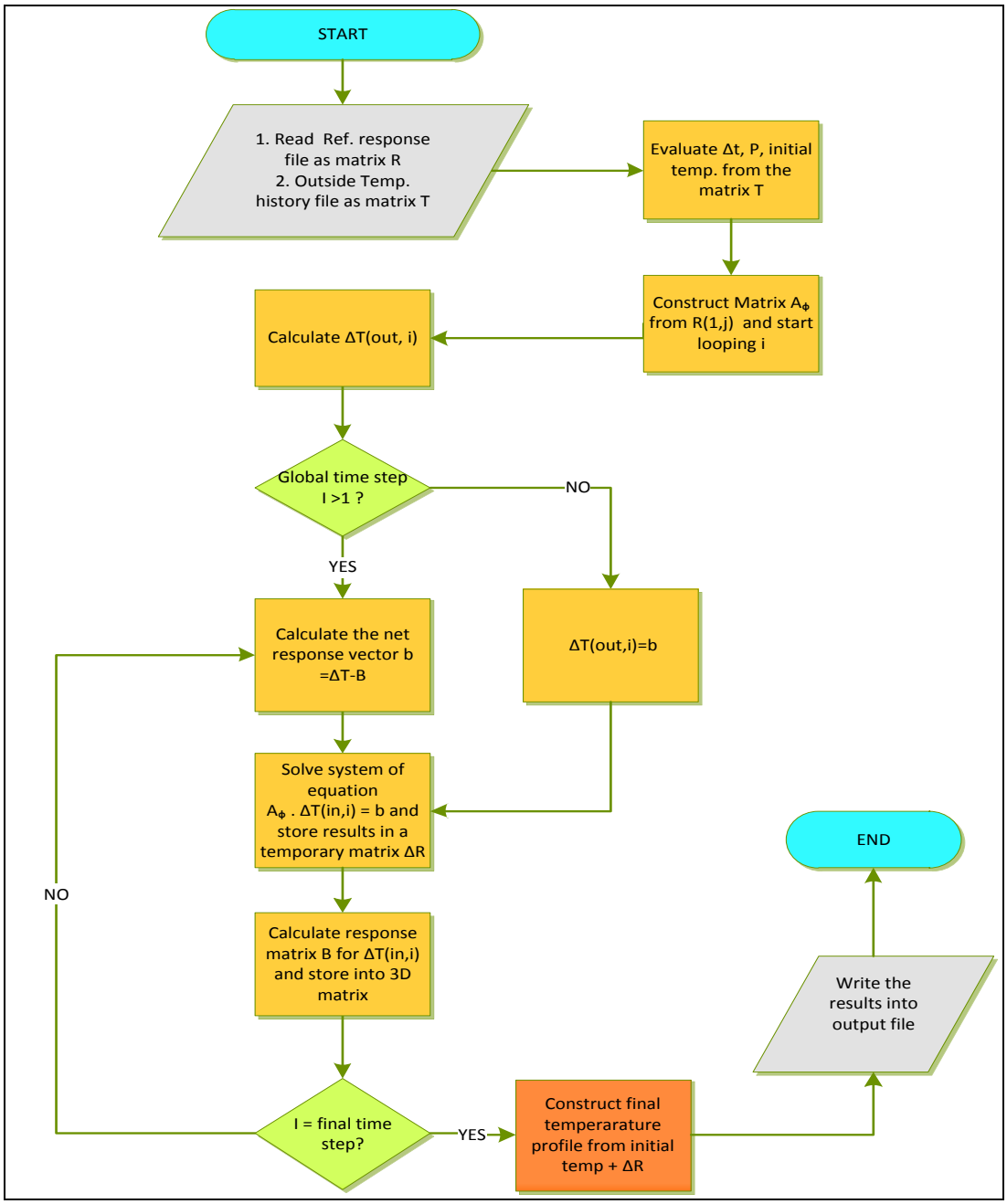


Figure 6.1: Flow chart of the inverse algorithm

6.2 Results of 2L Problem

At first, the testing was based on a very simple and typical case with 2 zones, 20 time steps with a typical pipe thickness and diameter. Inner and outer wall temperature history and reference temperature responses are known. Forward algorithm reads the reference temperature matrix R^{21*5} and input temperature history matrix T_{in}^{21*2} from two input files and calculates the output temperatures history for corresponding two zones and then writes the results into a output file.

On the other hand, Inverse algorithm reads R^{21*5} and outer wall temperature history T_{out}^{21*3} , then calculates the results and store into a file. One extra column has been used here in T_{out}^{21*3} for counting time.

Since the problem is so simple and typical, both forward and inverse algorithms computes almost exact results in both cases as shown in the above figure, showing known results and algorithm generated results in a single graph. In the legend, inner temperature history and outer temperature are denoted by 'InTemp' and 'OutT' while the word 'Scilab' is added to indicate that the result are from algorithms computed by Scilab environment.

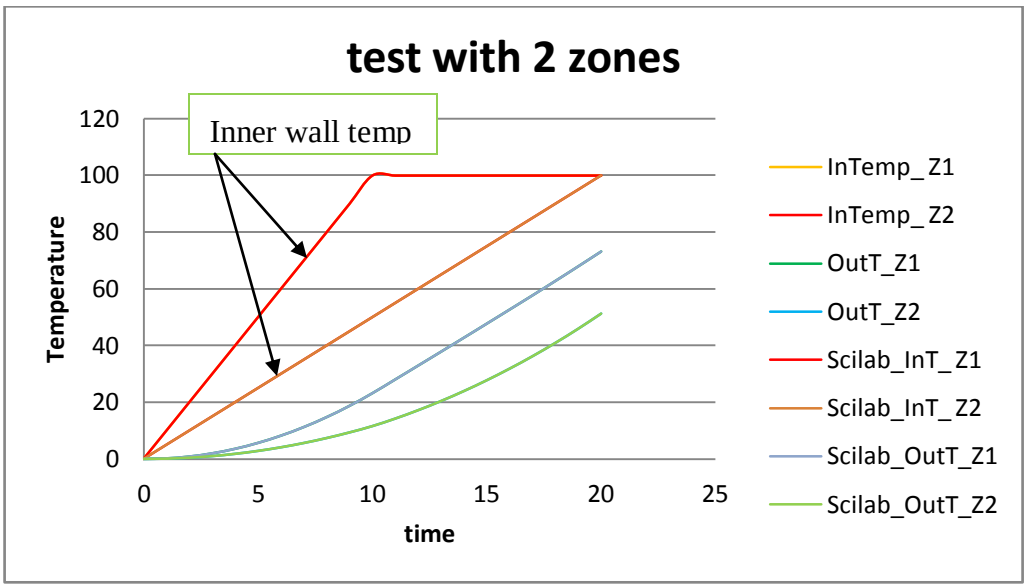


Figure 6.2: showing test results of two algorithms for 2L typical problem

6.3 Test and Results with 6L case

In this part we will check the algorithms for comparatively higher number of zones (six) than last one; other parameters are 10 mm thick, 90 mm inner radius and time domain is 200 second. Specifically we will check whether the algorithms are capable to return original input temperature history, e.g. whether inverse algorithm can give the correct profile using the data computed by the forward algorithm. Test results show that original temperature history is almost same as computed history. In short, forward algorithm generates outer wall temperature profile, while inverse algorithm uses that computed profile and give the inside history. If both original and computed profiles are same then it can conclude that both algorithms are working fine. Two different plots have been presented in this regard. Figure 6.3 shows the computed outer temperature profiles; while 6.4 compares original known inside profiles and inverse algorithm generated inside history. Later one is evident that two sets of inside history data are almost the same.

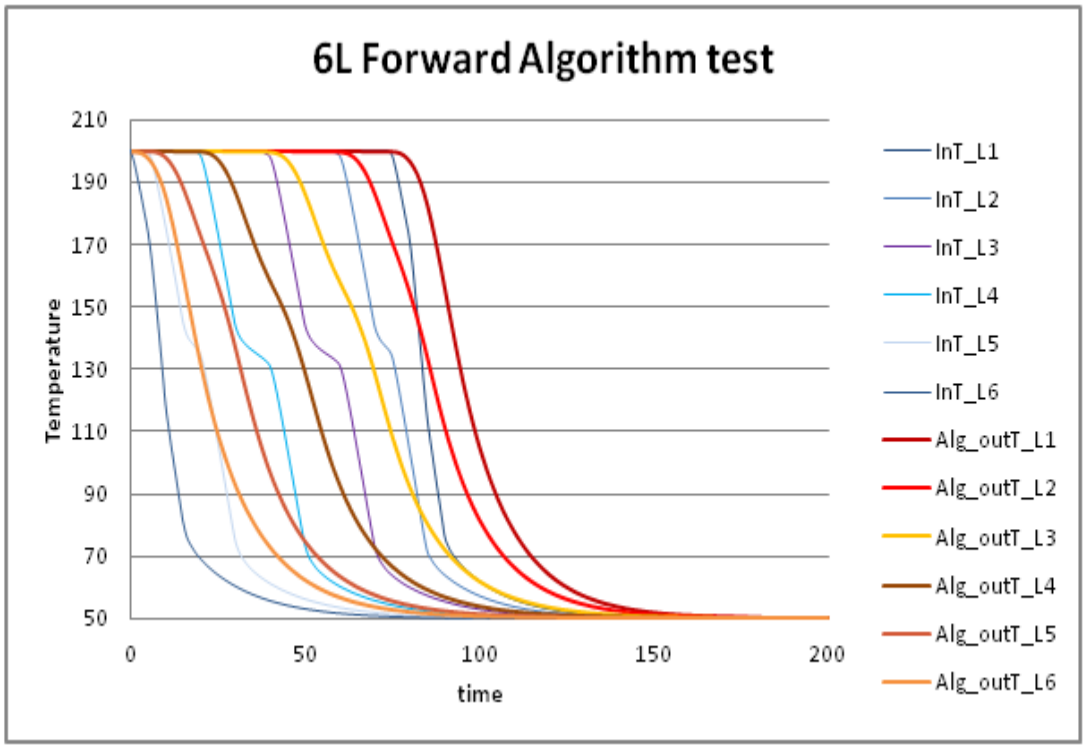


Figure 6.3: showing Inner and algorithm computed outer wall temperature history

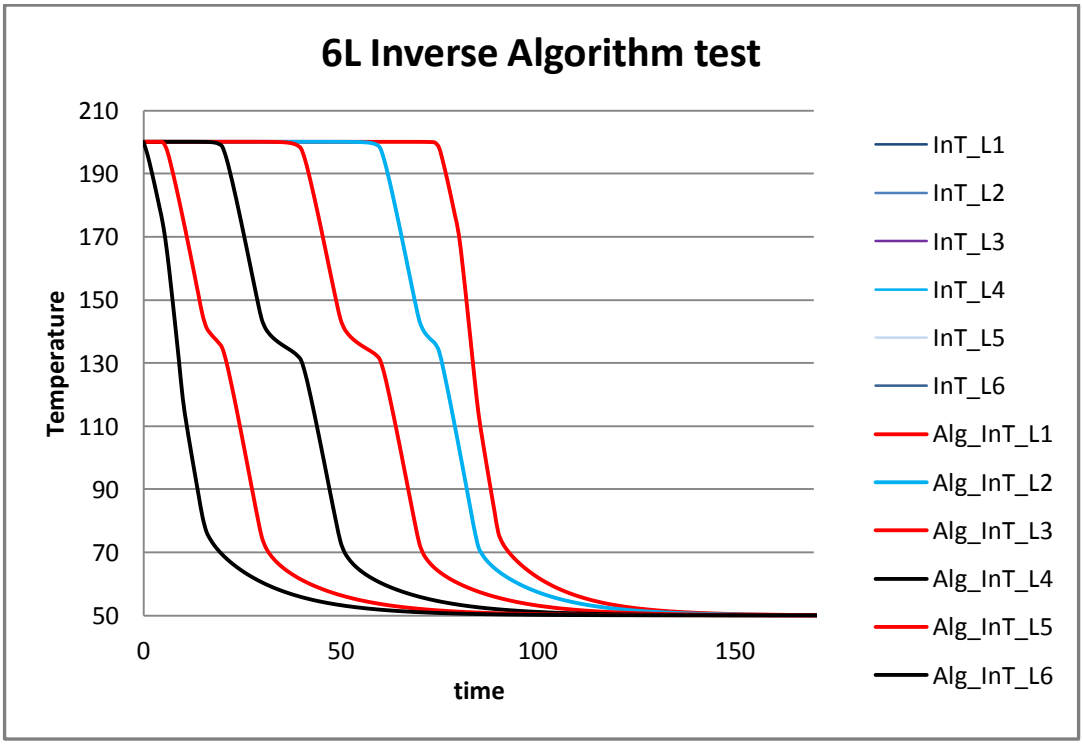


Figure 6.4: Comparing original and computed inner wall temperature history

6.4 Results of 18L problem with 10mm thickness

Now we will test with a higher number of zones e.g. with 18 zones, whether the algorithms are able to compute correctly. First we will check with two different temperature profiles and then will also check with a variety of temperature in different inner zones. For 2 different temperatures only, one temperature history is applied at the upper half of the 2D pipe and another one at the lower half inner surfaces. To verify algorithm computed results, an ANSYS simulation has also been performed. Related parameters are the same as mentioned in the previous part 6.3. ANSYS simulated forward problem can be represented by the following plot 6.5. Two sharp edged blue like straight line profiles are user chosen input history and other curved profiles are ANSYS simulated outer wall temperature profile as written in the legend.

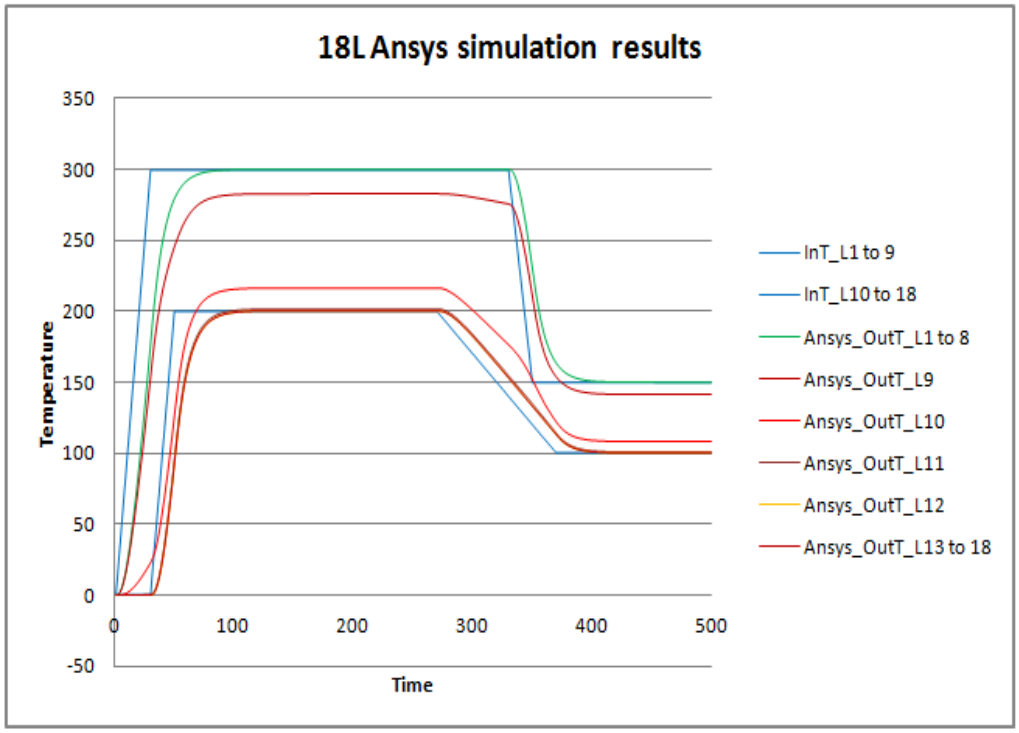


Figure-6.5: Inside and outside temperature history, simulated in ANSYS

Testing of this simulation has been done successfully with the forward algorithm which showed that both ANSYS simulated and algorithm computed outer wall temperature profiles are almost the same. Results of both cases are presented in a single graph [fig.6.6] which shows the overlapping of similar profiles. On the other hand, computed results from forward algorithm have been inserted into the inverse algorithm to check how compatible or how accurate to return original results. Test of inverse algorithm showed also very satisfactory results what are almost exact to the user chosen input data. Inverse algorithm generated results have been compared with the original input temperature history [fig.-6.7], which is again showing the overlapping profiles indicating the computational accuracy of the forward-inverse algorithm pair.

