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.

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

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

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

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

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:

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

Where, Δ***T(r, t)*** is the temperature change in ***t*** time. If we can calculate this Δ***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:

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

Or, in simpler form:

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

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

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

………

……………..

⇨ $T(r,t) = T_o(r,0) + \sum_{i=1}^{m} \Delta T(r, t_i) = T(r,t_{m-1}) + \Delta T(r, t_m)$ ………. (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 $m' \epsilon$ 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 $\Delta T\ (\partial\Omega_i, t_2)$

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

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

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

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

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

……….

…………

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

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

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

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

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

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

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

$$\Rightarrow \quad T(r,t) = T(r,0) + \Delta T(r, t) = T(r,0) + \sum_{i=1}^{m} \Delta T(r, t_i)$$

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

⇨ $$\boxed{\boldsymbol{T(r,t) = T(r,0) + \sum_{i=1}^{P}\sum_{i=1}^{m}\sum_{k=1}^{m-j+1}\Delta T_i^r\left(r,\lambda_i,t_k\right)}} \qquad (4.8)$$

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:

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

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 ΔT^{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.

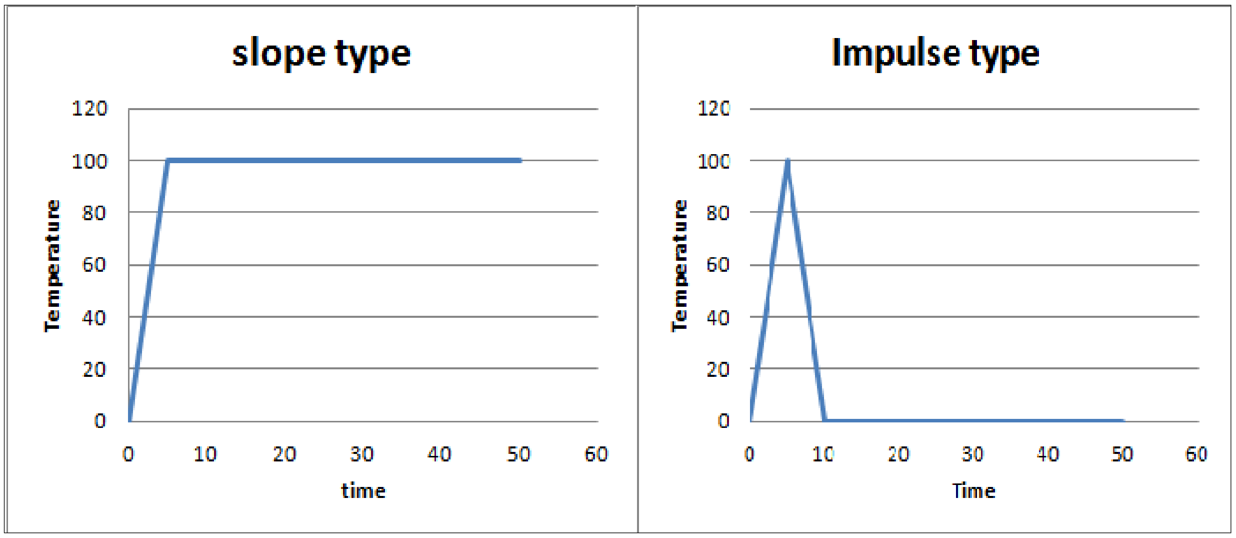


Fig-4.1: Slope and impulse type reference temperature change

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.

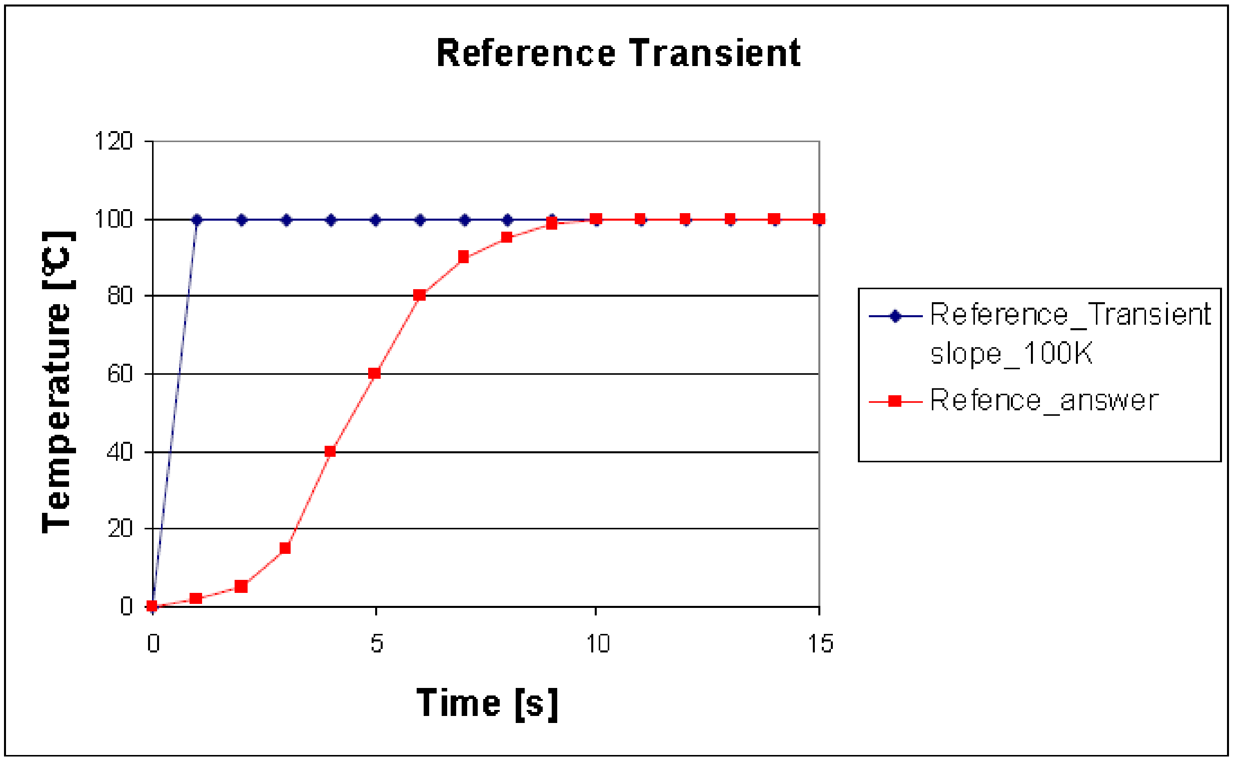


Fig.4.2: Slope type reference temperature profile and its typical response for a 2-D pipe[42]

