Implementation of the C- and F- Processes heat conduction model and particle based formulation via the GS4 framework: Moving external source problems

A Thesis
SUBMITTED TO THE FACULTY OF
UNIVERSITY OF MINNESOTA
BY

Brandon Henry

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

Prof. Kumar K. Tamma
(Faculty Adviser)

February 2018

Acknowledgements

Firstly, I would like to thank Professor Kumar K. Tamma for affording me the opportunity of attending graduate school and joining his lab.

Furthermore, I would also like to thank my lab mates for their continued support and acting as a sound board during my research into this topic, specifically, Tao Xue as well as David Tae, Rohit Deokar, Dean Maxam, and Masao Shimada.

Dedication

This thesis is dedicated to my parents Jeff and Marian Henry, my siblings Jeffrey and Ashley, and my girlfriend Christina. Without their support I wouldn't have made it to where I am today. I simply can't thank you all enough for all your love and support over the years.

Abstract

A wide variety of engineering problems involve thermal considerations and more specifically various heat conduction processes. Many of which involve heat sources that either are stationary or moving; such as laser welding, laser shot peening, etc. Such problems are difficult or expensive to analyze either analytically or experimentally leaving numerical methods as a viable alternative. The research presented herein will focus on the implementation of the C- and F- processes heat conduction model in a particle based framework utilizing the GS4 *i*Integration time stepping schemes with a focus on applying this to moving heat source type problems. Numerous illustrative numerical examples verifying and validating the present methodology are highlighted, and the pros/cons of the proposed models are discussed in comparison to past practices and traditional approaches.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

Processes such as arc welding, laser/plasma cutting, and 3D printing are similar in that they each require the utilization of a moving heat source to accurately capture the underlying physics. The ability to model such processes is required to enable users to better understand changes in material properties, required heat injection, and various other parameters to best utilize and implement these processes.

Moving heat sources were first studied, to this author's knowledge, by Rosenthal in order to have a better analytical understanding of arc welding and cutting [1], as well as a better understanding of metal treatments [2]. Since his initial investigations, research into numerical methods for various problems exhibiting moving sources has been conducted. Predominately the finite element method (FEM) was utilized as in Goldak et al. in [3] and Ascough in [4], as well as by many other authors.

The main drawbacks of utilizing a mesh based model, such as FEM, for modeling a heat source is that along the source's path a very fine mesh is required. Furthermore, if any phase changes or plastic deformation manifests itself during the simulation the mesh must be adapted to fit the new geometry/phases, and any stored solutions must be transferred therein to the new mesh, leading to computational complexities. Since the development of the so called "meshless" or "particle methods", some of the complexities, such as but not limited to local mesh refinement and moving meshes, are more suitable and easily adapted. Meshless or particle methods differ from meshed methods in that they utilize nodal type frameworks to represent geometries, in contrast to element type frameworks utilized in meshed methods. This difference in framework allows for such methods to utilize less complex discretization and allow for more flexibility than the corresponding meshed methods.

Well known meshless type methods include the smoothed particle hydrodynamic (SPH) model developed by Monaghan [5] and the moving particle semi-implicit (MPS) method developed by Koshizuka [6]; which are used predominately for fluid flow and heat transfer; as well as the discrete element method (DEM) developed by Cundall et al. [7] used predominately for granular structures.

Other meshless methods such as the meshless element free Galerkin (MEFG) method, utilized by Pham in [8], and the finite pointset method (FPM), utilized by Resendiz-Flores and Saucedo-Zendejo in [9], have been applied directly to the moving heat source

problems. Both methods produce accurate results but in their current form are resigned to modeling only parabolic, Fourier-type, heat conduction because of the constitutive models that have been implemented in these formulations.