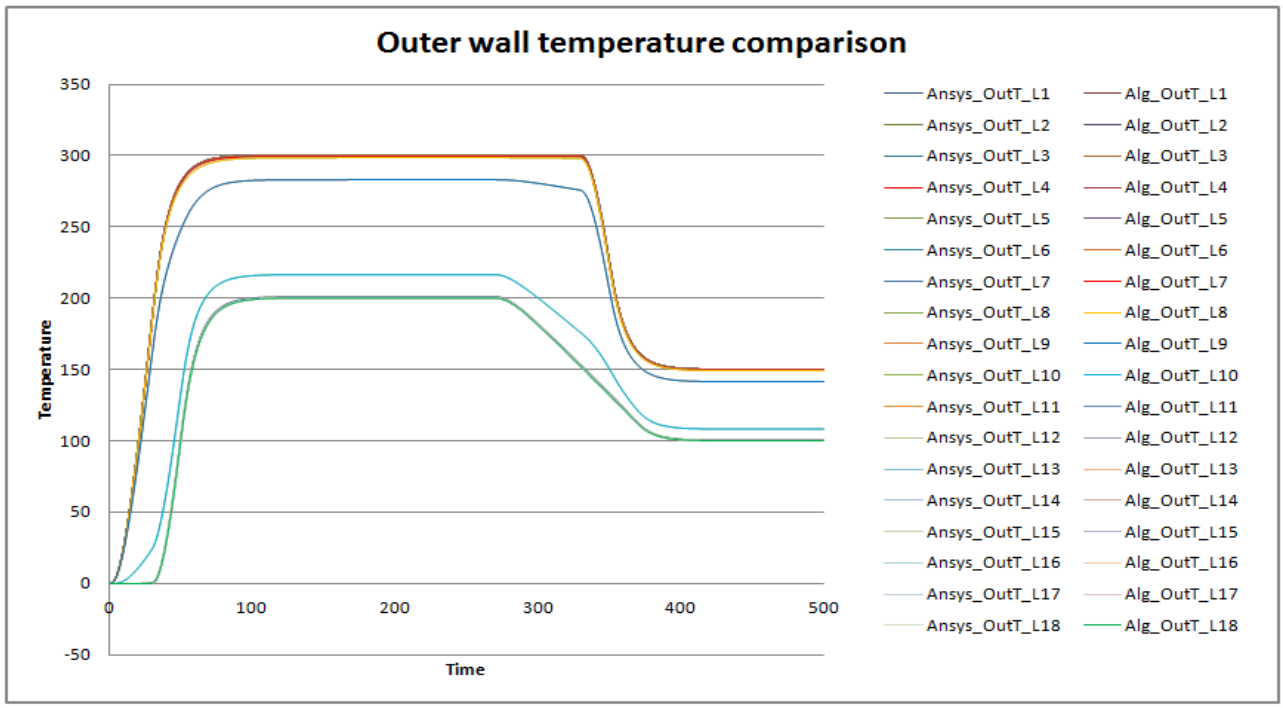


Figure-6.6: Comparison of outside temperature histories both from ANSYS and forward algorithm

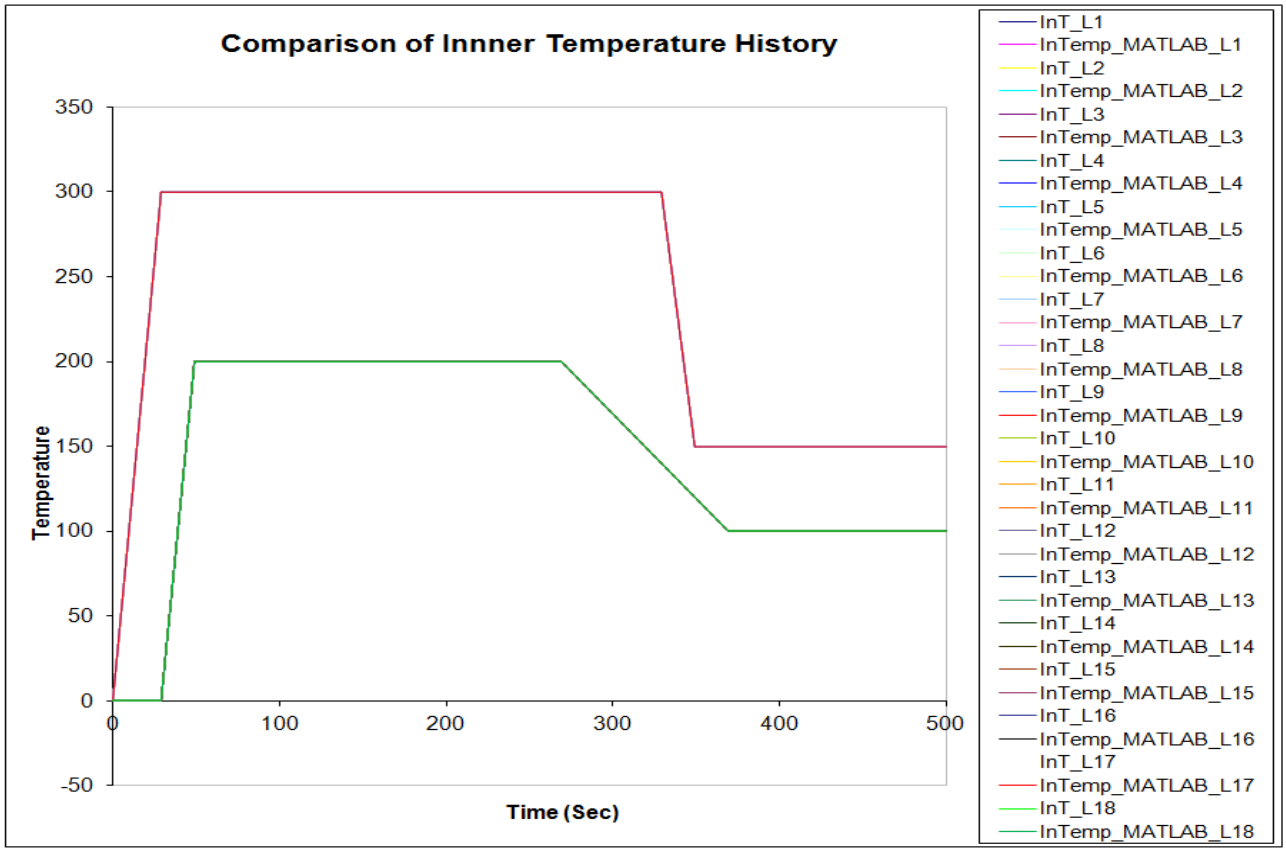


Figure-6.7: Inside temperature data both from ANSYS and inverse algorithm

Same problem has been tested with nine different temperatures in the inner side keeping same other parameters. The complete simulation has been done with ANSYS, simulation results can be summarized by the figure 6.8.

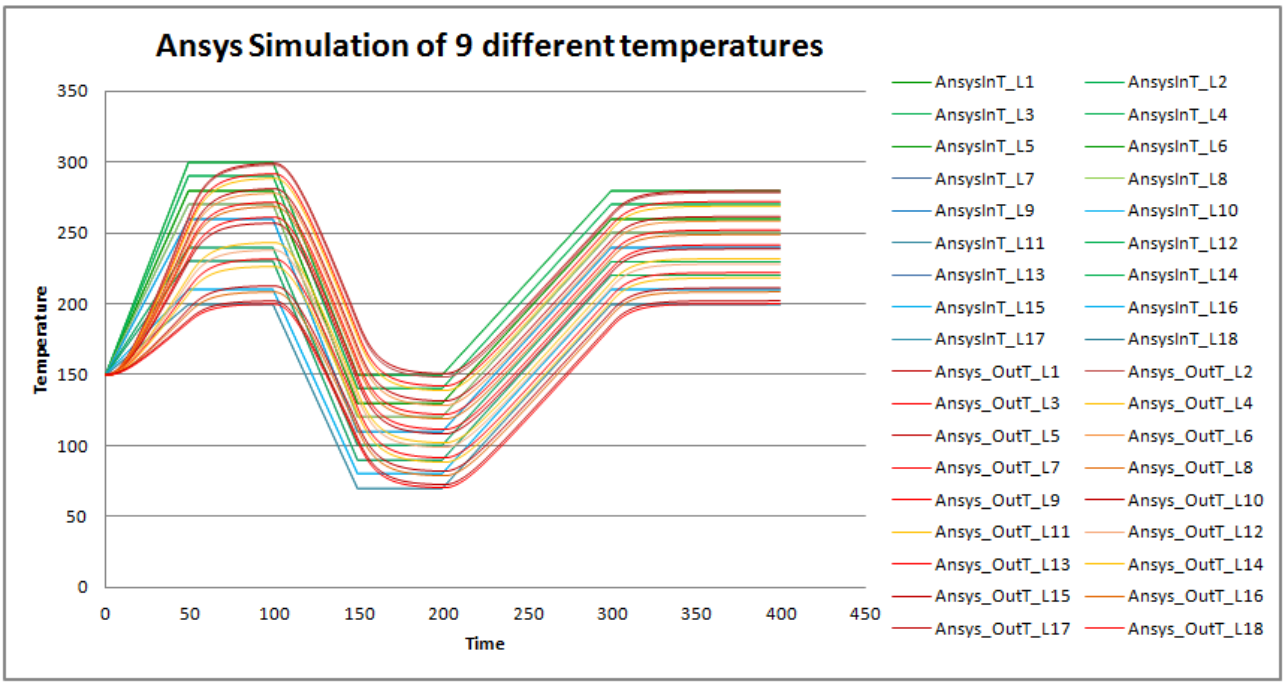


Figure 6.8: Showing inside and outside result of ANSYS simulation

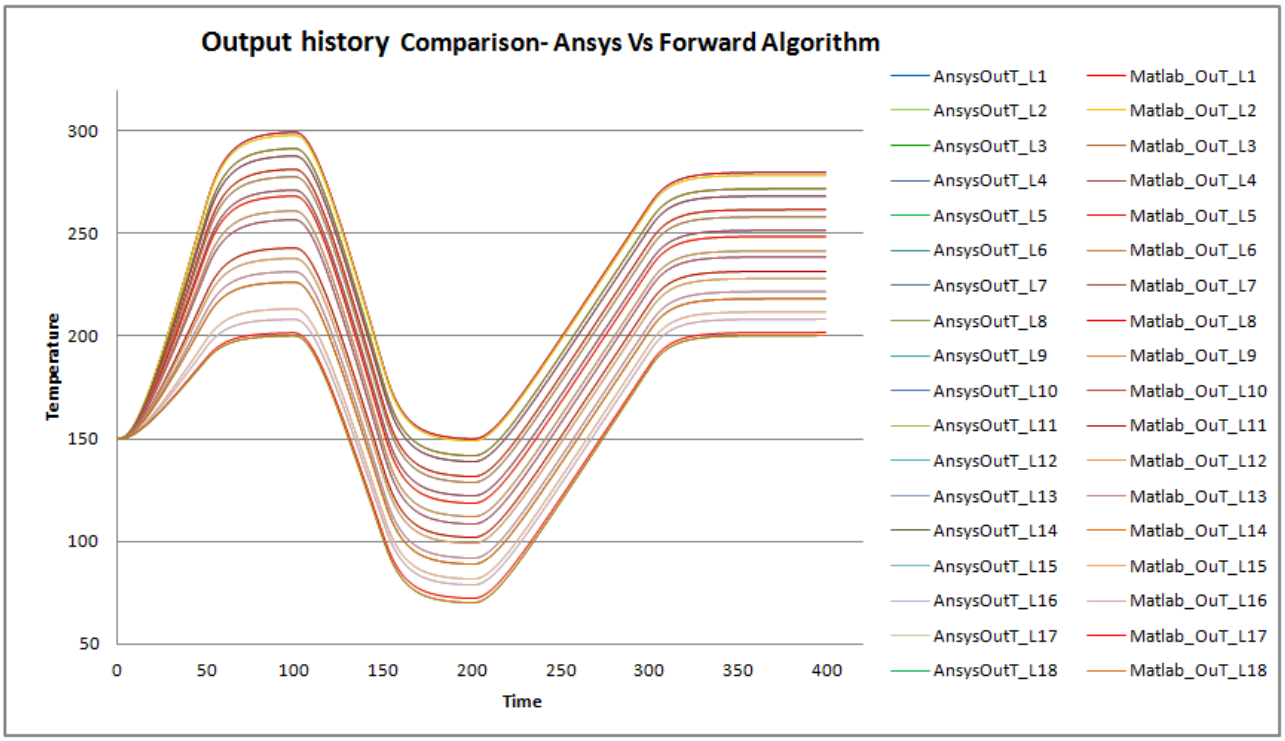


Figure 6.9: Comparison of outer side temperature profiles, ANSYS Vs algorithm

Forward algorithm takes inside temperature history as input and gives outside history, while inverse algorithm takes computed outer wall data as input and gives inner side temperature history. As per figure 6.8, blue-green like colors indicate inside data and red-orange like colors are showing outside history. For better visualization, temperatures are chosen gradually decreasing from top to bottom layers inside the pipe wall. Figure 6.9 is showing outside history comparing both ANSYS simulation and algorithm computation, while 6.10 is giving inner side temperature variations comparison of algorithm and original data. All these three graphs are showing that algorithm computed histories are almost exact to original profiles, which is evident of the fact that, if the input data, or measured data in practical cases, are correct then both algorithms are capable to give nearly exact expected profiles.

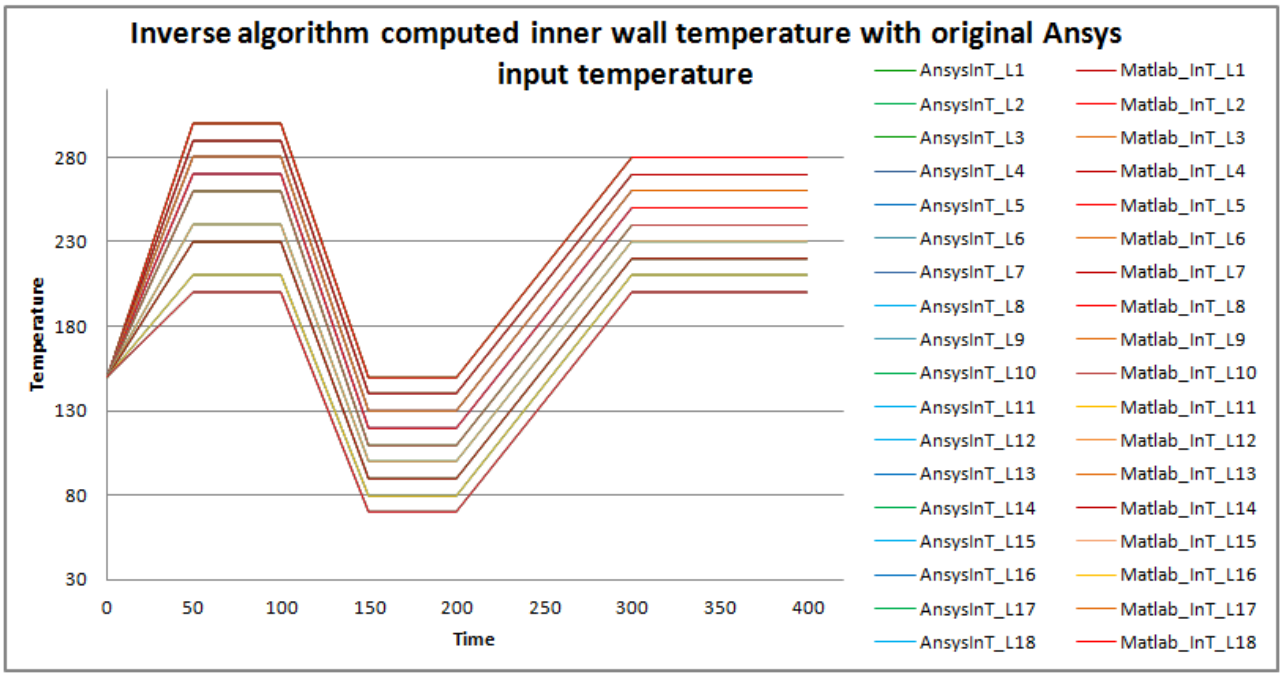


Figure 6.10: Comparison of inner side temperature profiles, known Vs inverse algorithm computed data

6.5 Real test with field data, D_{in}=406.4 mm, thickness=19 mm

In all the previous typical tests, outer wall temperature history is almost exact and hence we found that computed results are also with the highest possible accuracy. However, it is not expected that FAMOS measured data in the real case will provide with such accuracy as there will have some measurement errors and interpolation error as well. We will see what happens in an inverse problem with measurement data. Pipe parameters and thermo physical properties for this real case are listed in the following table 6.11.

Before going to apply the algorithm we will have to do some additional tasks of rearranging FAMOS data into a format that is necessary for the implementation of the algorithm. We can see that FAMOS generates outer wall history do not maintain any fixed time step; rather it stores a temperature value when it sees some difference with the previous value in order to save memory spaces. When the temperature rise or fall is very sharp, it takes at each second and stores the data. In this regard, a prototype is snapped for the thermocouple position-1 and presented here with the figure number 6.12. Seven thermocouples data were actually plotted and shown at the beginning, in the introductory chapter in figure-1.2.