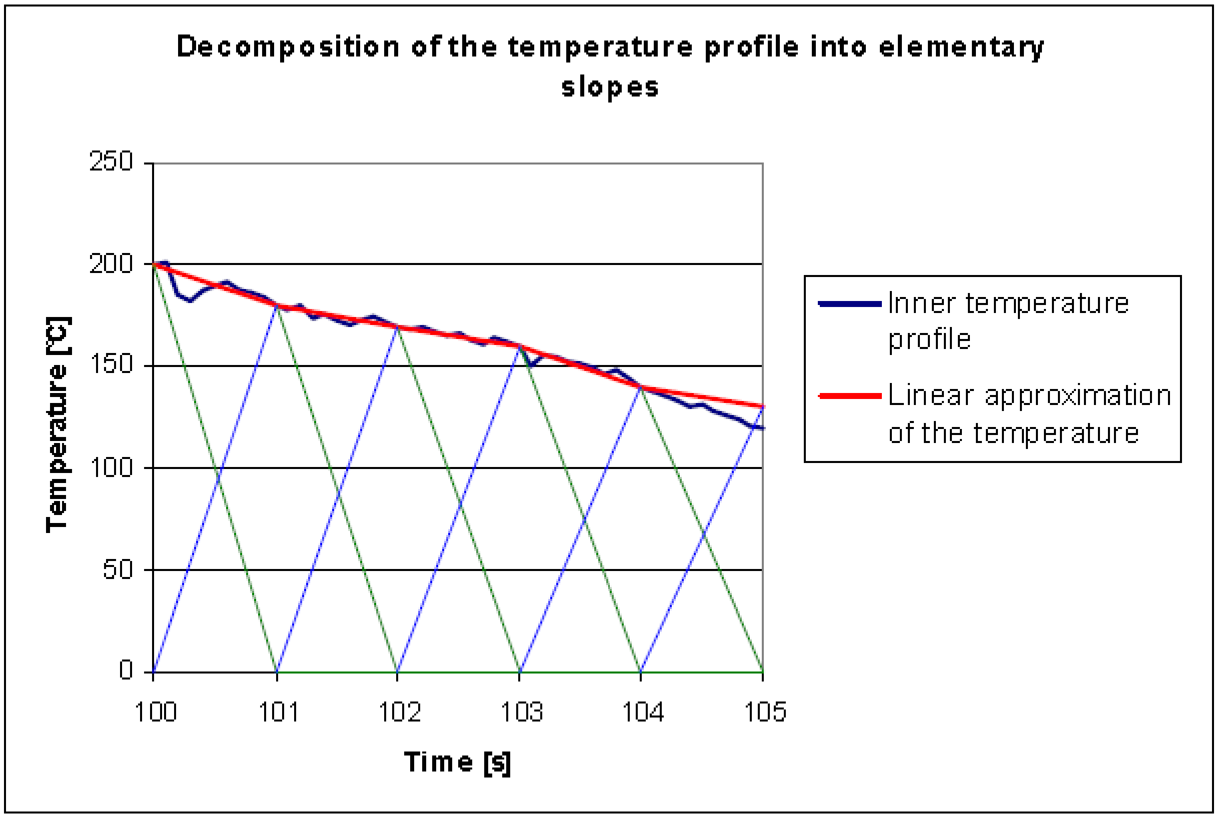


Fig.4.3: Decomposition of impulse type reference temperature[42]

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.

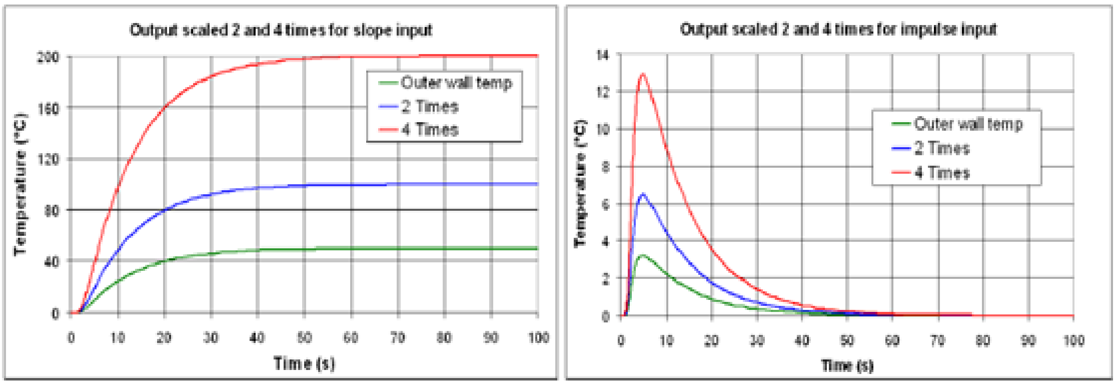


Fig.-4.4: Showing scalability in slope and impulse type ΔT^{ref} [3]

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 $\boldsymbol{\Omega_n}$, $\boldsymbol{n=\{1,2,\ldots..P\}, P\epsilon N}$

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 ***m*n*** entries, where **m** is the total number of global time steps and ***n*** is the number of concerned points.