Furthermore, the previously mentioned FPM implementation utilizes the implicit Euler time stepping algorithm, which has various disadvantages; such as numerical instability, error, as well as only possessing first order convergence in time. Other time integration schemes used for first-order time-dependent systems; such as Crank-Nicolson [10] or more broadly the generalized trapezoidal family of algorithms [11], could also be utilized for solving these types of problems with each individual method having its own drawbacks. The most universal issue with these types of methods is that they are resigned to only first-order time-dependent systems. Consequently, for multi-physical/ disciplinary type problems which usually involve a combination of first-order and second-order time dependent systems, such as fluid-structure interaction, one may need to couple first-order schemes with second-order time integration schemes for a fully coupled system solution.

To address the aforementioned issues and advance/enhance the previous studies, we propose to implement the generalized heat diffusion model, termed C- and F- processes heat conduction model, to the moving heat source problems. The particle-based method, MPS, and the generalized single step single field (GS4) time integration are exploited to carry out the spatial and temporal discretization, respectively. By using the C- and F- heat

conduction model developed by Tamma et al. [12], we can readily model both parabolic and hyperbolic heat conduction problems by varying the dimensionless heat parameter, F_T. Furthermore, by implementing the isochronous integration [*i*Integration] framework for time stepping, we have the flexibility to utilize not only past efforts that are second order time accurate but also new advances; as they are recoverable within *i*Integration, while also allowing for the extension to second order type multi-physics problems if need be. Lastly, by employing meshless type methods, specifically the MPS method, we can utilize uniform discretization instead of having to refine a mesh around points of interest. All of this allows for a simple, yet versatile, framework that can model many heat conduction problem as well as moving source type problems.

The remainder of this thesis will be structured as follows: Chapter 2 gives a brief overview of the original MPS method developed for fluid flows. Chapter 3 covers numerical time integration, specifically the isochronous integration [*i*Integration] framework. Chapter 4 explains the C- and F- processes heat conduction model which entails both parabolic and hyperbolic heat conduction, as well as, anything in-between. Chapter 5 is the first chapter of practical implementation where the C- and F- model within an MPS framework shows various illustrative examples, without any type of source terms. Chapter 6 builds upon Chapter 5 but introduces stationary source terms with various examples. Chapter 7 finally introduces the moving source term with various illustrative examples. Lastly in Chapter 8 a conclusion will be presented.

Chapter 2: Moving Particle Semi-Implicit (MPS) Method

Background

The moving particle semi-implicit (MPS) method is a meshless method originally developed by Koshizuka and Oka to simulate incompressible fluid flows [1]. As a meshless method, the MPS method utilizes a Lagrangian approach in contrast to mesh based methods; such as the widely used finite element method, that utilizes the Eulerian approach for numerical formulations. Since its development it has been extended to weakly compressible flows [2] and structural problems [3].

Governing Equations

The governing equations for isothermal, incompressible fluid flow are the continuity and the Navier-Stokes equations, respectively,

$$\frac{\partial \rho}{\partial t} = 0 \tag{1}$$

$$\frac{D\boldsymbol{u}}{Dt} = -\frac{1}{\rho}\nabla P + \boldsymbol{f} \tag{2}$$

where ρ is the fluid density, $\boldsymbol{u}$ is the fluid velocity vector, ∇P is the pressure gradient, and $\boldsymbol{f}$ is any applied external force.

Particle Interaction Model

In the MPS method, like the more widely known smoothed particle hydrodynamics (SPH) method [4], particle interactions are determined by the distance between particles and a prescribed weight function, also referred to as a kernel function. The originally specified weight function is as follows,

$$w(r) = \begin{cases} \dfrac{r_e}{r} - 1 & (0 \le r < r_e) \\ 0 & (r_e < r) \end{cases} \tag{3}$$

Where r is the distance between two particles and r_e is cutoff distance of influence, meaning that any two particles with a distance apart less than r_e exert influence upon each other. An illustration of a group of particles highlighting the radius of influence is shown below,

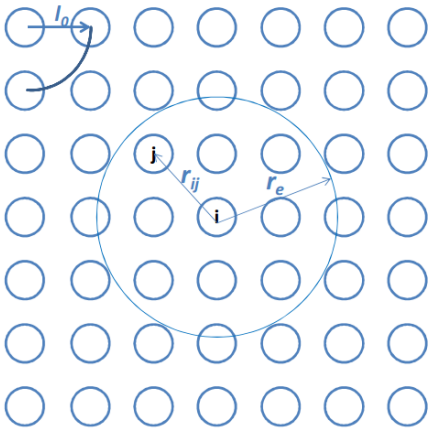


Figure 1. Illustration of particle discretization and radius of influence for MPS method