Parameter	Unit	Value at room temp.
Outer Dia.	mm	406.4
Thickness	Mm	19
Elastic Modulus	MPa	212000
Specific Heat	J/g.K	0.461
Density	kg/m3	7850
Heat conductivity	W/m.K	42.5
Material Type	Ferritic Steel	

Figure 6.11: Physical and thermal properties in a table

At first we will have to interpolate the data at each second for each thermocouple since our reference temperature and reference functions are considered for one second time step size. Linear interpolation has been applied to get each second value for each thermocouple, which in turns introduces some interpolation error in the FAMOS measured data, what is very sensitive to error in ill-posed problem. Since we are considering the midpoints of zones, we will have to interpolate every second's interpolated temperature values for midpoints as well, which introduce more some error in the measured data. Next task is to select a suitable data range from the supplied two days data which can be comfortably dealt with our implementation environment. In previous testing we observed that Scilab environment cannot handle even a problem with 18 zones and 500 second due to an intermediate matrix of higher dimension and bigger size. This time we checked MATLAB environment how large data size it can handle effectively. We found that MATLAB can deal with maximum size of a variable of 1 GB, what is equivalent to 6 zones problem with 1836 second. So we need to select a range less than the time domain of 1836 second. For testing the current problem, 1800 second time domain has been chosen from the supplied and twice interpolated that supplied two days FAMOS data. Interpolated and selected history from the outer wall temperature is represented in the following graph 6.13. The selected temperature history contains increasing, decreasing even flat trends as we want to see whether the solution accuracy is rate (dT/dt) independent or not.

RL040T401.txpn - Notepad

File Edit Format View Help

```
RL040T401        FFE_strat
MFWL SG 1 NOZ MS 27     Main Feedwater Line Steam Generator 1 close to
Steam Generator Nozzle  27      12 o'clock
09.08.2011 00:11:42     150.25031363936
09.08.2011 00:15:53     150.04971146
09.08.2011 00:23:14     151.628912742795
09.08.2011 00:24:06     152.781991383197
09.08.2011 00:26:00     151.554291105408
09.08.2011 00:27:18     150.802032503828
09.08.2011 00:28:48     150.128325144592
09.08.2011 00:31:07     149.8240341812
09.08.2011 00:34:03     150.848600137999
09.08.2011 00:35:04     152.048885280123
09.08.2011 00:36:06     153.105037211235
09.08.2011 00:37:12     154.004367485182
09.08.2011 00:38:35     154.763337616079
09.08.2011 00:41:04     153.26320721601
09.08.2011 00:42:09     152.332479181055
09.08.2011 00:43:34     153.50963583973
09.08.2011 00:44:43     152.33837839853
09.08.2011 00:45:47     151.404233680592
09.08.2011 00:47:22     150.877194815652
09.08.2011 00:50:02     150.5762917772
09.08.2011 00:56:14     150.4509155112
09.08.2011 01:06:03     150.67659279
09.08.2011 01:11:38     150.3756897516
09.08.2011 01:26:15     150.676592789993
09.08.2011 01:30:19     151.220456956698
09.08.2011 01:31:06     152.306421968174
09.08.2011 01:32:26     151.529159827834
09.08.2011 01:34:03     150.977955097683
09.08.2011 01:37:00     150.601367184468
09.08.2011 01:41:49     150.200162979583
09.08.2011 01:45:26     149.8992599408
09.08.2011 01:49:38     151.048805004554
09.08.2011 01:50:29     152.230673615517
09.08.2011 01:51:33     153.205729426954
09.08.2011 01:52:40     154.06174352245
09.08.2011 01:54:00     154.813947382655
09.08.2011 01:55:29     155.465895421873
09.08.2011 01:56:56     153.914759128836
09.08.2011 01:57:45     152.71119918979
09.08.2011 01:58:49     151.758299594068
09.08.2011 02:00:19     151.027716902001
```

Figure 6.12: FAMOS generated outer wall temperature history for thermocouple position-1

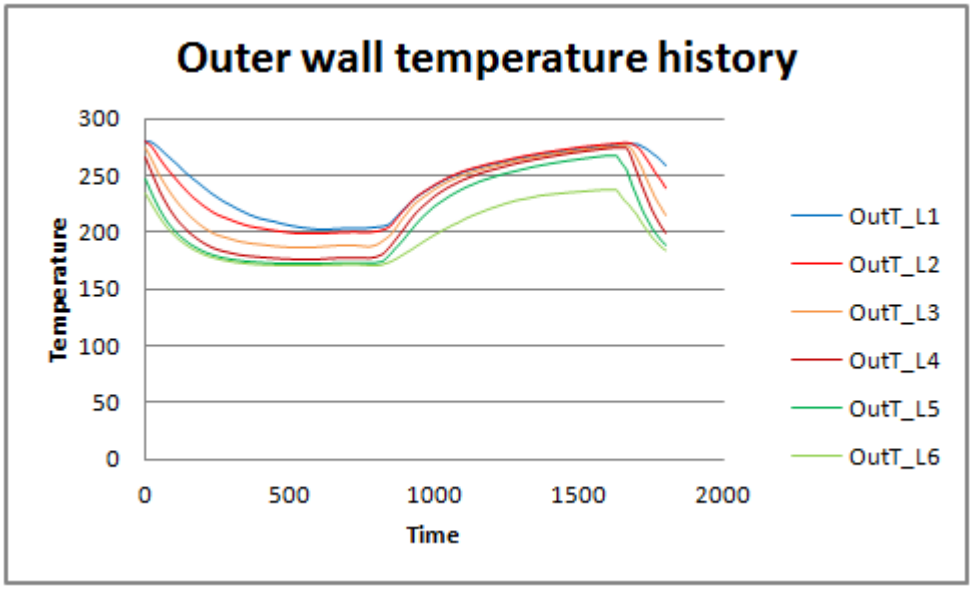


Figure 6.13: Famos generated outer wall temperature history, 1800 sec

For 1800 second, 6 zones, 19 mm thick pipe; the inverse algorithm has been applied with real field measured data which shows a reasonable trend but with huge noises and oscillations in the inside temperature history. The noise and fluctuation is not unexpected as we mentioned several times that an inverse problem is usually gives erroneous or noisy data as it is ill-conditioned or highly measurement error sensitive. To have a better visualization, the plot for outer surface and computed inner side history is divided into two parts of 900 seconds each as represented by the figure 6.14 and 6.15.

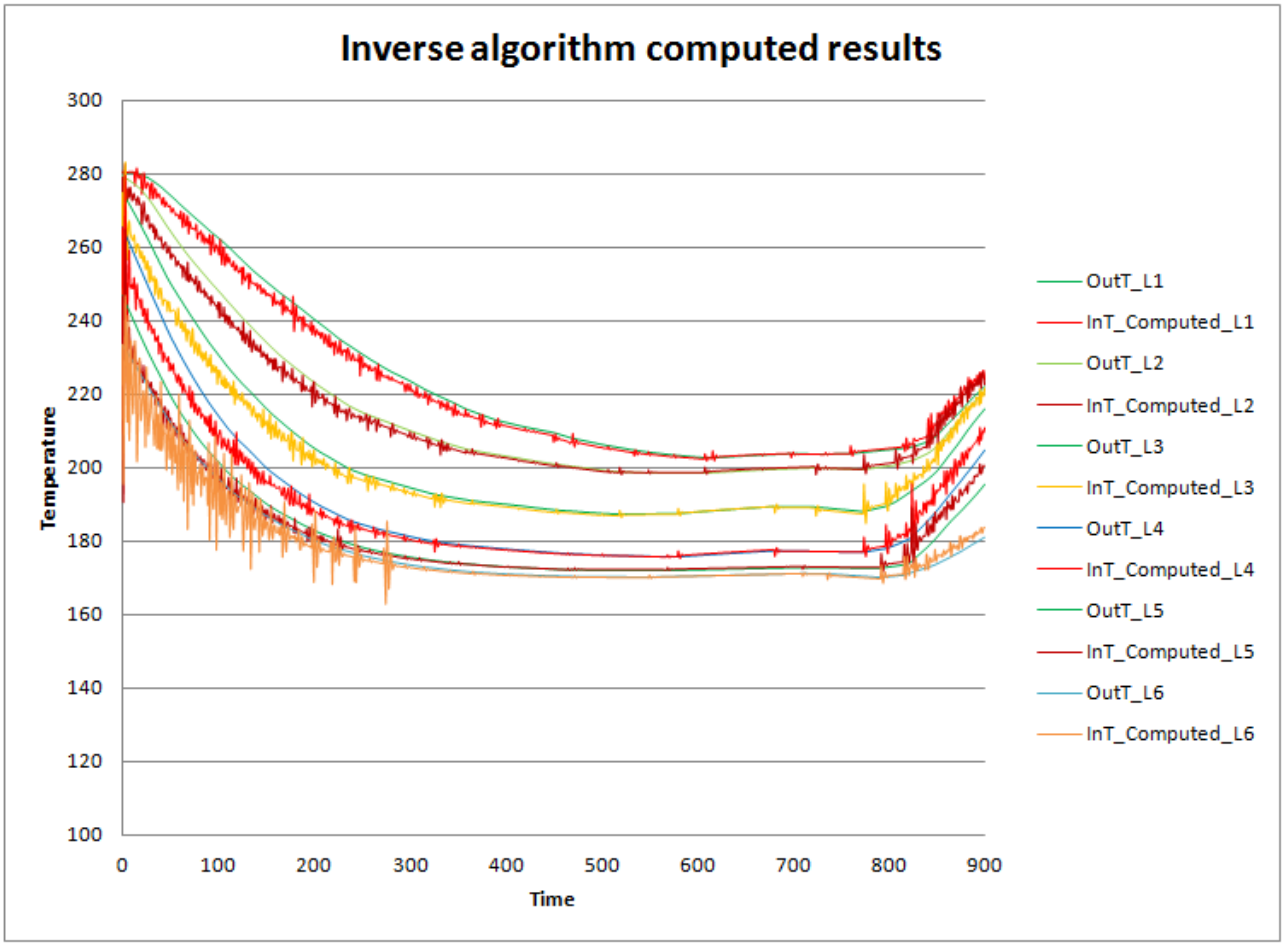


Figure 6.14: Inverse algorithm generated inner wall temperature history with corresponding outer wall history for first 900 second

With the computed noisy inner temperature profiles, we put it into forward algorithm to check the strength of the algorithm pair. Interestingly, algorithm pair generated output temperature history is the same as the measured outer wall history. To compare both data sets, figure 6.16 has been plotted and shown. In the next section we will show how to deal with the computed noisy data.

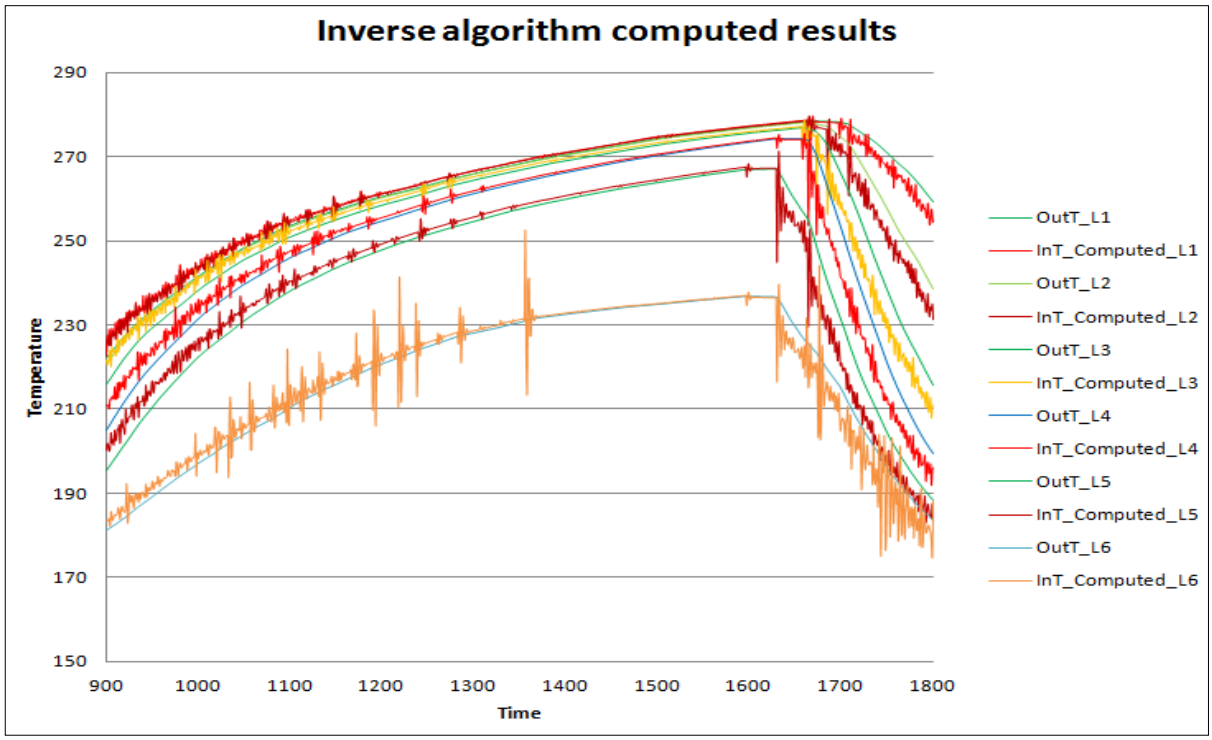


Figure 6.15 Inverse algorithm generated inner wall temperature history with corresponding outer wall history for 900 to 1800 second

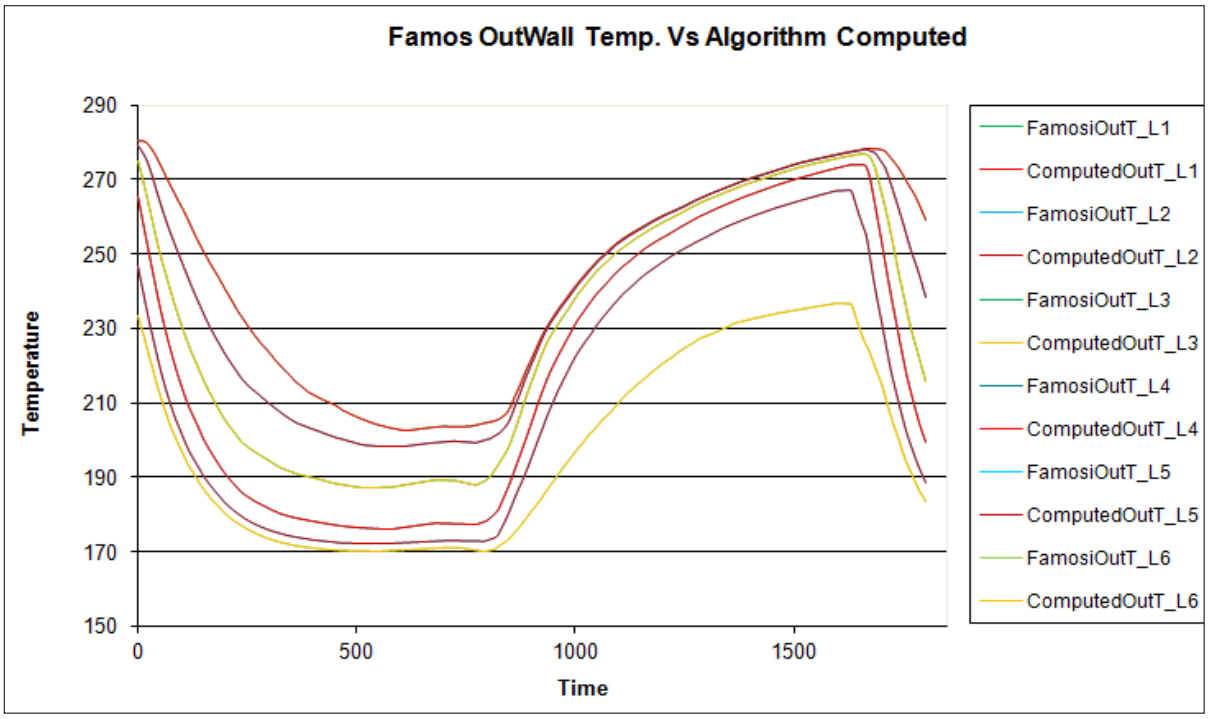


Figure 6.16: FAMOS measured and inverse-forward algorithm pair generated outer wall temperature history

6.6 Noise Filtering

In this section, first we will see how to get more acceptable trend from the inverse algorithm computed noisy results. If we consider the inside temperature as signal then it is like predictable signal as the computed profile is supposed to follow nearly similar to outside temperature profile. Since the error in the measurement is random, the noise characteristic is also random. In signal processing theory there are various types of noise filtering mechanisms depending on the types, characteristics of the signal. Moving average (MA) is the most common filter in digital signals processing (DSP), mainly because it is the easiest filter to understand and use. In spite of its simplicity, the moving average filter is *optimal* for a specific common task, reducing random noise while retaining a sharp step response. This makes it the premier filter for time domain encoded signals.[49] Another one probably could be the more effective for filtering our algorithm generated inside history, which is already discussed in the chapter-4, is the Kalman filter. Since MA is simple and common one, here we will apply MA filter to identify more acceptable pattern of the inside temperature history. As the name implies, the moving average filter operates by averaging a number of points from the input signal to produce each point in the output signal. In equation form, this is written:

$$y[i\,] = \frac{1}{M}\sum_{j=0}^{M-1} x[i+j]$$

Where y[i] is the filtered value, M is the number of points considered for averaging, x[i] is the noisy data. It is observed that higher points based MA causes shifting the average or expected profile while lower number of points cannot reduce noises significantly. Hence an iterative MA has been applied to get an optimal compromise of negligible noise with acceptable pattern. However, higher iteration also pushes the profile from the averaged one. As an attempt to optimum approach, 3 points based 4^{th} time iterated MA seems giving very smooth and reasonable profile of the expected data.