Step-4: Assemble the matrices into a single matrix where number of rows will be ***m*** and number of columns will be ***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 $\boldsymbol{R^{m*nn}= R_1^{m*n}} \cup \boldsymbol{R_2^{m*n}} \cup \boldsymbol{\ldots\ldots} \cup \boldsymbol{R_P^{m*n}}$

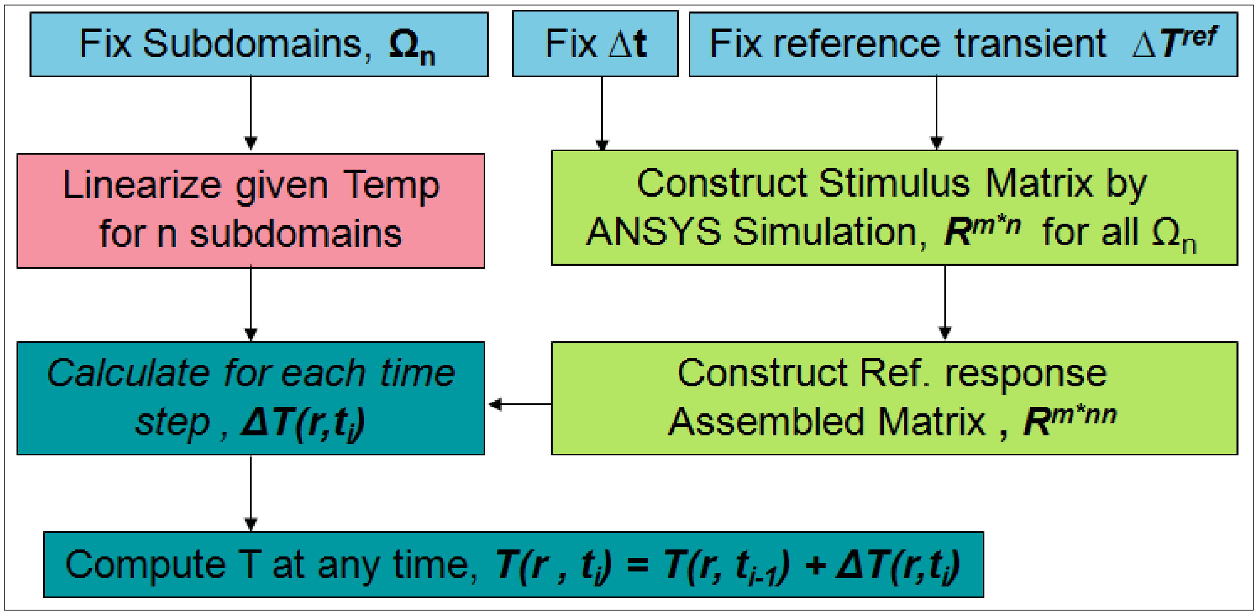
Step-5: Linearize $\boldsymbol{T(r,t)}|_{\partial\Omega}$ for all sub-domain $\boldsymbol{\Omega_n}$ according to $\Delta \boldsymbol{t}$ and store into another matrix $\boldsymbol{U_{in}^{m*n}}$ that will be treated as input temperature matrix for forward problem.

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

Step-7: Compute needed temperature at point ***r***, ***T(r , t$_i$) = T(r, t$_{i-1}$) + ΔT(r, t$_i$)***

Step-8: Compute temperature for all required points

Let's the reference temperature ΔT^{ref} and its response function $\Phi_i^r(\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{t_s})$ 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:



Fix Subdomains, Ω_n
Fix Δt
Fix reference transient ΔT^{ref}
Linearize given Temp for n subdomains
Construct Stimulus Matrix by ANSYS Simulation, R^{m*n} for all Ω_n
Calculate for each time step , $\Delta T(r,t_i)$
Construct Ref. response Assembled Matrix , R^{m*nn}
Compute T at any time, $T(r , t_i) = T(r, t_{i-1}) + \Delta T(r,t_i)$

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 $\frac{\partial T(x,t)}{\partial n} = \mathbf{0}$ respectively, where ***T$_b$(x,t)*** is temperature distribution during ***t*** as shown in the following figure 6.6 and $\frac{\partial T(x,t)}{\partial n} = \mathbf{0}$ 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:

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

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

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

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

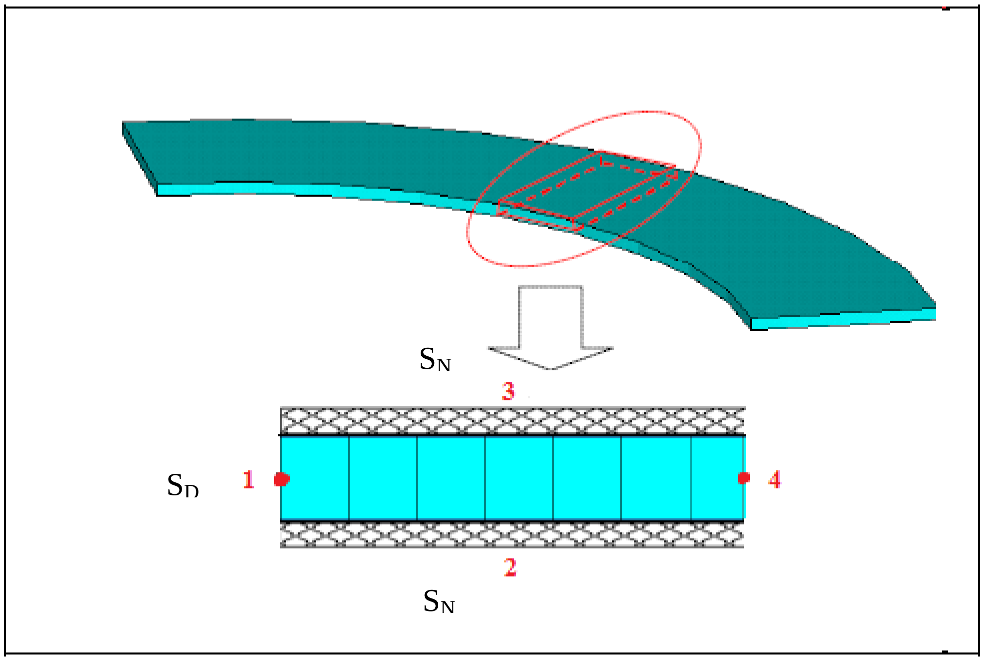


Fig.-4.6: Converting a heat flow problem through the pipe wall into 1 dimensional [43]

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 $\boldsymbol{T_b(x,t)}$ we see that $\boldsymbol{T_b(x,3)= 30}$. So this approach fully maintains the additivity or superposition. Decomposed profiles are also shown in the following graph 6.7.