However, work has also been conducted to characterize different weight functions based upon their benefits and drawbacks [5]. In 2D, the weight function is a circular region determined by a radius of influence, also known as the cutoff radius, whereas in 3D, the weight function is spherical in nature. Furthermore, it should be noted that there is no explicit collision model for particle interactions. From the weight function one can determine the particle number density for each particle which is defined by the following relation,

$$\langle n \rangle_i = \sum_{j \neq i} w(|\boldsymbol{r}_j - \boldsymbol{r}_i|) \tag{4}$$

If each particle has the same mass, then the fluid density is proportional to the particle number density, meaning that the continuity is satisfied if the particle number density is constant, usually denoted by n^0.

Gradient, Laplacian, Divergence, and Rotation Operators

Gradient Operator

The gradient operator for the MPS method as originally develop by Koshizuka and Oka [6], for a given physical quantity ϕ between two particles i and j at positions $\boldsymbol{r}_i$ and $\boldsymbol{r}_j$ is given by

$$\nabla \phi_{ij} = \frac{(\phi_j - \phi_i)(\boldsymbol{r}_j - \boldsymbol{r}_i)}{|\boldsymbol{r}_j - \boldsymbol{r}_i|^2} \tag{5}$$

Therefore, the gradient vector for particle i with respect to all other particles is a weighted average using the kernel function above, specifically,

$$\langle \nabla \phi \rangle_i = \frac{d}{n^0} \sum_{j \neq i} \frac{\phi_j - \phi_i}{|\boldsymbol{r}_j - \boldsymbol{r}_i|^2} (\boldsymbol{r}_j - \boldsymbol{r}_i) w(|\boldsymbol{r}_j - \boldsymbol{r}_i|) \tag{6}$$

where d is the number of spatial dimensions. An alternate form of the gradient operator was also proposed in [1], which satisfies the following relation,

$$0 = \sum_{j \neq i} \frac{(\boldsymbol{r}_j - \boldsymbol{r}_i)}{|\boldsymbol{r}_j - \boldsymbol{r}_i|^2} w(|\boldsymbol{r}_j - \boldsymbol{r}_i|) \tag{7}$$

From which the original operator can be rearranged as follows,

$$\langle \nabla \phi \rangle_i = \frac{d}{n^0} \sum_{j \neq i} \frac{\phi_j - \phi_i'}{|\boldsymbol{r}_j - \boldsymbol{r}_i|^2} (\boldsymbol{r}_j - \boldsymbol{r}_i) w(|\boldsymbol{r}_j - \boldsymbol{r}_i|) \tag{8}$$

Where ϕ_i' can be any constant value, however the value is usually specified as,

$$\phi_i' = min(\phi_j) \tag{9}$$

for any j satisfying

$$w(|\boldsymbol{r}_{ij}|) \neq 0$$

Laplacian Operator

The Laplacian for the MPS method originally developed by Koshizuka and Oka [6] , for a given quantity ϕ in linear diffusion is defined as,

$$\langle \nabla^2 \phi \rangle = \frac{2d}{n^0 \lambda} \sum_{j \neq i} (\phi_j - \phi_i)\, w(|\boldsymbol{r}_j - \boldsymbol{r}_i|) \tag{10}$$

where

$$\lambda = \frac{\int_V w(r) r^2 dv}{\int_V w(r)\, dv} \approx \frac{\sum_{j \neq i} w(|\boldsymbol{r}_j - \boldsymbol{r}_i|) |\boldsymbol{r}_j - \boldsymbol{r}_i|^2 v_j}{\sum_{j \neq i} w(|\boldsymbol{r}_j - \boldsymbol{r}_i|)\, v_j} \tag{11}$$

It is claimed [6], that this Laplacian formulation is a conservative model since any quantity ϕ lost by particle i is simply gained by particle j.