For applying MA filter, we selected some range of data and applied the filter. A profile of 300 s second is shown before filtering in the figure 6.17. Depending on the amount of noise and oscillation, MA with different points has been applied. Number of points considered here from 3 to 7 for different profiles. Profiles with noise and corresponding filtered one for 300 s have been shown in the figure 6.18. In the legend, considered numbers of points are also mentioned for respective profiles.

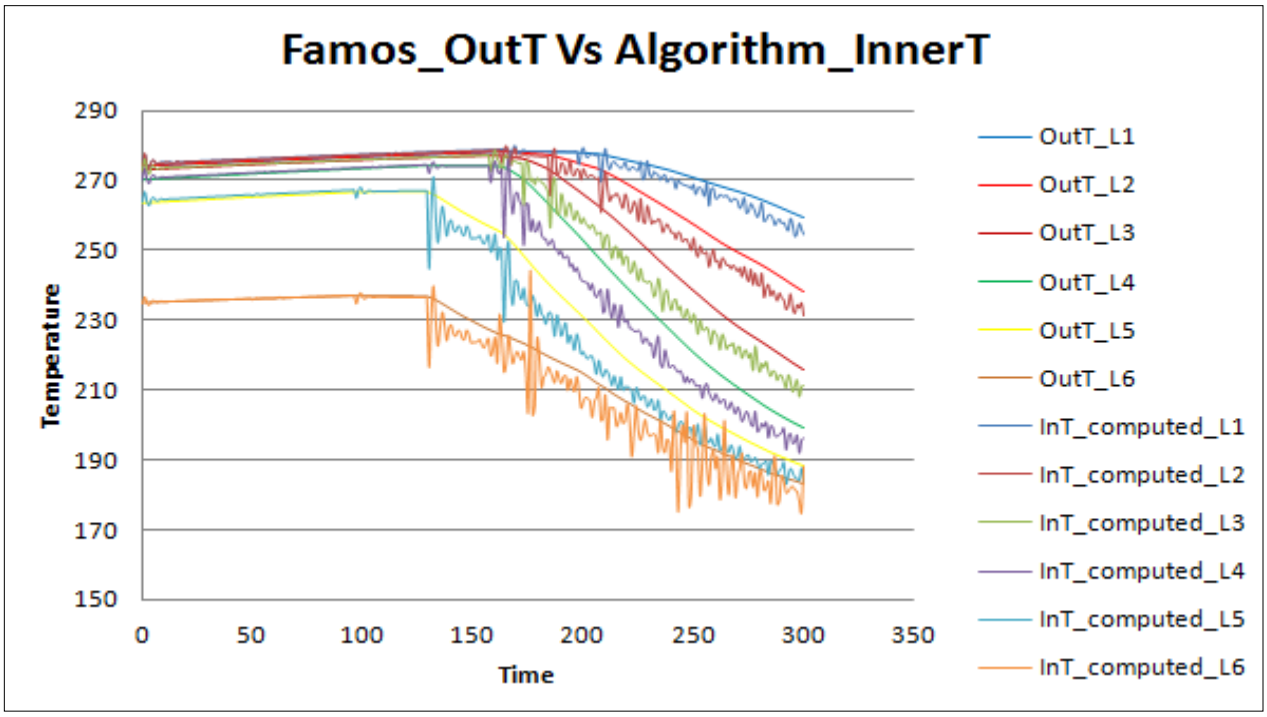


Figure 6.17: 300 sec Outer wall and noise inner wall history before applying filter

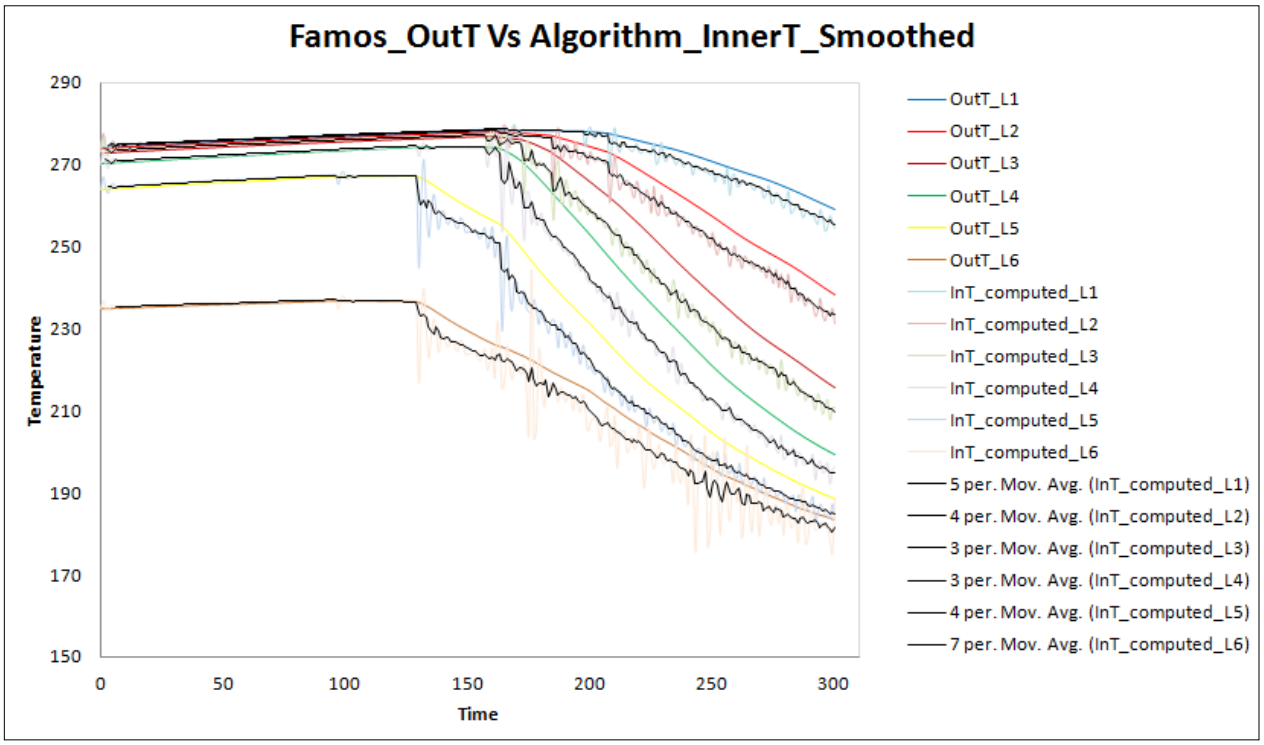


Figure 6.18: 300 sec profiles, original and filtered

Another set of profiles for 500 second have been adopted with MA filter. Figure 6.19 and 6.20 are showing the history before and after applying MA filter. The noisiest one e.g. profile number-6 took 6 point base MA filter to have a reasonable pattern. At the end number of point selection is a user choice. Observation shows that it should not be more than 6 or less than 3.

For the noisiest one 3-point based 4th iteration of MA shows a very small downward shift of the average profile as shown below figure 6.21. Interesting point is, higher points result an upward shift while iterated MA filtering shows a downward shifting trend. Hence it is observed that 3 point based 3rd iteration could be a good choice for most profiles.

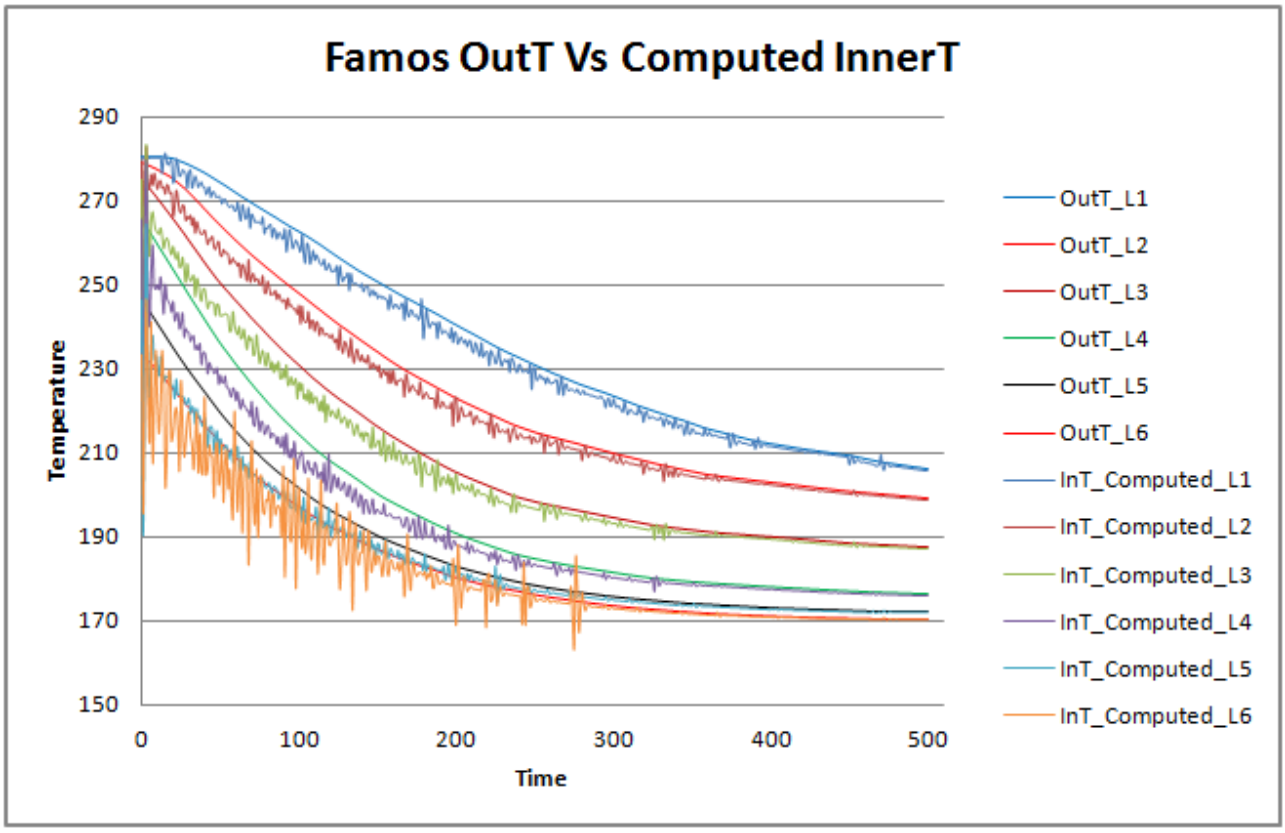


Figure 6.19: Temperature history of 500 sec before filtering

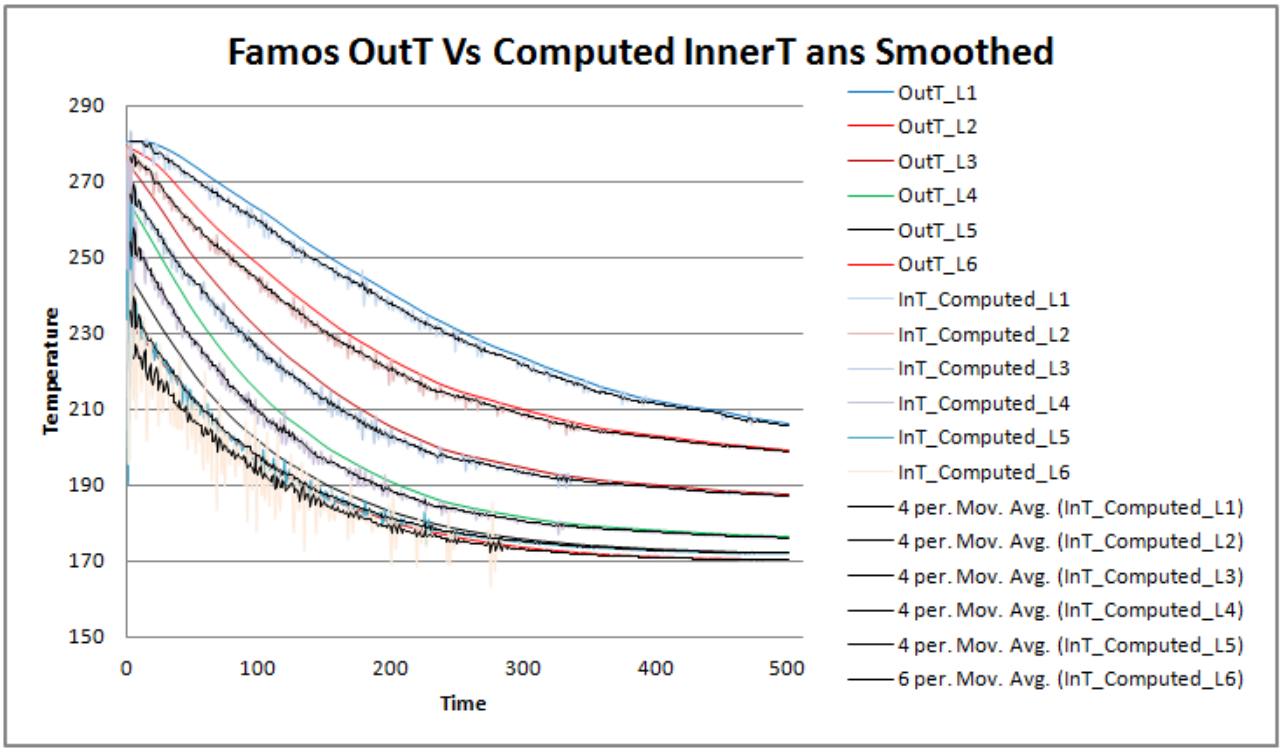


Figure 6.20: Temperature history of 500 sec after filtering

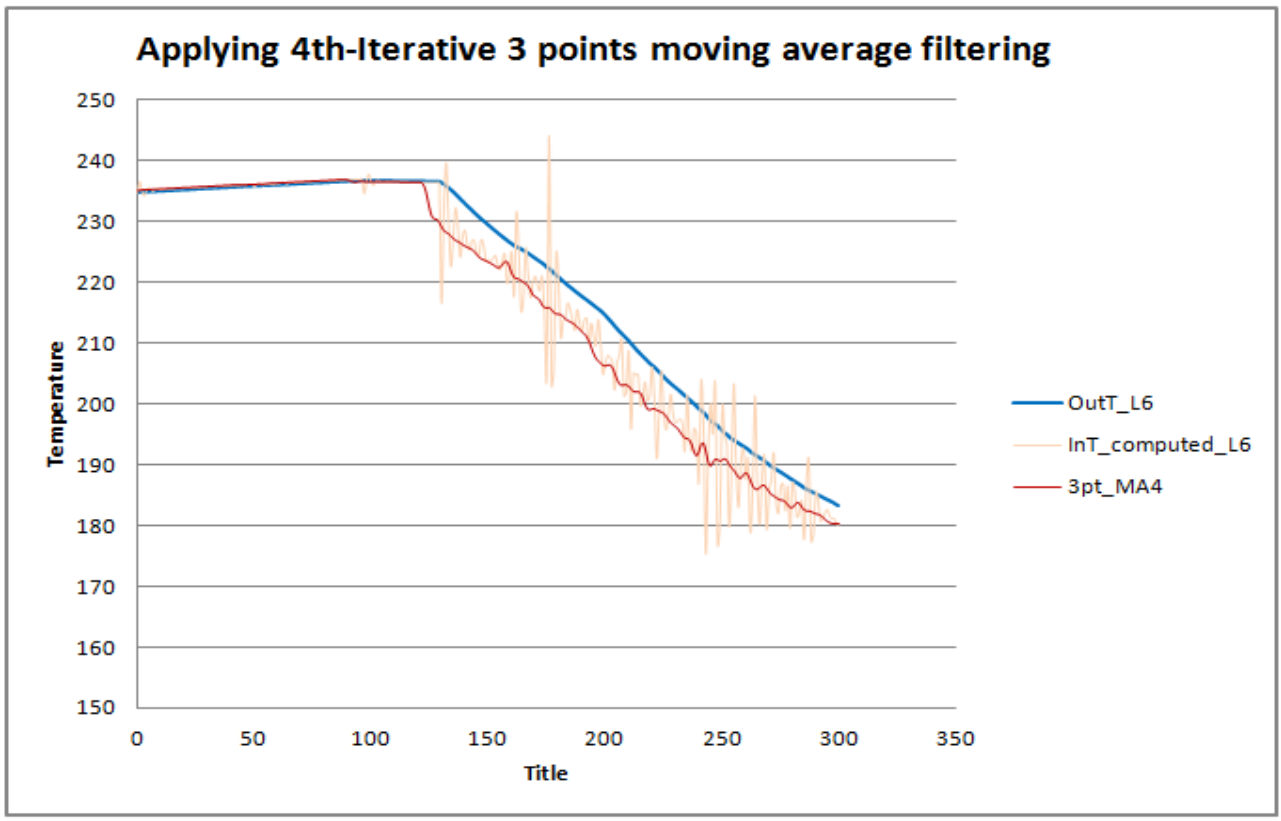


Figure 6.21: Filtering treatment of the noisiest input profile

6.7 Matrix Conditioning

One of the most important feature of this inverse algorithm is to solve a system of linear equation in each time step. In spite of having some error in the measured data or in b vector of the equation $A_{\emptyset}.x = b$, the role of matrix $A_{\emptyset}$ is significant. Generally a good conditioned matrix gives good result and vice versa. For inverse problems, role of the matrix $A_{\emptyset}$ becomes more significant; when the matrix conditionig is not excellent the system of equation may converge towards another set of solution indicating the non uniqueness of the system. It is observed that for matrix conditioning of condition number 14 gives non unique solution in each time step. So, before testing any problem, it is necessary to construct the operator matrix using ANSYS simulations data and to check what the conditioned number is. Since the geometry is simple , axisymmetric and it is comfortable to use structured grid, it is expected that the matrix conditioning would be very good. For instance, our first simulation with 6 zones results the matrix with condition number 1.026 which is excellent, and for 18 zones matrix it was around 1.62. It is worth to mention that condition number of a matrix is the ratio of highest eigenvalue to the lowest one, which indicates how good the invertability of the matrix . Interestingly, the resultant matrix $A_{\emptyset}$ from structured grid axisymmetric geometry maintans all the great properties of a good matrix; like strictly diagonaly dominance, symmetric positive definiteness and excellent conditioning indicating a cluster of eigenvalues.