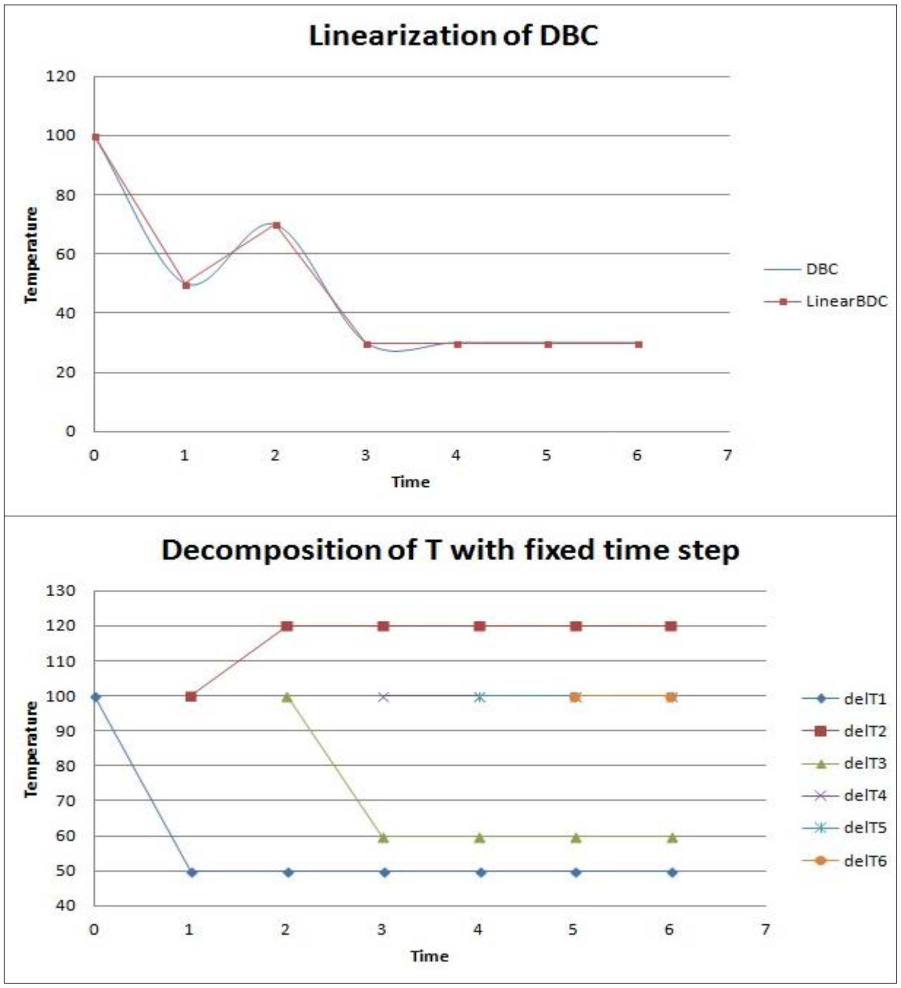
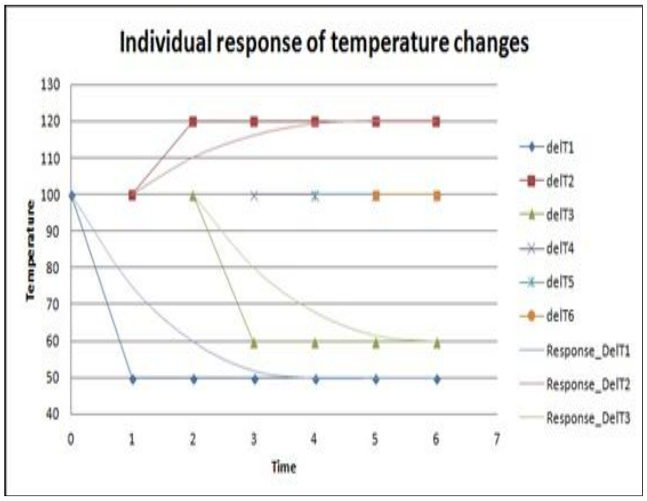


Fig. 4.7, 4.8: given temperature profile at the Dirichlet boundary (4.7) and its decomposed version (4.8)

Our next task is to calculate responses at the outer surface for each decomposed temperature by scaling with standard reference response $\boldsymbol{\Phi^r(r,t_s)}$ **.** 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.



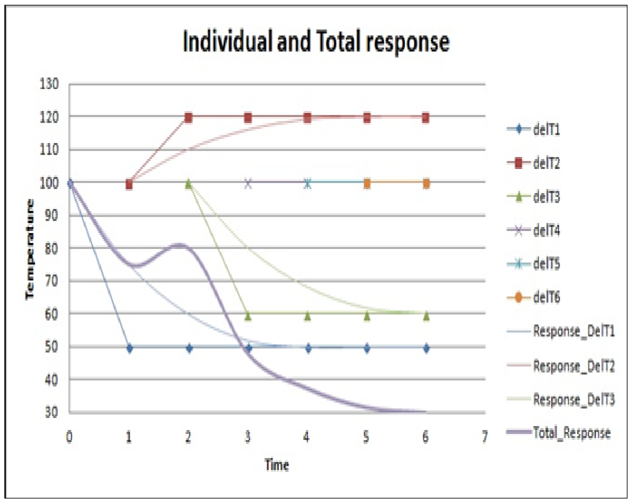


Fig. 4.9, 4.10: Typical responses of each decomposed temperature change (4.9) and cumulative response (4.10)