Divergence Operator

There was not a divergence operator originally developed for the MPS method by Koshizuka and Oka, however one was developed by Yoon in [13]. Similar to how the gradient operator was determined, the divergence of two vectors $\boldsymbol{\phi_i}$ and $\boldsymbol{\phi_j}$, is defined as,

$$\nabla \cdot \boldsymbol{\phi_{ij}} = \left(\boldsymbol{\phi_j} - \boldsymbol{\phi_i}\right) \cdot \frac{\boldsymbol{r}_j - \boldsymbol{r}_i}{\left|\boldsymbol{r}_j - \boldsymbol{r_i}\right|^2} \tag{12}$$

Therefore, the divergence for particle i with respect to all other particles is a weighted average using the kernel function,

$$\langle \nabla \cdot \boldsymbol{\phi} \rangle_i = \frac{d}{n^0} \sum_{j \neq i} \frac{\left(\boldsymbol{\phi}_j - \boldsymbol{\phi}_i\right) \cdot \left(\boldsymbol{r}_j - \boldsymbol{r}_i\right)}{\left|\boldsymbol{r}_j - \boldsymbol{r_i}\right|^2} \, w\left(\left|\boldsymbol{r}_j - \boldsymbol{r}_i\right|\right) \tag{13}$$

Rotation Operator

When the MPS method was extended to solid mechanics, rotation had to be addressed, because an operator was not originally described by Koshizuka and Oka. However, there exists one that has been presented by Robortella, Kazuo, and Cheng [14]. It is as follows for any two particles the rotation experienced is defined as,

$$\nabla \times \boldsymbol{\phi_{ij}} = \left(\boldsymbol{\phi_j} - \boldsymbol{\phi_i}\right) \cdot \frac{\boldsymbol{s_{ij}}}{\left|\boldsymbol{r}_j - \boldsymbol{r}_i\right|} \tag{14}$$

Therefore, the rotation for particle i with respect to all other particles is a weighted average using the kernel function, specifically,

$$\nabla \times \boldsymbol{\phi} = \frac{d}{n^0} \sum_{j \neq i} \frac{\left(\boldsymbol{\phi}_j - \boldsymbol{\phi}_i\right) \cdot \boldsymbol{s_{ij}}}{\left|\boldsymbol{r}_j - \boldsymbol{r}_i\right|} \, w\left(\left|\boldsymbol{r}_j - \boldsymbol{r}_i\right|\right) \tag{15}$$

Chapter 3: Time Integration (GS4 isochronous integration [*i*Integration])

The isochronous framework [*i*Integration] is a novel time integration framework developed by Tamma et al. [15], based upon the previous generalized single step single solve (GSSSS or GS4) time integration schemes. This framework consists of a unified time integration framework applicable for both first order and second order systems. It allows greater flexibility than other time stepping methods that are resigned mostly to either first order or second order transient systems. Typical examples are the Crank-Nicolson method for first order systems, the Newmark method for second order systems, and numerous other methods targeted solely to either first order systems or second order systems.

This time stepping algorithm was chosen for the model because of its versatility; and allows for easy extension to multiphysics, however due to the two-field form of the model; which will be introduced and discusses later, the model itself is only first-order in time. In the sections to follow, both the first-order (GS4-1) and second-order (GS4-2) time integration frameworks and schemes will be presented, as well as the unified framework that allows for time integration of both first order and second order systems.