6.8 Reducing noises by improving eigenvalues

Many characteristic behaviors including stability, uniqueness etc. when solving a linear system Ax = b are governed by the properties of the operator matrix A. Eigenvalues are the most representative entity of a matrix. We have discussed earlier that uniqueness of the solution depends on the condition number of a matrix, which is in fact the ratio of the highest and lowest eigenvalues. It is observed that noise behavior is also governed by the choice of Eigen properties of the operator matrix. Spectrum (set of eigenvalues) of the matrix A with smaller values introduces more noises in the solution than that of higher valued spectrum. In this real case problem we observed that the eigenvalues of the operator matrix $A_{\emptyset}$ is governed by the time step size, e.g. 2 sec time step size will cause $A_{\emptyset}$ with larger eigenvalues than that of 1 sec. In fact, in this problem, eigenvalues are actually related to the amount of maximum response at the outer surface due to reference temperature applied in the inner surface.

In all examples discussed before, the time step size in each case was 1 sec. In the last example of this real case problem, 1 sec step size leads to the spectrum of {1.0526, 1.0522, 1.0517, 1.0511, 1.0506 and 1.0507} whereas for 2 sec step size, the spectrum is {3.0863, 3.0859, 3.0852, 3.0846, 3.0829 and 3.0829}, which is almost 3 times increasing of eigenvalues. A plot with inner wall results with increased eigenvalues has been shown by the figure 6.22. If we compare the noise characteristic of increased spectrum with the original lower valued eigenvalues e.g. figure-6.22 with 6.17, we will clearly observe the reduction of noises. A separate plot comparing noises for

both cases is also shown by the figure 6.23. Red colored profiles are showing reduced noises in the temperature history.

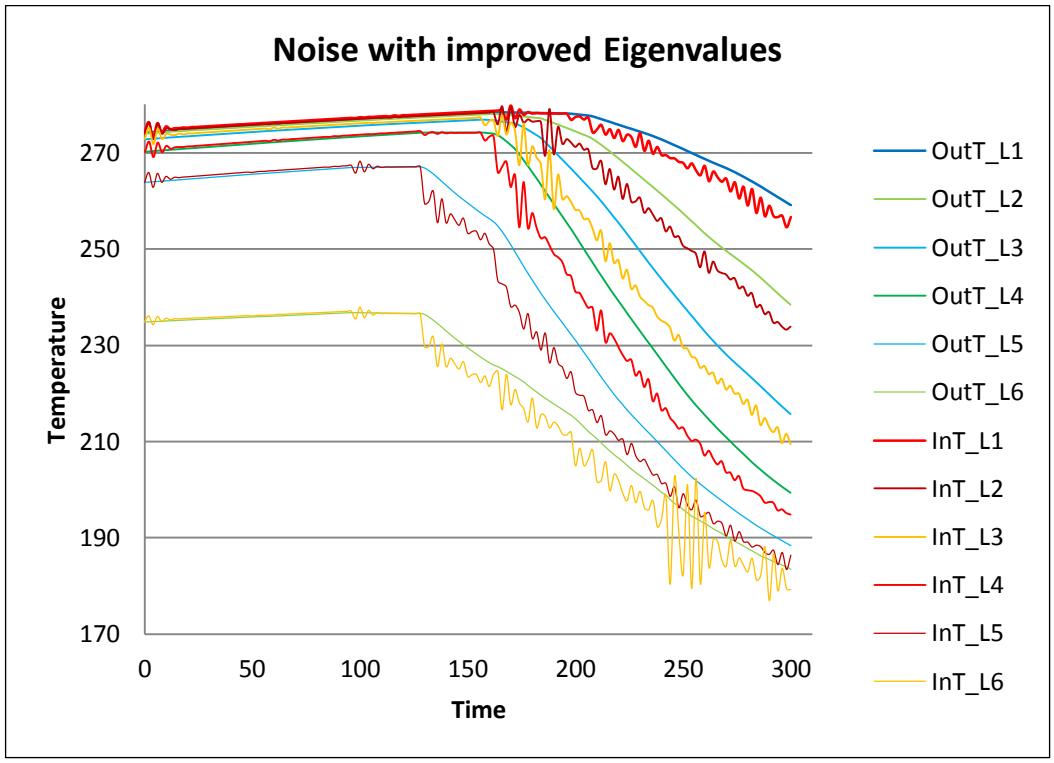


Figure 6.22: Noise reduction by increasing eigenvalues

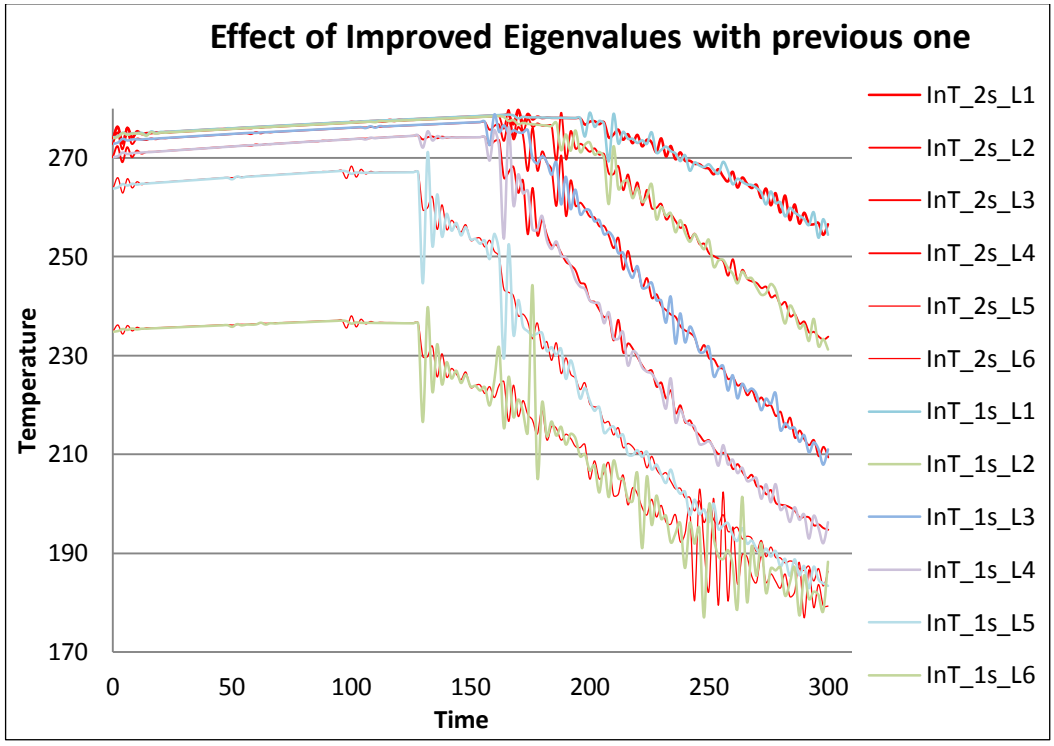


Figure 6.23: Comparison of noise character for lower and higher eigenvalues

The interesting thing, if we apply moving average filter on the reduced noisy data, we will find very smooth profiles as compared to others shown before, no iterative refinement of the MA filter is necessary. The newest profiles (blue-green) based on single time applied 3 points based MA filter are shown in the following graph 6.24.

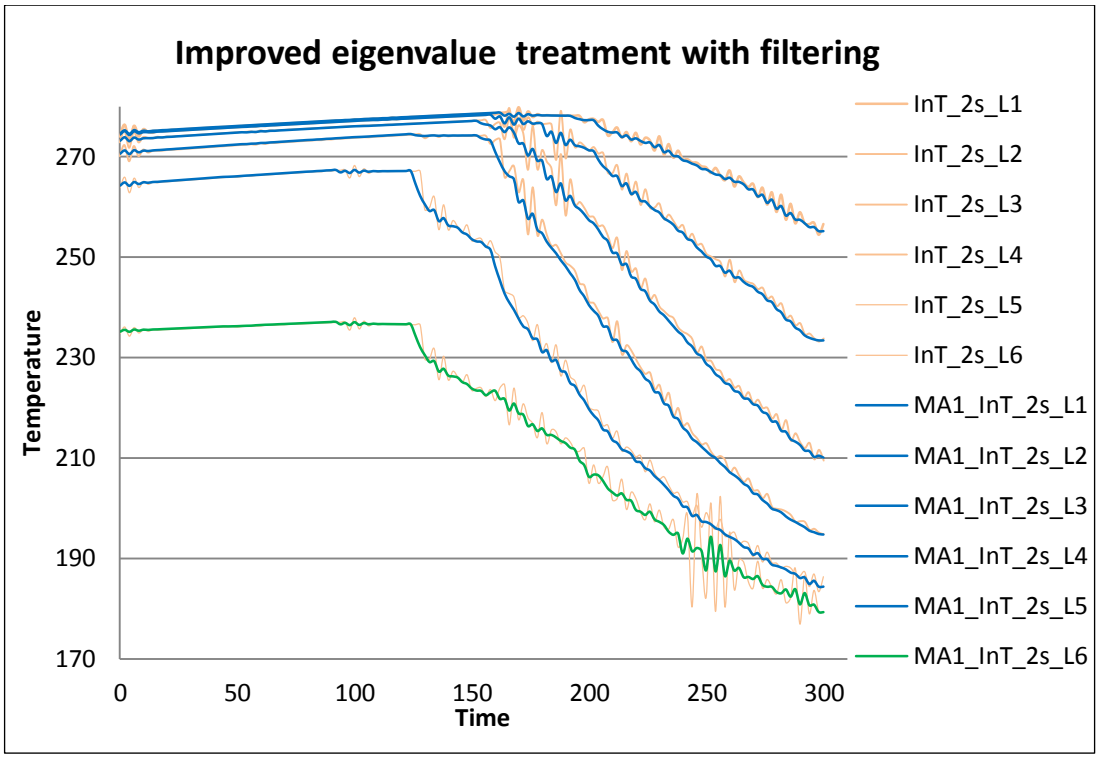


Figure 6.24: Profiles after increase of eigenvalues and filtering

6.9 Comparison of 2-D with 1-D

In this section, we will apply both 2-D and 1-D algorithm to a typical case for comparing inner wall temperature history. We want to see whether there is any significant difference in two computation methods. To do this, 10 mm thick pipe with 90 mm inner radius has been considered as standard typical case. It is already seen in section 6.4 that we have a simulation with nine different inner temperature profiles as shown earlier in the figure-6.8. So we have user chosen inner profiles and ANSYS simulated outer side history. These outer wall known data will be applied in the inverse algorithm with one zone (1-D) and 18 zones (2-D). For 1-D case, 18 different outer profiles will have to apply individually. To apply algorithm with one zone, a separate simulation has been performed to know response function of reference temperature change. Test results shows that there are significant differences, sometimes very big differences is observed. Two plots (6.22 and 6.23) are shown in this regard, where first one shows nine original input temperature profiles (blue-green) computed by 2-D algorithm with 18 inner profiles computed by 1-D algorithm. In the plot 6.22, blue-green colored profiles are for 2-D case while red-orange colored profiles indicate 1-D result. The second one (6.23) shows error

percentage for 1-D case. It is evident that error percentage could be even around 7% which is so significant in amount. From the error percentage chart, it can be easily concluded that bigger differences in neighboring inner temperature profiles will introduce more error when we will apply 1-D inverse algorithm. Another point can be easily pointed out that 1-D inverse algorithm causes fluctuated inner profile where dT/dt change is sharp, even the measured data are accurate, what is not appeared in multi-layers problem. It should be remember that this example for only 10 mm thick pipe. If thickness increases then the amount of error will be more since strong mixing zone is wider for thicker pipe. The route cause is 1-D algorithm completely disregards the consideration of inter-zone mixing of temperature profiles. The concluding point about error percentage in 1-D inverse algorithm is the combination of big differences in neighboring layers along with lower base temperature values like 40^{o} or 50^{o}. Nevertheless, here supplied outer surface history is almost accurate. For measured (erroneous) outer side data, this error will certainly be high for plug flow assumption.

Another plot (6.24) is also presented at the end of the section to have a better idea about error characteristics. Original outside history along with computed inner history for 1-D and 2-D and their respective error percentage are shown, which shows that error percentage is high at low temperature region where sudden change of dT/dt occurs.

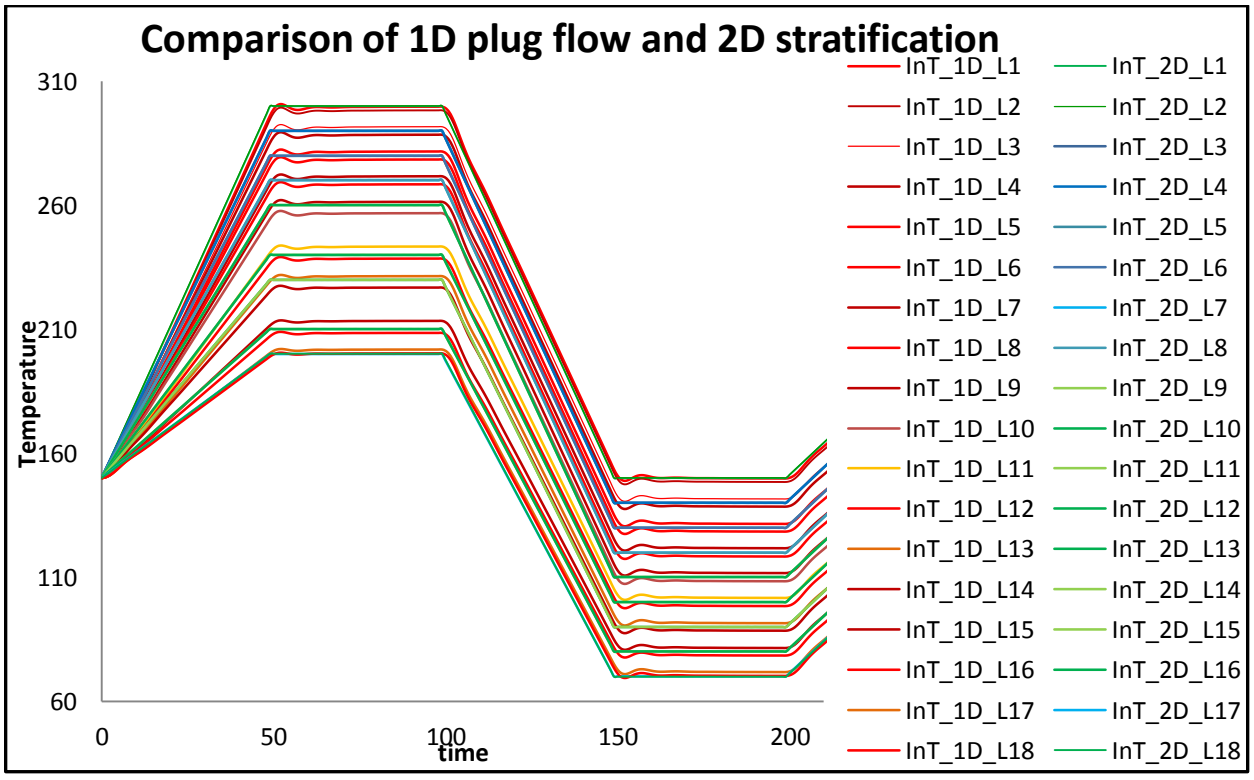


Figure 6.25: Comparison of plug flow assumption and 18L stratification (10 mm thick)