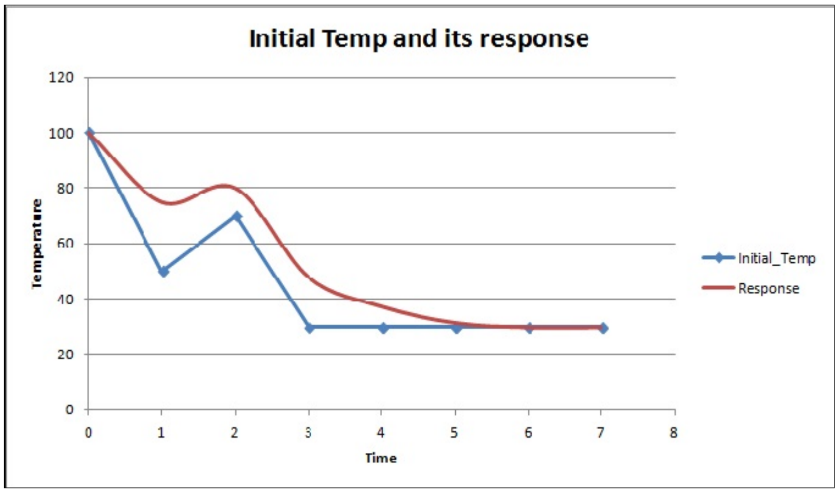


Fig. 4.11: Dirichlet temperature history at the inner wall and its response at the outer wall

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 $\boldsymbol{\Delta T}(\boldsymbol{\partial \Omega_i}, \boldsymbol{t_k})$ in the inaccessible boundary by knowing it's net response in the accessible part or accessible boundary at first local time step. By knowing $\boldsymbol{\Delta T}(\boldsymbol{\partial \Omega_i}, \boldsymbol{t_k})$ 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:

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

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

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:

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

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

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

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:

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

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

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

…………………………………………………………………………………………...

…………………………………………………………………………………………...

…………………………………………………………………………………………...

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

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

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

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

4.3.2 Computing unknown temperature changes at second time step

At second global time step $\boldsymbol{t_2}$, we can easily calculate the temperature difference at accessible points, $\Delta \boldsymbol{T}(\boldsymbol{r}, \boldsymbol{t_2})$ from the known temperature history. To compute required $\Delta \boldsymbol{T}_{2,i}$ ***or*** $\Delta \boldsymbol{T}_{2,i}(\boldsymbol{r}, \boldsymbol{t_2})$**,** 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 $\Delta \boldsymbol{T}^r(\boldsymbol{r}, [\Delta \boldsymbol{T}_{2,i}], \ \boldsymbol{t_2})$ caused by ***[***$\Delta \boldsymbol{T}_{2,i}$***]*. 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 ***[***$\Delta \boldsymbol{T}_{2,i}$***]*** + response caused by ***[***$\Delta \boldsymbol{T}_{1,i}$***]***

⇨ $\Delta \boldsymbol{T}(\boldsymbol{r}, \boldsymbol{t_2}) = \Delta \boldsymbol{T}^r(\boldsymbol{r}, [\Delta \boldsymbol{T}_{2,i}], \boldsymbol{t_2}) + \Delta \boldsymbol{T}^r(\boldsymbol{r}, [\Delta \boldsymbol{T}_{1,i}], \boldsymbol{t_2})$

⇨ $\Delta \boldsymbol{T}(\boldsymbol{r}, \boldsymbol{t_2}) = \Delta \boldsymbol{T}^r(\boldsymbol{r}, [\Delta \boldsymbol{T}_{2,i}], \boldsymbol{\lambda_1}) + \Delta \boldsymbol{T}^r(\boldsymbol{r}, [\Delta \boldsymbol{T}_{1,i}], \boldsymbol{\lambda_2})$

⇨ $\Delta \boldsymbol{T}^r(\boldsymbol{r}, [\Delta \boldsymbol{T}_{2,i}], \boldsymbol{t_2}) = \Delta \boldsymbol{T}(\boldsymbol{r}, \boldsymbol{t_2}) - \Delta \boldsymbol{T}^r(\boldsymbol{r}, [\Delta \boldsymbol{T}_{1,i}], \boldsymbol{\lambda_2})$ ***(4.12)***

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

$[\Delta T_{1,i}]$ is already computed; now we could also compute its response up to steady state time t_s. Computing the responses of $[\Delta T_{1,i}]$ 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 $\Delta T^r(r, [\Delta T_{2,i}], \lambda_2)$ due to temperature change in the first time step $[\Delta T_{1,i}]$ by applying forward algorithm. Once we are done with this computation, we could easily compute the net response $\Delta T^r(r, [\Delta T_{2,i}], \lambda_2)$ caused by$[(\Delta T_{2,i})]$. To get the unknown vector in the inaccessible part at second global time step e.g. $[(\Delta T_{2,i})]$, we will have to calculate the whole net response vector$[\Delta T^r(r, [\Delta T_{2,i}], \lambda_2)]$. Putting this vector $[\Delta T^r(r, [\Delta T_{2,i}], \lambda_2)]$ 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:

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

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

OR,

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

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

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

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 $[\Delta T_{2,i}]$.

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:

$$\begin{bmatrix} \Delta T^r(r_1,[\Delta T_{2,i}],t_2) \\ \Delta T^r(r_2,[\Delta T_{2,i}],t_2) \\ \Delta T^r(r_3,[\Delta T_{2,i}],t_2) \\ \vdots \\ \Delta T^r(r_P,[\Delta T_{2,i}],t_2) \end{bmatrix} = \begin{bmatrix} \Delta T(r_1,t_2) - \Delta T^r(r_1,[\Delta T_{1,i}],\lambda_2) \\ \Delta T(r_2,t_2) - \Delta T^r(r_2,[\Delta T_{1,i}],\lambda_2) \\ \Delta T(r_3,t_2) - \Delta T^r(r_3,[\Delta T_{1,i}],\lambda_2) \\ \vdots \\ \Delta T(r_P,t_2) - \Delta T^r(r_P,[\Delta T_{1,i}],\lambda_2) \end{bmatrix}$$

⇨ $$[\Delta T^r(r,[\Delta T_{2,i}],t_2)] = [\,\Delta T(r,t_2)] - [\,\Delta T^r([\Delta T_{1,i}],\lambda_2)] \qquad (4.14)$$

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

⇨ $[\,\Delta T^r(r,[\Delta T_{3,i}]),t_3] = [\,\Delta T(r,t_3)] - [\,\Delta T^r(r,[\Delta T_{1,i}],\lambda_3)] - [\,\Delta T^r([r,\Delta T_{2,i}],\lambda_2)]$

For forth time step it will be:

⇨ $[\,\Delta T^r(r,[\Delta T_{4,i}],t_4)] = [\,\Delta T(r,t_4)] - [\,\Delta T^r(r,[\Delta T_{1,i}],\lambda_4)] - [\,\Delta T^r(r,[\Delta T_{2,i}],\lambda_3)] - [\,\Delta T^r(r,[\Delta T_{3,i}],\lambda_2)]$

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

⇨ $[\,\Delta T^r(r,[\Delta T_{k,i}],\ t_k)] = [\,\Delta T(r,\ t_k)] - [\,\Delta T^r(r,[\Delta T_{1,i}],\lambda_k)] - [\,\Delta T^r(r,[\Delta T_{2,i}],\lambda_{k-1})] - \ldots\ldots - [\,\Delta T^r([\Delta T_{k-1,i}],\lambda_2)]$

⇨ $$[\,\Delta T^r(r,[\Delta T_{k,i}],\ t_k)] = [\,\Delta T(r,\lambda_k)] - \sum_{j=1}^{k-1}[\,\Delta T^r(r,[\Delta T_{j,i}],\lambda_{k-j+1})] \qquad \mathbf{(4.15)}$$

Thus inverse heat conduction algorithm for any time step 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 Δ***t***, 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 $A_{\o}^{n*n}$ picking the first row of R^{m*nn}. Let the elements of R^{m*nn} are $R_{i,j}$, then the constructed matrix $A_{\o}^{n*n}$ will look like:

$$A_{\o}^{n*n} = \begin{bmatrix} R_{1,1} & R_{1,2} & R_{1,3} & .. & .. & R_{1,n} \\ R_{1,n+1} & R_{1,n+2} & R_{1,n+2} & .. & .. & R_{1,2n} \\ \because & \because & \therefore & :: & :: & :: \\ .. & .. & .. & .. & .. & .. \\ R_{1,(n-1)n+1} & R_{1,(n-1)n+2} & R_{1,(n-1)n+3} & .. & .. & R_{1,nn} \end{bmatrix} \qquad (4.17)$$

Step-3: Construct the system of linear equation with the operator matrix $A_{\o}^{n*n}$, 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[$\Delta T^r(r, [\Delta T_{k+1,i}], t_{k+1})$] 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,