GS4-1 Framework for First Order Transient Systems

The implementation of GS4 for first order systems; hence forth called GS4-1, is as follows. Consider a first order linear transient system of form with given initial condition

$$\boldsymbol{M}\dot{\boldsymbol{u}}(\boldsymbol{t}) + \boldsymbol{K}\boldsymbol{u}(\boldsymbol{t}) = \boldsymbol{F}(\boldsymbol{t}) \tag{16}$$

$$\boldsymbol{u}(0) = \boldsymbol{u_0} \tag{17}$$

where $\boldsymbol{M}$ is the mass matrix, $\boldsymbol{K}$ is the stiffness matrix, $\boldsymbol{u}$ is the field vector, and $\boldsymbol{F}$ is the externally applied force. This system can be written in algorithmic form as follows,

$$\boldsymbol{M}\tilde{\dot{\boldsymbol{u}}} + \boldsymbol{K}\tilde{\boldsymbol{u}} = \tilde{\boldsymbol{F}} \tag{18}$$

where the algorithmic values which are indicated by ($\tilde{}$), these values are defined as,

$$\tilde{\dot{\boldsymbol{u}}} = \dot{\boldsymbol{u}}_n + \Lambda_6 W_1 * \Delta\dot{\boldsymbol{u}} \tag{19}$$

$$\tilde{\boldsymbol{u}} = \boldsymbol{u_n} + \Lambda_4 W_1 \Delta t\, \dot{\boldsymbol{u}}_{\boldsymbol{n}} + \Lambda_5 W_2 \Delta t \Delta\dot{\boldsymbol{u}} \tag{20}$$

$$\tilde{\boldsymbol{F}} = \boldsymbol{F_n} + W_1(\boldsymbol{F_{n+1}} - \boldsymbol{F_n}) \tag{21}$$

GS4-1 applied to a given system

To apply the explicit form of GS4-1 to a first order, transient system with a known $\boldsymbol{u_n}$ and $\dot{\boldsymbol{u}}_{\boldsymbol{n}}$, one can find the values at the next time step $\boldsymbol{u_{n+1}}$ and $\dot{\boldsymbol{u}}_{\boldsymbol{n+1}}$ by solving the above system for $\Delta\dot{\boldsymbol{u}}$,

$$\begin{aligned}(\Lambda_6 W_1 \boldsymbol{M} + \Lambda_5 W_2 \Delta t \boldsymbol{K})\Delta\dot{\boldsymbol{u}} \\ = -\boldsymbol{M}\dot{\boldsymbol{u}}_{\boldsymbol{n}} - \boldsymbol{K}(\boldsymbol{u_n} + \Lambda_4 W_1 \Delta t \dot{\boldsymbol{u}}_{\boldsymbol{n}}) + \boldsymbol{F_n} + W_1(\boldsymbol{F_{n+1}} - \boldsymbol{F_n})\end{aligned} \tag{22}$$

Once $\Delta\dot{\boldsymbol{u}}$ has been found then the following updates can be made,

$$\boldsymbol{u_{n+1}} = \boldsymbol{u_n} + \lambda_4 \dot{\boldsymbol{u}}_{\boldsymbol{n}} \Delta t + \lambda_5 \Delta\dot{\boldsymbol{u}} \Delta t \tag{23}$$

$$\dot{\boldsymbol{u}}_{\boldsymbol{n+1}} = \dot{\boldsymbol{u}}_{\boldsymbol{n}} + \Delta\dot{\boldsymbol{u}} \tag{24}$$

Where,

$$\begin{aligned}\Lambda_6 W_1 = \frac{3 + \rho_\infty + \rho_\infty^s - \rho_s \rho_\infty^s}{2(1+\rho_\infty)}, \qquad W_1 = \frac{1}{1+\rho_\infty} \\ \Lambda_5 W_2 = \frac{1}{(1+\rho_\infty)(1+\rho_\infty^s)}, \qquad \lambda_5 = \frac{1}{1+\rho_\infty^s} \\ \Lambda_4 W_1 = \frac{1}{1+\rho_\infty}, \qquad \lambda_4 = 1\end{aligned} \tag{25}$$

The parameters ρ_s and ρ_∞^s are variables selected by the user and are referred to as the principal and spurious roots, respectively. These roots must satisfy the following relation,

$$0 \leq \rho_\infty^s \leq \rho_\infty \leq 1 \tag{26}$$

All of the algorithms encompassed by GS4-1 are second order time accurate in space, zero-order overshoot, unconditionally stable, and have controllable numerical dissipation with the option of selective control. The algorithms contained within GS4-1 can be seen in the table below,