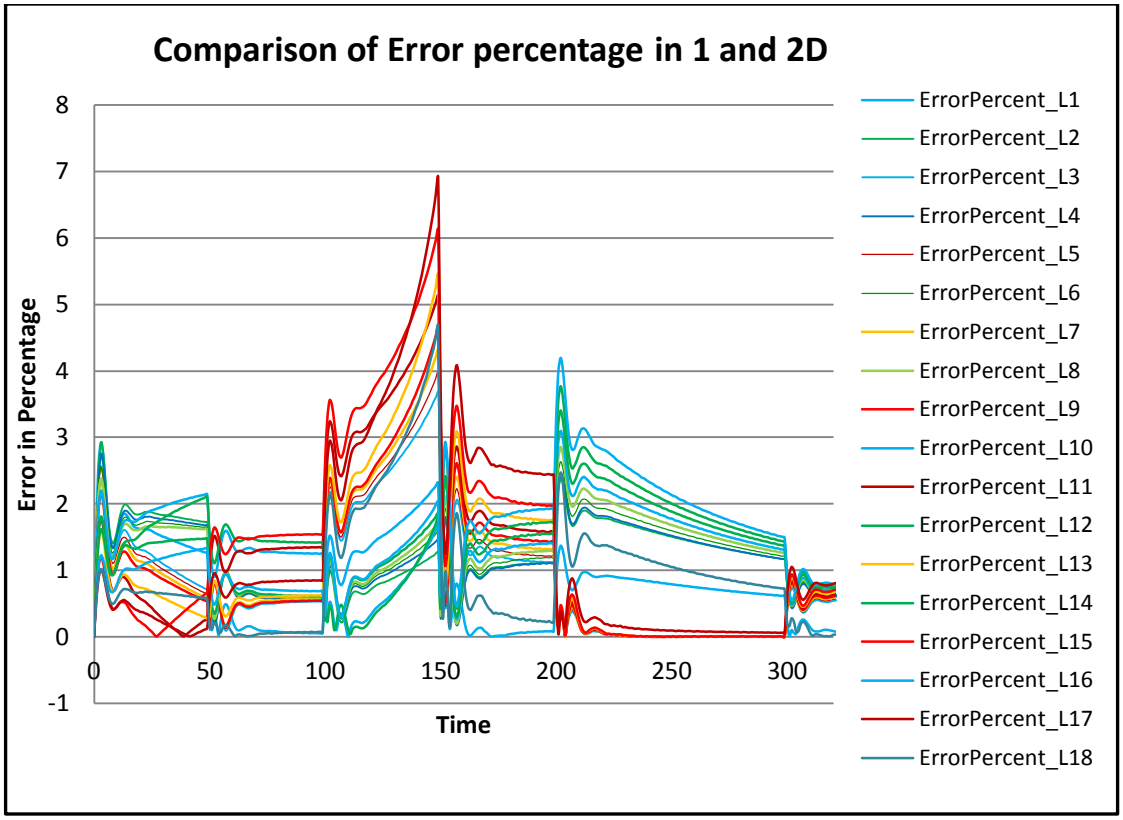


Figure 6.26: Error percentage for plug-flow event compared to 18L stratification

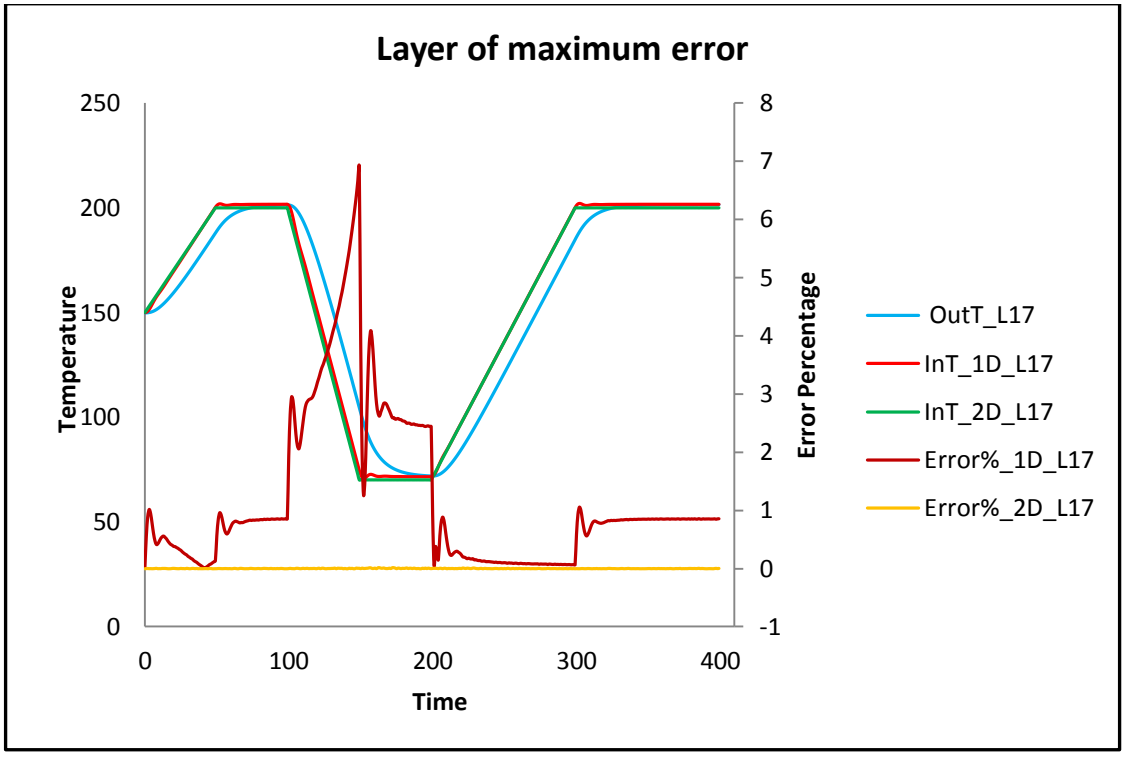


Figure 6.27: Comparison of error in 1-D and 2-D for the layer of maximum error

6.10 Regarding thick pipes

From the simulation of 30 mm thick pipe, we see that after 10 seconds there are some measurable responses at the outer surfaces. It is also noticeable that steady state time is around 600 seconds. On the other hand, for 40 mm pipes, the response and steady state time are even worse. As a result, we can possibly think that we can consider a time step of 10 seconds or more. But 10 seconds or higher time steps will not give the reasonable profile especially when the rate of change of temperature changes its direction. On the other hand, we showed that a better picture of inside profile demands at least 12 zones around the half circles. Considering both higher time step and higher number of zones would make some more complexity to apply these algorithms. However, an idea of dividing thickness of the pipe or dividing the problem into two parts was proposed in the previous work for one dimensional case. The same idea can be considered for higher dimensional inverse problems as well. The main point of dividing wall thickness is to work only with such smaller wall thickness, where the heat transfer process is quick and the corresponding intermediate boundary does not need a long period of time to detect changes in the inner-wall temperature. It should be noted that more oscillations are noticeable in the simulation results of this kind of problem even in one dimensional case. Moreover, testing with such thick pipe will essentially increase the inverse problem size for which an optimized coding of the current one is necessary. Considering the various difficulties like longer steady state time, higher number of zones, dividing the problem and hence multi-stage simulation of a single problem, and the limitations un-optimized code etc. the testing was not attempted, mainly for the fact that it need much time to test and study with different unexplored aspects.

6.11 Thermal Analysis Module (TAM) with C++

We have seen so far that developing an algorithm for higher dimension to evaluate thermal stratification phenomena by solving inverse heat conduction problem is the beginning of a big topic in the context of nuclear power plants components. To make the algorithm robust and to accommodate with current automated fatigue monitoring system in the nuclear plants, there are still lots of things to be done from hardware, analysis and software perspective. From analysis point of view some possible approaches are discussed in the whole work. Similarly, a software basis is also necessary to facilitate and accommodate further future research since the problem is considered a generalized one from the beginning.

To meet this purpose an object oriented reusable standardized module is necessary, and that's why an easy, reusable object oriented module, based on core scientific computing language C++, has been developed.

All headers are kept in a single project named thermal analysis module (TAM) since the module is very smaller. In the future, projects and modules can be added for other analysis as well like stress or fatigue or another possible numerical method. It should be noted that reusability and

user friendliness were considered with prime importance and optimization perspective has not been so far accounted except common sense optimization techniques. Object oriented paradigm is mostly a must for code reuse.

Structure of the module has been represented by a simple UML (unified modeling language) diagram. First we will see the main components of the project what is represented by simple block diagram 6.25. File management section is responsible for file reading, counting rows and columns, file copying and writing; computation is the mains component which is responsible for data type manipulation and algorithm implementation while user interface communicates with the user whether forward or inverse or both problems to be solved.

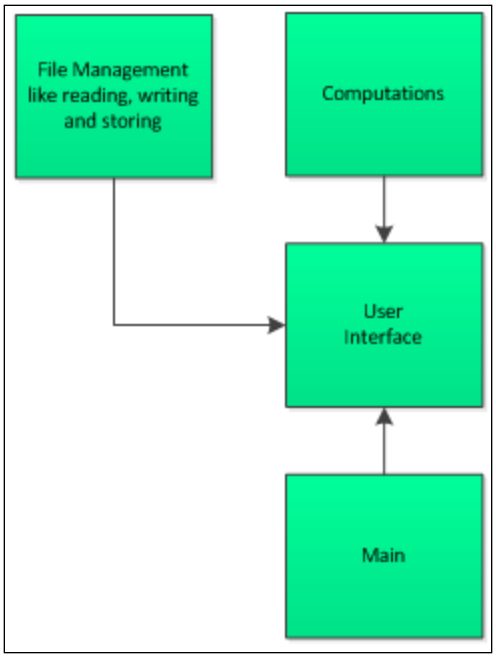


Diagram 6.28: main components of the module

Class diagram (6.26) shows that out of seven classes, fileReader.h and fileWriter.h are belonging to file manager section, while remaining five are responsible for computation. All data members and major function members of the corresponding classes are included in the class diagram, only inline functions are excluded. If we carefully observe the diagram we will see that only fileReader.h, fileWriter.h and Matrix.h classes can define data abstraction, e.g. only these three classes can create individual characteristic objects. Remaining classes create some scope with some functions attributes. Sometimes less objects offer better readability to a group of user, while more objects make the thing lengthier and complex. In the whole development, most coding standards have been maintained from the beginning. Prototypes of some classes will be included in the appendix.

It should be noted that 2-dimensional vector from standard library has been used to define a matrix. Throughout the program, 1 and 2-D vectors have been tried to use since standard library vector functions are highly optimized and vector itself a dynamic memory manageable object. Another important thing, for solving system of linear equation, Conjugate gradient solver has been implemented. The conjugate gradient method is a straightforward and powerful iterative technique for solving linear and nonlinear inverse problems. In this module, CG solver from gradient descent family is only valid for symmetric positive definite matrix. Simplified structure diagram of the module is shown below in the diagram 6.26.

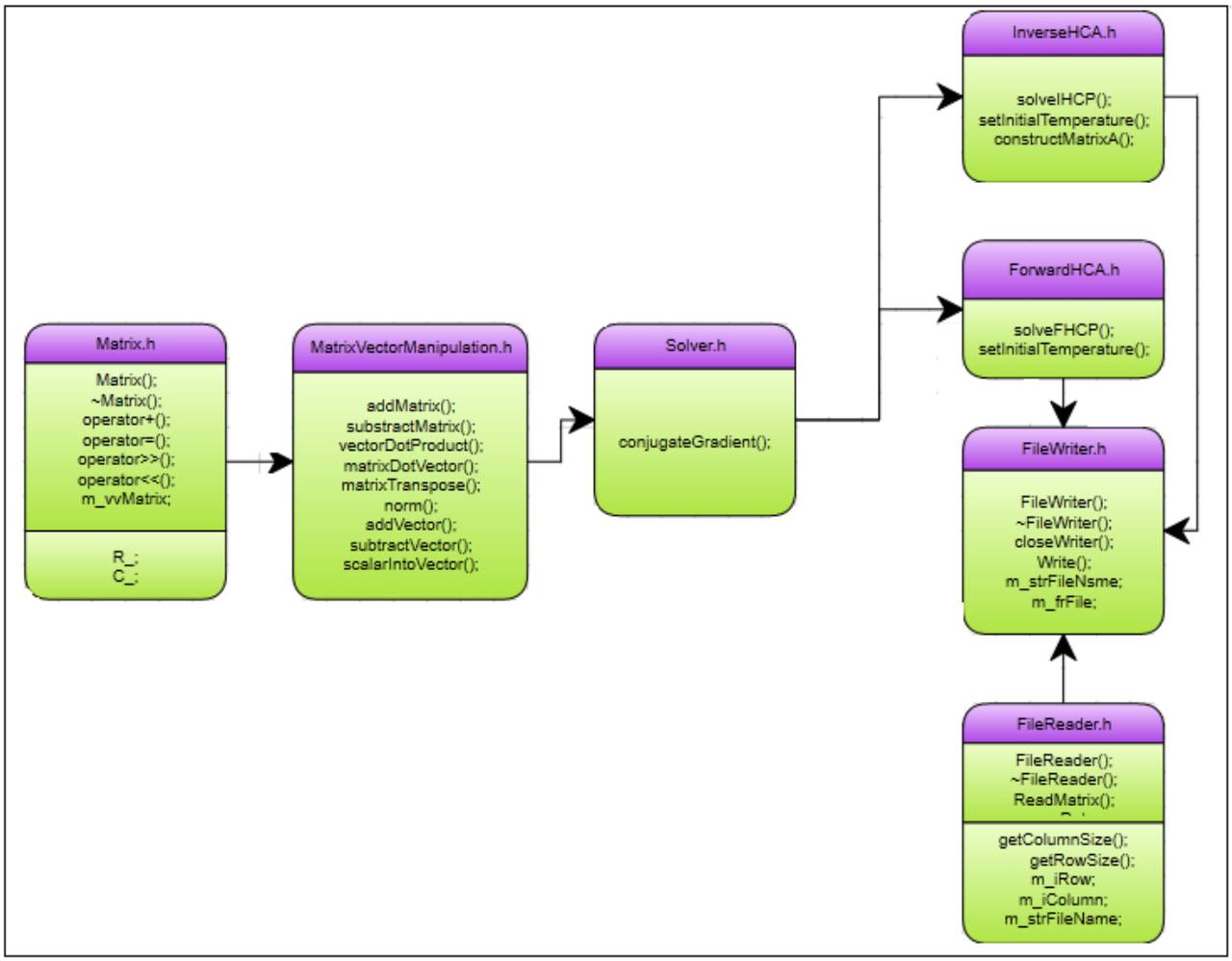


Diagram 6.29: structure or class diagram of the project

7. Conclusion and Further Scopes

In this concluding part, now we are going to summarize all possible important issues discussed so far and the possibility of further work on this topic.

1. Computing inner or inaccessible wall temperature history from know outer wall measured temperature profile is an inverse problem and hence it is ill-conditioned system which mean a small error in the measurement may often causes large amount of error. With some exceptions (very simple geometry or 1D problem), noise is expected in almost every practical case.
2. There are two types of treatment in the solution approaches of an inverse heat conduction problem, numerical treatment before computing inner history or after. The most popular treatment which is applied during the solution phase is known as regularization method, in which Tikhonov regularization method is recommended in the literatures as it is very powerful. After treatment is basically the filtering of the noises from the computed input history with oscillations. Moving average filter is reasonably good one whereas there might have more powerful filters than MA as well. Kalman filter might be a more effective one as prescribed by some researchers.
3. The ill-conditioning of a problem does not mean that a meaningful approximate solution cannot be computed. Rather ill-conditioning implies that standard methods in numerical linear algebra cannot be used in a straightforward way to compute such a solution instead, more sophisticated methods must be applied in order to ensure the computation of a meaningful solution. This is the essential goal of regularization methods.
4. For a pipe like axisymmetric geometry, the operator matrix $A_{\emptyset}$ coming from the response functions or matrix $\boldsymbol{R}^{m*nn}$ should be very good conditioning, more specifically the condition number should be below 2. If not, one can easily conclude that the simulation is somehow not very good. In such case inverse algorithm could result unexpected set of unknown parameter due to non uniqueness of the solution.
5. Forward algorithm will work perfectly once input data and reference simulated data are correct, since there is no complex computation. On the other hand, if there are some perturbations in the input data, still the forward algorithm will give some acceptable results. The first sentence is true for an inverse algorithm as well, however, not the second one.
6. There are a number of numerical methods to solve inverse heat conduction problem, unfortunately most of them are either cannot approximate solution correctly or limited to very specific and simple 1-D case. Trefftz function method has been recommended in the recent literature and is successfully applied with other methods in various cases indicating extended robustness in some degree. However, without applying regularization technique, noise free solution is not guaranteed.
7. Small errors in the system might come from various possible aspect of interpolation. Hence, interpolation of reference response data and very small time step selection should

be avoided during the ANSYS simulation part; number of zones is preferable in the range of 10 to 15; while the most dangerous thing cannot be avoided, is several interpolations in the FAMOS measured data.