$$[T(r_{in}, t_k)] = [T(r_{in}, t_{k-1})] + [\Delta T_{k,i}]$$

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

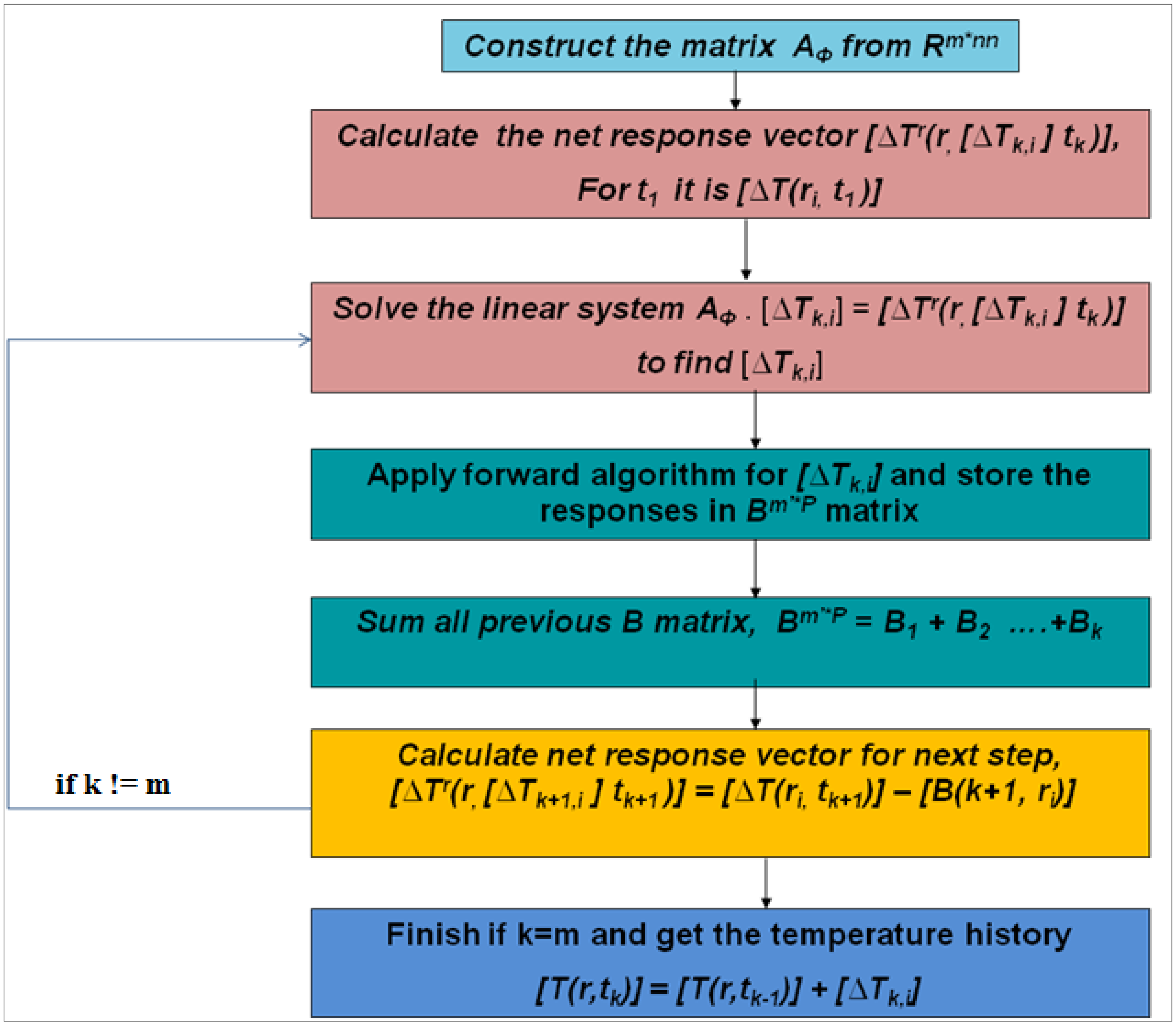

Construct the matrix A_Φ from R^{m*nn}
Calculate the net response vector $[\Delta T^r(r, [\Delta T_{k,i}] t_k)]$, For t_1 it is $[\Delta T(r_i, t_1)]$
Solve the linear system $A_\Phi . [\Delta T_{k,i}] = [\Delta T^r(r, [\Delta T_{k,i}] t_k)]$ to find $[\Delta T_{k,i}]$
Apply forward algorithm for $[\Delta T_{k,i}]$ and store the responses in B^{m*P} matrix
Sum all previous B matrix, $B^{m*P} = B_1 + B_2 ....+B_k$
Calculate net response vector for next step, $[\Delta T^r(r, [\Delta T_{k+1,i}] t_{k+1})] = [\Delta T(r_i, t_{k+1})] - [B(k+1, r_i)]$
if k != m
Finish if k=m and get the temperature history $[T(r,t_k)] = [T(r,t_{k-1})] + [\Delta T_{k,i}]$

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 *$[\Delta T(r_i, t_1)]$* 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: $\Delta T_{in}(t_1) = \frac{\Delta Tout\ (t_1) * \Delta Tin\_ref\ (\lambda_1)}{\Delta To\_ref\ (\lambda 1)} = \frac{0.8*100}{2} = 40$

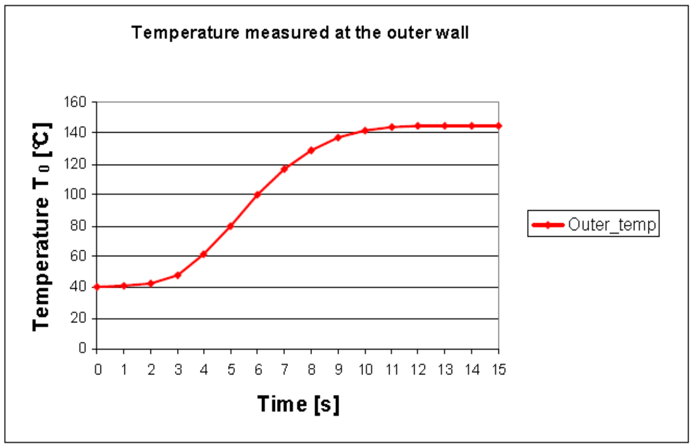


Fig- 4.13: Given outside temperature history [42]

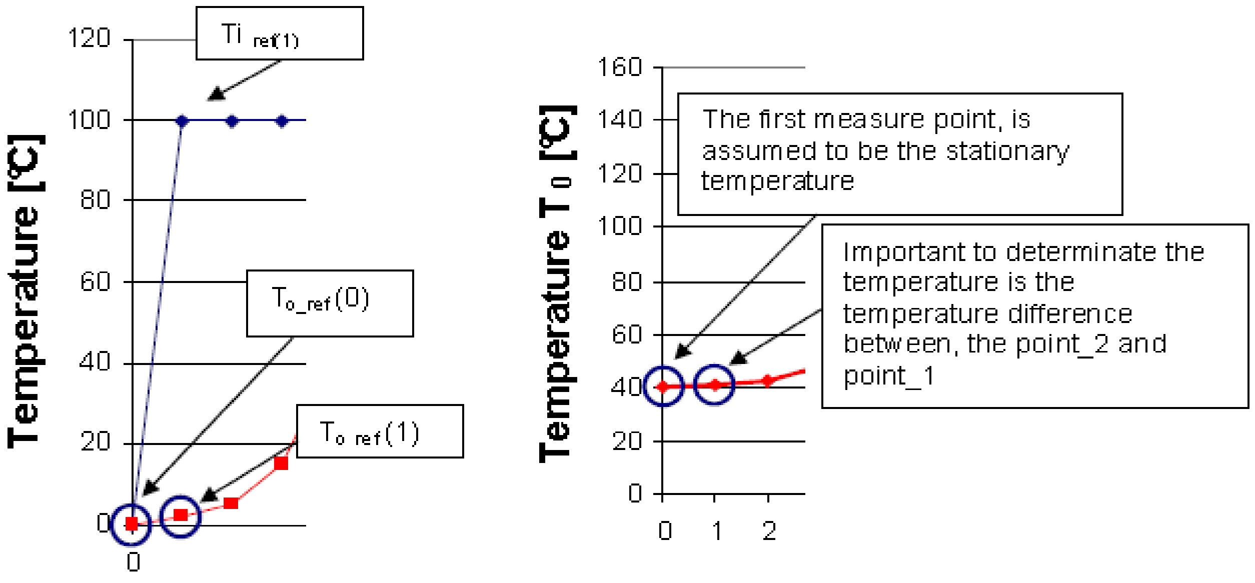


Fig. 4.14, 4.15: Showing ΔT^{ref} and its response, 4.15 is in magnified view [42]

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 $\Delta T_{in}(t_3)$.

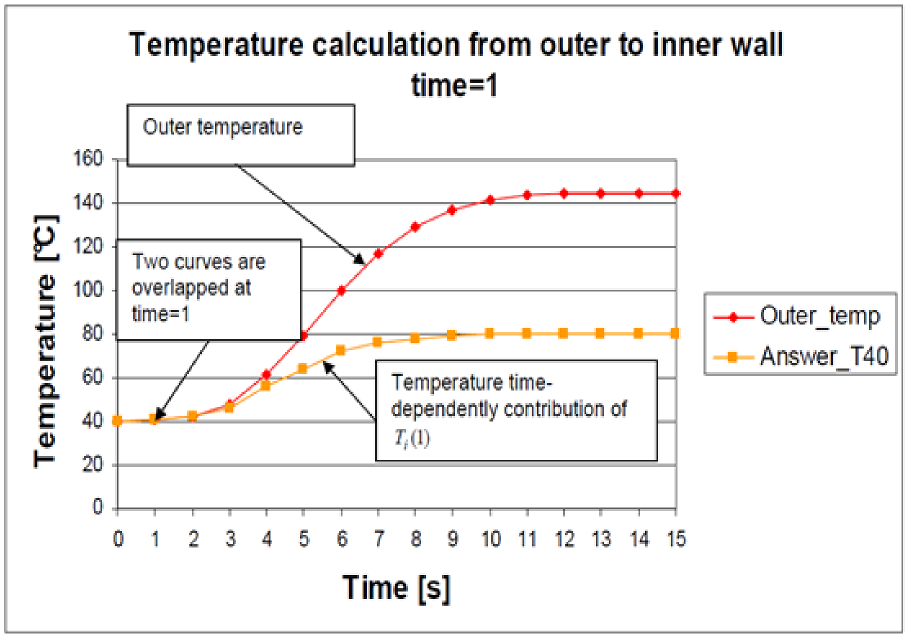


Figure 4.16: Outer wall temperature history and the contribution of $\Delta T_{in}(t_1)$ [42]