8. Simulation time domain for the inverse algorithm now is restricted to 30 minutes for 6L problem which is essentially very small. A little optimization in the code e.g. using good memory management technique in 3-D matrix handling could give the ability to deal with possibly several hours.
9. Since the algorithm is generalized for one or more dimensions, if a good post treatment strategy is adopted, the algorithm can handle fairly complex shape as well.
10. For thermal fatigue assessment in the context of nuclear power plant components, simulating response functions in each fatigue relevant location for applying inverse algorithm is sometimes cumbersome and dealing with huge data might be hazardous. So it is probably better to adopt a numerical technique, which is comparatively robust, method of adjoin equation coupled with conjugate gradient or Trefftz function coupled with Tikhonov regularization might be two good choices.
11. Regarding the number of zones, it is necessary to comment that 6 thermocouple based system will not be so effective to solve multidimensional inverse heat conduction problem. Interpolation attempt of thermocouples data might introduce more uncertainty and oscillation in the solution. Depending on the thickness, number of zones should be in the range of 10 to 15 and thus required higher number of thermocouples is necessary instead of currently installed seven.
12. The algorithm is stable with small or comparatively bigger times step and can accommodate abrupt change in temperature as compared to the previous one dimensional algorithm.
13. The difference in inner temperature computation between 1-D and 2-D case is sometimes significantly high. If approximately 30^o temperature difference in two neighboring zones causes around 7% error in an ideal case then 100^o difference might cause more than 20% error in a practical case with bigger thickness.
14. An optimal eigenvalues treatment of the operator matrix $A_\emptyset$ can reduce noise in the results of an inverse algorithm, whereas improved eigenvalued with filtering treatment can give very good profiles.

At the end a short schematic flow chart like diagram (7.1) is also given below to get the whole picture of the work, where M is used for measured data and BC for inner surface boundary data. Starting with thickness 25 mm is an approximation, it might be more or less depending on thermo-physical properties.

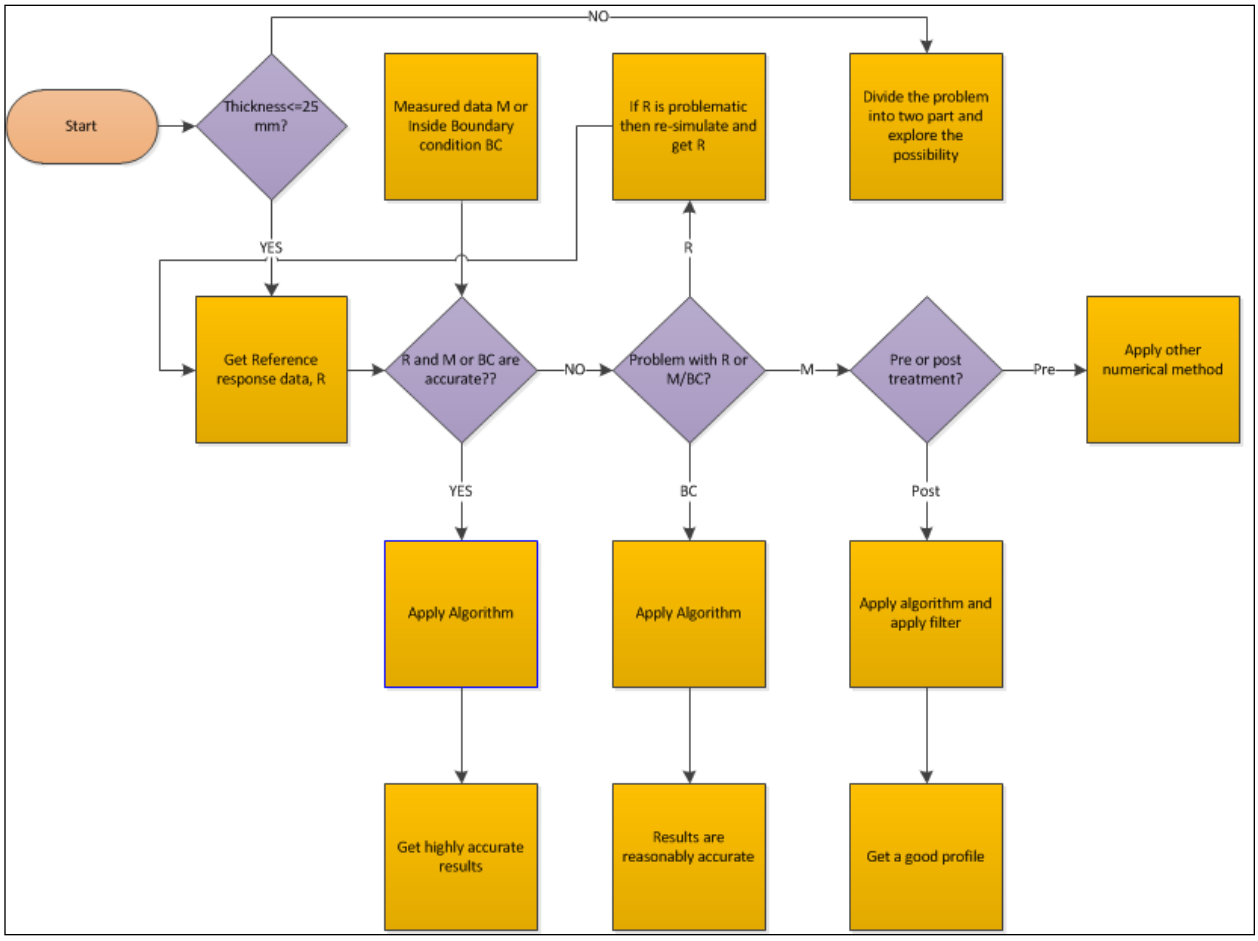


Diagram 7.1: Showing summary and further work scopes

8. References

[1] Reinhardt H. J., Hao D. N., Frohne J. & Suttmeier F. T., (2007), *Numerical Solution of Inverse Heat Conduction Problems in Two Spatial Dimensions*, Journal of Inverse and Ill-posed Problems, Volume 15, Issue 2, Pages 181-198, ISSN (online) 1569-3953, ISSN (Print) 0928-0219

[2] Chattopaddhay S., (2011), *Feed Water Line Cracking in Pressurized Water Reactor Plants*, Nuclear Power - Control, Reliability and Human Factors, Publisher: Intech, ISBN: 978-953-307-599-0

[3] Vargas R. G., (2010), *Solution of the inverse Fourier heat law for the automated fatigue assessment*, master thesis, AREVA GmbH

[4] Boley B. A. & Weiner J. H., (1960) *Theory of thermal Stresses*, John Wiley & Sons, New York, **ISBN-13:** 9780471086796

[5] Rudolph J., Bergholz S., Heinz B., and Jouan B., (2012) *AREVA Fatigue Concept – A Three Stage Approach to the Fatigue Assessment of Power Plant Components***,** "Nuclear Power Plants", book edited by Soon Heung Chang, ISBN 978-953-51-0408-7

[6] *Aging Management: AREVA Fatigue Concept***,** AREVA Nuclear Newsletter, April 2011

[7] Bouchet V., (2010), *Toolerstellung zur automatischen Ermüdungsberechnung, Diplomarbeit*, AREVA GmbH

[8] Jouan B., (2010), *Stress calculation for components under thermal and pressure Load with a unit transient approach*, PEEA-G/2010/EN/0282, AREVA GmbH

[9] Taler, J., Duda, P., (2006), *Solving Direct and Inverse Heat Conduction Problems*, Springer-verlag, ISBN: 978-3-540-33470-5 (Print) 978-3-540-33471-2 (Online)

[10] Beck J. V., BLACKWELL B., CLAIR C. R., (1985), *INVERSE HEAT CONDUCTION: Ill-posed Problems*, ISBN 0-471-08319-4, John Wiley & Sons, Inc.

[11] Wang L., Zhou X., Wei X., (2008), *Heat Conduction: Mathematical Models and Analytical Solutions*, ISBN-13 978-3-540-74028-5

[12] Grysa K., (2011), *Inverse Heat Conduction Problems*, Heat Conduction - Basic Research, Prof. Vyacheslav Vikhrenko (Ed.), ISBN: 978-953-307-404-7, InTech, Available from: http://www.intechopen.com/books/heat-conduction-basic-research/inverse-heat-conduction-problems

[13] Woo K. C. & Chow L. C., (1981), *Inverse Heat Conduction by Direct Inverse Laplace Transform*, Numerical Heat Transfer, 4:4, 499-504

[14] Bakushinsky, A. B. & Kokurin M. Yu., (2004), *Iterative Methods for Approximate Solution of Inverse Problems,* Springer, ISBN 1-4020-3121-1

[15] Hadamard, J., (1923), *Lectures on the Cauchy's Problem in Linear Partial Differential Equations,*Yale University Press, New Haven, recent edition: Nabu Press, 2010, ISBN 9781177646918

[16] Kurpisz, K. & Nowak, A. J., (1995), *Inverse Thermal Problems,* Computational Mechanics Publications, ISBN 1 85312 276 9, Southampton, UK

[17] Hohage, T., (2002), *Lecture Notes on Inverse Problems*, University of Goettingen. Date of access : June 30, 2011, Available from http://www.mathematik.uni-stuttgart.de/studium/infomat/Inverse-Probleme-Kaltenbacher-WS0607/ip.pdf

[18] Anderssen, R. S. (2005), *Inverse problems: A pragmatist's approach to the recovery of Information from indirect measurements, Australian and New Zealand Industrial and Applied Mathematics Journal* Vol.46, pp. C588--C622, ISSN 1445-8735

[19] Özisik, M. N. & Orlande, H. R. B., (2000), *Inverse Heat Transfer: Fundamentals and Applications,* Taylor & Francis, ISBN 1-56032-838-X, New York, USA

[20] Dinh Nho Hao, (1998), *Methods for Inverse Heat Conduction Problems*. Peter Lang Verlag, Frankfurt/Main, Bern, New York, Paris

[21] Alifanov O. M., (1994), *Inverse Heat Transfer Problems*, Springer

[22] Bangerth W., Hartmann R., Kanschat G., (2006), Dealii, Differential Equations and Analysis Library. http://www.dealii.org

[23] Baumeister J., (1987), *Stable solution of inverse problems*. Vieweg.

[24] J. V. Beck, B. Blackwell and C. R. St-Clair Jr., Inverse Heat Conduction: Ill-Posed Problems, Wiley Intersciences, New York, 1985.

[25] Dinh Nho Hao, Reinhardt H.-J., Jarny Y., (1998), A *variational method for multi-dimensional linear inverse heat conduction problems*. Matimyas matematika (Special Issue), 48-56.

[26] Dinh Nho Hao, (1992), *A noncharacteristic Cauchy problem for linear parabolic equations II: A variational method*. Numerical Functional Analysis and Optimization, 13 (5&6), 541- 564.

[27] JARNY Y., OZISIK M. N. & BARDON J. P., (1991), *A general optimization method using adjoint equation for solving multidimensional inverse heat conduction*, International Journal for Heat & Mass Transfer. Vol. 34, No. 11, P. 2911-2919

[28] Ling X., Atluri S.N., (2006), *Stability analysis for inverse heat conduction problems*, CMES, Vol.13, No.3, pp.219-228

[29] Stolz, G. (1960), *Numerical Solutions to an Inverse Problem of Heat Conduction for Simple Shapes*, Trans. ASME, J. Heat Transfer**,** vol. 82C, pp. 20-26.

[30] Beck, J. V. (1968), *Surface Heat Flux Determination using an Integral Method, Nuclear Engineering Design**,*** vol. ***7*** **,** pp. 170-1 78.

[31] Sparrow E. M., Haji-Sheikh A. and Lundgren T**.** S., (1964), *The Inverse Problem in Transient Heat Conduction*, Trans. ASME, J. Appl. Mech**.,** vol. 86E, pp. 369-375.

[32] Beck J. V., (1979.), *Criteria for Comparison of Methods of Solution of the Inverse Heat Conduction Problem*, Nucear Engineering Design**,** VOL. 53, ***PP.*** 11-22

[33] Trefftz, E. (1926), Ein Gegenstuek zum Ritz'schen Verfahren. *Proceedings of the 2nd International Congress of Applied Mechanics*, pp.131–137, Orell Fussli Verlag, Zurich

[34] Ciałkowski, M. J. & Grysa, K., (2010), *Trefftz method in solving the inverse problems, Journal of Inverse and Ill-posed Problems*, Vol.18, No.6, pp. 595–616, ISSN 0928-0219

[35] Hansen, P. C., (1992), *Analysis of discrete ill-posed problems by means of the L-curve*, SIAM Review, Vol.34, No.4, pp. 561–580, ISSN 0036-1445

[36] Hansen, P.C. & O'Leary, D.P., (1993), *The use of the L-curve in the regularization of discrete ill-posed problems*, *SIAM Journal of Scientific Computing*, Vol.14, No.6, pp. 1487–1503, ISSN 1064-8275

[37] Tikhonov, A. N. & Arsenin, V. Y. (1977), *On the solution of ill-posed problems*, John Wiley and Sons, ISBN 0-470-99124-0, New York, USA

[38] Kalman, R. E. (1960), *A New Approach to Linear Filtering and Prediction Problems*, Transactions of the ASME – Journal of Basic Engineering, Vol.82, pp. 35-45, ISSN 0021-9223

[39] Norton, J. P. (1986), *An Introduction to identification*, Academic Press, ISBN 0125217307, London

[40] Wikipedia, *http://en.wikipedia.org/wiki/Superposition_principle*

[41] Wikipedia, *http://en.wikipedia.org/wiki/Scilab*

[42] Jouan, B., (2010), *Calculation of inner wall temperature and stress time history for plugflow events in pipes with SINTOC-ETV*, PEEA-G/2010/en/0280, AREVA GmbH

[43] Jouan B., (2009), *Development of a calculation method to determine the stress due to an unsteady temperature state in power plant components*, master thesis, AREVA GmbH

[44] Busby H. R. & **D.** M. TRUJILLO, (1 985*), Numerical Solution to a Two-dimensional Inverse Heat Conduction Problem*, International Journal for Numerical Methods in Engineering**,** Volume 21, Pages 349-359

[45] Alnajem N. M., & Ozisik M. N., (1986), *A direct analytic approach for solving two-dimensional linear inverse heat conduction problems***,** Waerme-und Stofftibertragung 20, 89-96, Springer-Verlag

[46] H Reinhardt H. J., (1991), *A Numerical Method for the Solution of Two Dimensional Inverse Heat Conduction Problems*, International Journal for Numerical Methods in Engineering, Vol. *32, 363-383*

[47] BECK J. V., BLACKWELL B. & HAJI-SHEIKH A., (1996), *Comparison of some inverse heat conduction methods using experimental data***,** Int. J. Heat Mass Transfer. Vol. 39, No. 17, pp. 3649-3657

[48] Grysa, K., (2010), *Trefftz functions and their Applications in Solving Inverse Problems*, Politechnika więtokrzyska, PL ISSN 1897-2691

[49] Smith S. W., (2002), *Digital Signal Processing: A Practical Guide for Engineers and Scientist*, Demystifying Technology Series, Elsevier Science. ISBN: 0-750674-44-X

[50] http://home.comcast.net/~szemengtan/InverseProblems/chap3.pdf

9. Appendix

Appendix 1: ANSYS Scripts for 1-D Forward Heat Conduction Simulation

```
/CLEAR,START
/file,1DPipe

!=========================================
!Variable declaration and initialization

!film coefficient, see below for details
hf    =10000

!unit transient time steps
t1    =1e-9
t2    =1
t3    =200 !approximaltely after 100 sec, it becomes stable

!initial and final temp of UT
Ti    =0
Tut   =100

!Number of subdivision of lines
s1        =10
s2        =5
!================================================
!Preprocessing starts
/prep7
!const properties
MPTEMP, 1,  0.0000000
MPDATA,DENS,      1, 1, 7.930000000E-09,
MPTEMP, 1,  0.0000000
MPDATA,KXX ,      1, 1,  15.0000000
MPTEMP, 1,  0.0000000
MPDATA,C   ,      1, 1,  470000000.
!element type
et,1,PLANE77

!keypoints
k,1,0,0
k,2,10,0
k,3,10,5
k,4,0,5

!lines
l,1,2
```

```
l,2,3
l,3,4
l,4,1
alls

!area selection
lsel,s,,,1,4
al,all
numcmp,node !compressing node numbering

!subdivision of lines
lsel,s,,,1,
lsel,a,,,3
lesi,all,,,5
alls
amesh,all !meshing

!!specifying nodes for applying boundary conditions
lsel,s,,,4
cm,l_InnArea,line
nsll,s,1
cm,n_InnArea,node

lsel,s,,,1
lsel,a,,,3
cm,l_insulated,line
nsll,s,1
cm,n_insulated,node
alls
!=====================================================

!Solution part starts

/solu
antype,trans
kbc,0
outres,all,all
cmsel,s,n_InnArea
```

```
cmsel,a,n_insulated
sf,all,conv,hf,Ti

time,t1 !first load step        !starting time is 0 = 1e-9
timi,off
auto,off                        ! Reset the focus and distance specifications to "automaticaly
calculated"
alls
solve

timi,on
auto,on

time,t2 !second load step
cmsel,s,n_insulated
sf,all,conv,0,20
cmsel,s,n_InnArea
sf,all,conv,hf,Tut
deltime,.5,0.1,1
alls
solve

time,t3 !third load step
deltime,1,1,1 !now results are in each second???
alls
solve

fini
save
!================================================
!for post-processing
/post1
```

Appendix 2: ANSYS Scripts for a 2-D Forward Heat Conduction Simulation using SF (One simulation from dozens)

```
/CLEAR,START
/file,18L_9T
hf    =10000 !film coefficient

!time segments
t1    =1e-9
t2    =30
t3    =50
t4    =270
t5    =330
t6    =350
t7    =370
t8    =500

!Temperature profile
T1    =0
T2    =100
T3    =140
T4    =150
T5    =120
T6    =200
T7    =300

!inner radius and thickness
ri        =90
s         =10

!le is nos of sections along the thickness, le_2 is along the 90 degree arc
le=2
le_2=4

!Preprocessing part starting
/prep7

!const properties
```

```
MPTEMP, 1,  0.0000000
MPDATA,DENS,      1, 1, 7.930000000E-09,
MPTEMP, 1,  0.0000000
MPDATA,KXX ,      1, 1,  15.0000000
MPTEMP, 1,  0.0000000
MPDATA,C   ,      1, 1,  470000000.

!element type
et,1,PLANE77

!keypoints specifying
csys,1 !temporarily shifting into polar coordinate

!keypoints
k,1,0,0
*do,i,1,36
  k,i+1,ri,i*10-10
*enddo

*do,j,38,73
  k,j,ri+s,(j-38)*10
*enddo

!drawing circular lines
*do,i,1,35
  larc,i+1,i+2,1,ri
*enddo
larc,37,2,1,ri

*do,i,38,72
  larc,i,i+1,1,ri+s
*enddo
larc,73,38,1,ri+s

csys,0  !back into original coordinate system

!4 lines along the thickness
l,2,38
```

```
l,11,47
l,20,56
l,29,65

!indenfitying mesh area
alls
lsel,s,,,1,36
lsel,a,,,37,72
al,all

!subdividing lines for meshing
numcmp,node !compressing node numbering

lsel,s,,,1,36
lsel,a,,,37,72
lesi,all,,,le_2
lsel,s,,,73,76
lesi,all,,,le

alls
amesh,all

!!specifying zones
lsel,s,,,8,11
cm,l_zone1,line
nsll,s,1
cm,n_zone1,node

lsel,s,,,6,7
lsel,a,,,12,13
cm,l_zone2,line
nsll,s,1
cm,n_zone2,node

lsel,s,,,4,5
lsel,a,,,14,15
cm,l_zone3,line
nsll,s,1
```

cm,n_zone3,node

lsel,s,,,2,3
lsel,a,,,16,17
cm,l_zone4,line
nsll,s,1
cm,n_zone4,node

lsel,s,,,36
lsel,a,,,1
lsel,a,,,18,19
cm,l_zone5,line
nsll,s,1
cm,n_zone5,node

lsel,s,,,34,35
lsel,a,,,20,21
cm,l_zone6,line
nsll,s,1
cm,n_zone6,node

lsel,s,,,32,33
lsel,a,,,22,23
cm,l_zone7,line
nsll,s,1
cm,n_zone7,node

lsel,s,,,30,31
lsel,a,,,24,25
cm,l_zone8,line
nsll,s,1
cm,n_zone8,node

lsel,s,,,26,29
cm,l_zone9,line
nsll,s,1
cm,n_zone9,node
alls

```
!solution phase
/solu

antype,trans
kbc,0
outres,all,all


cmsel,s,n_zone1
sf,all,conv,hf,200
cmsel,s,n_zone2
sf,all,conv,hf,190
cmsel,s,n_zone3
sf,all,conv,hf,180
cmsel,s,n_zone4
sf,all,conv,hf,170
cmsel,s,n_zone5
sf,all,conv,hf,160
cmsel,s,n_zone6
sf,all,conv,hf,140
cmsel,s,n_zone7
sf,all,conv,hf,130
cmsel,s,n_zone8
sf,all,conv,hf,110
cmsel,s,n_zone9
sf,all,conv,hf,100


time,t1         !starting time is 0 = 1e-9
timi,off        !Timint,key,lab>> Turns on transient effects, off menas steady state
auto,off        !Reset the focus and distance specifications to "automaticaly calculated"
alls
solve


timi,on
auto,on

time,50
```

cmsel,s,n_zone1
sf,all,conv,hf,300
cmsel,s,n_zone2
sf,all,conv,hf,290
cmsel,s,n_zone3
sf,all,conv,hf,280
cmsel,s,n_zone4
sf,all,conv,hf,270
cmsel,s,n_zone5
sf,all,conv,hf,260
cmsel,s,n_zone6
sf,all,conv,hf,240
cmsel,s,n_zone7
sf,all,conv,hf,230
cmsel,s,n_zone8
sf,all,conv,hf,210
cmsel,s,n_zone9
sf,all,conv,hf,200
deltime,1,1,1
alls
solve

time,100
deltime,1,1,1
alls
solve

time,150
cmsel,s,n_zone1
sf,all,conv,hf,150
cmsel,s,n_zone2
sf,all,conv,hf,140
cmsel,s,n_zone3
sf,all,conv,hf,130
cmsel,s,n_zone4
sf,all,conv,hf,120
cmsel,s,n_zone5

sf,all,conv,hf,110
cmsel,s,n_zone6
sf,all,conv,hf,100
cmsel,s,n_zone7
sf,all,conv,hf,90
cmsel,s,n_zone8
sf,all,conv,hf,80
cmsel,s,n_zone9
sf,all,conv,hf,70
deltime,1,1,1
alls
solve

time,200
deltime,1,1,1
alls
solve

time,300
cmsel,s,n_zone1
sf,all,conv,hf,280
cmsel,s,n_zone2
sf,all,conv,hf,270
cmsel,s,n_zone3
sf,all,conv,hf,260
cmsel,s,n_zone4
sf,all,conv,hf,250
cmsel,s,n_zone5
sf,all,conv,hf,240
cmsel,s,n_zone6
sf,all,conv,hf,230
cmsel,s,n_zone7
sf,all,conv,hf,220
cmsel,s,n_zone8
sf,all,conv,hf,210
cmsel,s,n_zone9
sf,all,conv,hf,200
deltime,1,1,1

alls
solve

time,400
deltime,1,1,1
alls
solve

fini
save

!for post processing
/post1

Appendix 3: ANSYS Scripts for a 2-D Forward Heat Conduction Simulation using D (one simulation code from many)

```
/CLEAR,START
/CONFIG,NRES,50000 !nos of resulf set to be put in the result file

/file,L1From12L

!film coefficient, see below for details
!hf    =10000

!unit transient time steps
t1    =1e-9
t2    =1
t3    =1000!approximaltely after 100 sec, it becomes stable

!initial and final temp of UT
Ti    =0
Tut    =100
!inner radius and thickness
ri        =80
s         =30

!le is nos of sections along the thickness, le_2 is along the 90 degree arc
le=15
le_2=6

!Preprocessing part
/prep7

!const properties
MPTEMP, 1,  0.0000000
MPDATA,DENS,      1, 1, 7.930000000E-09,
MPTEMP, 1,  0.0000000
MPDATA,KXX ,      1, 1,  15.0000000
MPTEMP, 1,  0.0000000
MPDATA,C   ,      1, 1,  470000000.
```

```
!element type
et,1,PLANE55

!keypoints specifying
csys,1 !temporarily shifting into polar coordinate

!keypoints
k,1,0,0
*do,i,1,24
  k,i+1,ri,(i-1)*15
*enddo

*do,j,26,49
  k,j,ri+s,(j-26)*15
*enddo

!drawing circular lines
*do,i,1,23
  larc,i+1,i+2,1,ri
*enddo
larc,25,2,1,ri

*do,i,26,48
  larc,i,i+1,1,ri+s
*enddo
larc,49,26,1,ri+s

csys,0  !back into original coordinate system

!lines along the thickness
*do,i,2,25
  l,i,i+24
*enddo

alls
!indenfitying area
*do,i,1,23
  al,i,i+24,i+48,i+49
```

```
*enddo
al,24,72,48,49

!subdividing lines for meshing
numcmp,node !compressing node numbering
lsel,s,,,1,48
lesi,all,,,6

lsel,s,,,49,72
lesi,all,,,10

alls
amesh,all

!!specifying zones
lsel,s,,,6,7
cm,l_area1,line
nsll,s,1
cm,n_area1,node

lsel,s,,,1,5
lsel,a,,,8,24
cm,l_OtherInnerNodes,line
nsll,s,1
cm,n_OtherInnerNodes,node
alls

!Solution Part
/solu
antype,trans
kbc,0
outres,all,all

cmsel,s,n_area1
cmsel,a,n_OtherInnerNodes
D,all,temp,0


time,t1                !starting time is 0 = 1e-9
```

```
timi,off          !Timint,key,lab>> Turns on transient effects, off menas steady state
auto,off          !Reset the focus and distance specifications to "automaticaly calculated"
alls
solve


timi,on
auto,on

time,1
cmsel,s,n_area1
D,all,temp,100
cmsel,s,n_OtherInnerNodes
D,all,temp,0
deltime,0.5,0.1,1
alls
solve

time,t3
deltime,1,1,1 !now results are in each second
alls
solve

fini
save
```

Appendix 4: Scilab Code for Forward Heat Conduction algorithm

```
clear;
stacksize('max');                //for maximum possible stack size
initialTemp=0;
temp_ref_inv = 1.0/100.0;        //reference temperature of the simulation run with ANSYS

input_temp = fscanfMat('Input_Temp.txt');//fscanMat*() returns a matrix from file
ref_temp = fscanfMat('Ref_Out_Temp.txt');

Time = size(input_temp,"r");    // 'time' is the total time for the simulation, size() returns nos of rows or columns from a matrix
responseTimeOfUT = size(ref_temp,"r");
Layer= size(input_temp,"c");    //Number of zones

UT_outwall_temp(Time,Layer,Time)= 0;    //3D matrix for storing calculated valued per timestep
UT_outwall_temp(:,:,:)= 0;  //initializing with 0
OutWall_temp(Time,Layer)=0;
OutWall_temp(:,:)=initialTemp;      //2D matrix, for storing final results

//pause;                         //used for dubugging purpose

//main part of the algorithm
for i = 1 : Time-1               //third dimension of the matrix to be calculated
   for j = 1 : Layer
      for k = i : i+responseTimeOfUT-1
         if k <= Time then
            for l = 1 : Layer
               //=+ is not supported
               UT_outwall_temp(k,j,i)= //intentionally removed for confidential reason
               //pause;
            end
         end
      end
   end
end
```

```
//pause;

//calculating total history
for i = 1 : Time
   for j = 1 : Layer
      for k = 1 : Time
         OutWall_temp(i,j)= //intentionally removed for confidential reason
      end
   end
end

print('output_temp.txt',OutWall_temp);       //saving into a file
```

Appendix 5: MATLAB Code for Inverse Heat Conduction algorithm*

*[Tested with academic version of the University of Erlangen-Nuremberg]

```
clear;
temp_ref = 100.0; %reference temperature of the simulation run with ANSYS
temp_ref_inv = 1.0/100.0;

OutWall_temp = importdata('outWall_temp.txt'); % returns a matrix after reading from file
OutT_of_UT = importdata('Ref_Out_Temp.txt');

[Time,x] = size(OutWall_temp);      % 'time' is the total time for the simulation, size() returns
nos of rows or columns from a matrix
[responseTimeOfUT,x] = size(OutT_of_UT);
[x,Layer]= size(OutWall_temp);
Layer=Layer-1;

%setting initial condition
initialTemp(Layer)=0;
for i = 1 : Layer
   initialTemp(i)=OutWall_temp(1,i+1);
end

%required matrces memory allocation
UT_influenced_outwall_T(Time,Layer*Layer,Time)= 0;  %3D matrix for storing calculated
valued per timestep
UT_influenced_outwall_T(:,:,:)= 0; %initializing with 0
DelInnWall_temp(Time-1,Layer)=0;
DelInnWall_temp(:,:)=0; %2D matrix, for storing final results

%constructing matrix A from Ax=b
A(Layer,Layer)=0;
for i = 1 : Layer
   for j = 1 : Layer
      A(i,j) =  % removed intentionally for confidential reason
   end
end

%initializing b with 0
b(Layer)=0;
```

```
%computations starts
for i = 1 : Time-1 %3rd dimension of the 3D matrix
   for j = 1 : Layer %Constructing b starts
      temporary=0.0;
      if i>1
         for k = 1 : i-1
            for l = 1 : Layer
               temporary =  % removed intentionally for confidential reason
            end
         end
      end

      b(j)= % removed intentionally
   end

   T = A\b';     %solver will automatically be selected, solving Ax=b

   %storing solutions
   for m = 1:Layer

      DelInnWall_temp(i,m)= T(m);
   end

   %forward algorithm starts
   for j = 1 : Layer*Layer

      for k = i+1 : i+responseTimeOfUT-1
         if k <= Time
            if Layer>1
               UT_influenced_outwall_T(k,j,i)= % removed intentionally for confidential reason
            end
         end
      end
   end

end
```

```
%setting initial condition to the result matrix
InnWall_Temp(Time,Layer)=0;
for it = 1 : Time
   for jt = 1 : Layer
      InnWall_Temp(it,jt)=initialTemp(jt);
   end
end

%completing resultant matrix with each second changed Temperature
for m = 2 : Time
   for n =1 : Layer
      InnWall_Temp(m,n) = DelInnWall_temp(m-1,n);
      InnWall_Temp(m,n)= % removed intentionally for confidential reason
   end
end

%Writing into a text file
save('input_temp.txt','InnWall_Temp','-ASCII');
```

Appendix 6 : C++ Class Prototype of Matrix.h in TAM

```
#ifndef _MATRIX_H
#define _MATRIX_H

#include <vector>
#include <algorithm>
#include <cassert>
#include <cstddef>
#include <cstdlib>
#include <fstream>
#include <iomanip>
#include <iostream>
#include <stdexcept>
#include "FileReader.h"

typedef std::vector<double>    dVector1D;

// Definition of the Matrix class
class Matrix
{
  public:

      Matrix();// Default constructor

      Matrix( FileReader& fm ); // Constructor for a matrix of size m x n

      ~Matrix(); // Destructor

      Matrix& operator=( const Matrix& rhs );// Copy assignment operator

      // Matrix writer to file
      void writeMatrixIntoFile( const std::string& outFile, const Matrix& m );

      // Matrix output file operator
      friend std::ofstream& operator<<( std::ofstream& ofs, const Matrix& m )
      {
            //implementation removed for confidential reason
      }

      // Matrix input file operator
      friend std::ifstream& Matrix::operator>>(std::ifstream& ifs, Matrix& m )
      {
            // implementation removed for confidential reason
      }

      // Rows function
      inline std::size_t rows() const
      {
            return m_;
      }
      // Columns function
      inline std::size_t columns() const
      {
            return n_;
      }
      // returning data of 2d vector
```

```
      inline const std::vector<dVector1D>& get2dVector() const
      {
            return this->m_vvMatrix;
      }

      // Matrix addition operator
      const Matrix operator+( const Matrix& rhs )
      {
            // implementation removed for confidential reason
      }

  private:
      std::size_t m_; // The current number of rows of the matrix.
      std::size_t n_; // The current number of columns of the matrix.
  public:
      std::vector<dVector1D> m_vvMatrix;
};
#endif
```

Appendix 7: C++ Class Prototype of FileReader.h in TAM

```
#ifndef _FILEREADER_H
#define _FILEREADER_H

#include <windows.h>
#include <stdlib.h>
#include <iostream>
#include <vector>
#include <fstream>
#include <string>
#include <limits>

typedef std::vector<std::string>    sVector1D;
typedef std::vector<double> dVector1D;
typedef std::vector<dVector1D> dVector2D;
typedef std::vector<dVector2D> dVector3D;

class FileReader
{
      public:
      //constructor to construct file objects from different possibilities
            FileReader();
            FileReader(const std::string&);

            FileReader(const std::string&, const std::size_t, const
std::size_t );
            //FileManager(const string&, const vector<doubleVector>& );
            ~FileReader(); //destructor

            //supporting function members
            int ReadMatrix();
            inline size_t GetNumRows(){ return m_iNumRows;}
            inline size_t GetNumColumns() { return m_iNumColumns; }
            void Clear();

// data members
      private:
            std::size_t getColumnSize();
            std::size_t getRowSize();

      public:
            std::vector<dVector1D>  m_vData;
      private:
            //stringVector                m_vTypes;
            std::string                         m_strFile;
            std::size_t                         m_iNumRows;
            std::size_t                         m_iNumColumns;

};
#endif //_FILEREADER_H
```

Appendix 8: C++ Class Prototype of MatrixVectorManipulation.h in TAM

```
#ifndef _MATRIXVECTORMANIPULATIONS_H
#define _MATRIXVECTORMANIPULATIONS_H

//this header has been defined to use standard library vector as 2 and 3D
//vector to facilitate using highly optimized vector functions
#include<vector>
#include "Matrix.h"
typedef std::vector<dVector1D>    dVector2D;

//matrix addition
const dVector2D addMatrix(const dVector2D& , const dVector2D& );

//matrix subtraction
const dVector2D subtractMatrix(const dVector2D&, const dVector2D& );

//vector dot product
const double vectorDotProduct(const dVector1D& , const dVector1D& );

//matrix vector product
const dVector1D matrixDotVector(const dVector2D&, const dVector1D& );

//for transposing a matrix
const dVector2D& matrixTranspose( dVector2D& );

//to calculate norm of a vector
const double norm(const dVector1D&);

//adding two vectors
const dVector1D addVector(const dVector1D& , const dVector1D& );

//subtracting two vectors
const dVector1D subtractVector(const dVector1D& , const dVector1D& );

//multiplying a scalar with vector
const dVector1D scalarIntoVector(const dVector1D& , const double& );


#endif
```