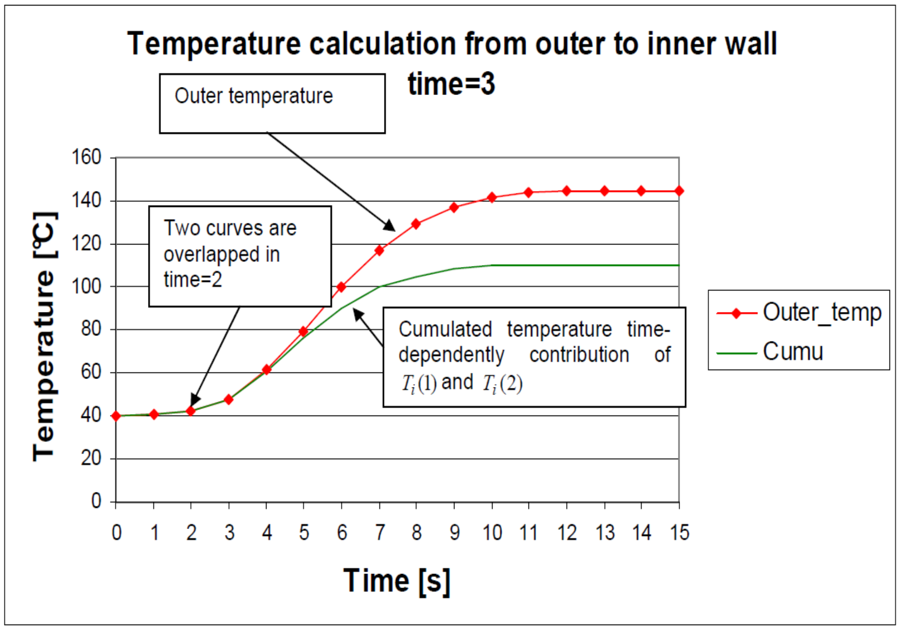


Figure 4.17: Cumulative contribution of $\Delta T_{in}(t_1)$ *and* $\Delta T_{in}(t_2)$ to the outer wall temperature history [42]

In this way we can proceed to compute the complete inner history step by step. The combined contribution of 40° and 30° 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 $\Delta \boldsymbol{q}(\boldsymbol{\partial \Omega_i}, \boldsymbol{t_k})$ 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 $\Delta \boldsymbol{T}^{ref}$ and its responses we will have to select some suitable $\Delta \boldsymbol{q}^{ref}$ and its responses $\Delta\Phi_i^r(\boldsymbol{r}, \boldsymbol{\lambda_j})$ **.** 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:

$$\frac{1}{\Delta T_{ref}}
\begin{bmatrix}
\Delta\Phi_1^r(\boldsymbol{r_1},\boldsymbol{\lambda_1}) & \Delta\Phi_2^r(\boldsymbol{r_1},\boldsymbol{\lambda_1}) & \dots & \dots & \dots & \Delta\Phi_P^r(\boldsymbol{r_1},\boldsymbol{\lambda_1}) \\
\Delta\Phi_1^r(\boldsymbol{r_2},\boldsymbol{\lambda_1}) & \Delta\Phi_2^r(\boldsymbol{r_2},\boldsymbol{\lambda_1}) & \dots & \dots & \dots & \Delta\Phi_P^r(\boldsymbol{r_2},\boldsymbol{\lambda_1}) \\
\Delta\Phi_1^r(\boldsymbol{r_3},\boldsymbol{\lambda_1}) & \Delta\Phi_2^r(\boldsymbol{r_3},\boldsymbol{\lambda_1}) & \dots & \dots & \dots & \Delta\Phi_P^r(\boldsymbol{r_3},\boldsymbol{\lambda_1}) \\
\dots & \dots & \dots & \dots & \dots & \dots \\
\dots & \dots & \dots & \dots & \dots & \dots \\
\Delta\Phi_1^r(\boldsymbol{r_P},\boldsymbol{\lambda_1}) & \Delta\Phi_2^r(\boldsymbol{r_P},\boldsymbol{\lambda_1}) & \dots & \dots & \dots & \Delta\Phi_P^r(\boldsymbol{r_P},\boldsymbol{\lambda_1})
\end{bmatrix}
\begin{bmatrix}
\Delta\boldsymbol{q}_{k,1} \\ \Delta\boldsymbol{q}_{k,2} \\ \Delta\boldsymbol{q}_{k,3} \\ \dots \\ \dots \\ \Delta\boldsymbol{q}_{k,P}
\end{bmatrix}
=
\begin{bmatrix}
\Delta\boldsymbol{T}^r\left(\boldsymbol{r_1},[\Delta\boldsymbol{q}_{k,i}],\ \boldsymbol{t_k}\right) \\
\Delta\boldsymbol{T}^r\left(\boldsymbol{r_2},[\Delta\boldsymbol{q}_{k,i}],\ \boldsymbol{t_k}\right) \\
\Delta\boldsymbol{T}^r\left(\boldsymbol{r_3},[\Delta\boldsymbol{q}_{k,i}],\ \boldsymbol{t_k}\right) \\
\vdots \\
\Delta\boldsymbol{T}^r\left(\boldsymbol{r_P},[\Delta\boldsymbol{q}_{k,i}],\ \boldsymbol{t_k}\right)
\end{bmatrix}
\tag{4.19}$$

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.