Table 1. Algorithms contained in GS4-1 framework [15]

Parameters	Algorithms
$\rho_\infty = \rho_\infty^S = 1$	Crank-Nicolson method
$\rho_\infty = \rho_\infty^S = 0$	Gear's method
$\rho_\infty = \rho_\infty^S$	Algorithms without selective control feature
$\rho_\infty \neq \rho_\infty^S$	Algorithms with selective control feature

GS4-2 Framework for Second Order Transient Systems

Consider a linear, second order transient system with the following governing equation and initial conditions,

$$\boldsymbol{M}\ddot{\boldsymbol{u}}(\boldsymbol{t}) + \boldsymbol{C}\dot{\boldsymbol{u}}(\boldsymbol{t}) + \boldsymbol{K}\boldsymbol{u}(\boldsymbol{t}) = \boldsymbol{F}(\boldsymbol{t}) \tag{27}$$

$$\begin{aligned} \boldsymbol{u}(0) &= \boldsymbol{u_0} \\ \dot{\boldsymbol{u}}(0) &= \dot{\boldsymbol{u}}_{\boldsymbol{0}} \end{aligned} \tag{28}$$

where $\boldsymbol{M}$ is the mass matrix, $\boldsymbol{C}$ is the damping matrix, $\boldsymbol{K}$ is the stiffness matrix, $\boldsymbol{u}$ is the displacement vector, and $\boldsymbol{F}$ is the externally applied force. Similar to in GS4-1, this system can be written in algorithmic form as follows,

$$\boldsymbol{M}\tilde{\ddot{\boldsymbol{u}}} + \boldsymbol{C}\tilde{\dot{\boldsymbol{u}}} \quad \boldsymbol{K}\tilde{\boldsymbol{u}} = \tilde{\boldsymbol{F}} \tag{29}$$

where the algorithmic values, indicated by ($\tilde{}$), which are given as follows,

$$\tilde{\ddot{\boldsymbol{u}}} = \ddot{\boldsymbol{u}}_{\boldsymbol{n}} + \Lambda_6 W_1 \Delta\ddot{\boldsymbol{u}} \tag{30}$$

$$\tilde{\dot{\boldsymbol{u}}} = \dot{\boldsymbol{u}}_{\boldsymbol{n}} + \Lambda_4 W_1 \Delta t \ddot{\boldsymbol{u}}_{\boldsymbol{n}} + \Lambda_5 W_2 \Delta t \, \Delta\ddot{\boldsymbol{u}} \tag{31}$$

$$\tilde{\boldsymbol{u}} = \boldsymbol{u}_{\boldsymbol{n}} + \Lambda_1 W_1 \Delta t \dot{\boldsymbol{u}}_{\boldsymbol{n}} + \Lambda_2 W_2 \Delta t^2 \ddot{\boldsymbol{u}}_{\boldsymbol{n}} + \Lambda_3 W_3 \Delta t^2 \Delta\ddot{\boldsymbol{u}} \tag{32}$$

$$\tilde{\boldsymbol{F}} = \boldsymbol{F}_{\boldsymbol{n}} + W_1(\boldsymbol{F}_{\boldsymbol{n+1}} - \boldsymbol{F}_{\boldsymbol{n}}) \tag{33}$$

Furthermore, the GS4-2 framework is comprised of two separate sub-frameworks categorized by their respective overshoot behavior; specifically, the U0 family of algorithms that is characterized by zero-order displacement overshoot or the V0 family of algorithms that is characterized by zero-order velocity overshoot. These two families of algorithms and their respective implementations, for the following system, can be seen below. With given $\boldsymbol{u_n}$, $\dot{\boldsymbol{u}}_{\boldsymbol{n}}$, and $\ddot{\boldsymbol{u}}_{\boldsymbol{n}}$, and utilizing equations (30) - (33), the values at the next time step, $\boldsymbol{u_{n+1}}$, $\dot{\boldsymbol{u}}_{\boldsymbol{n+1}}$, and $\ddot{\boldsymbol{u}}_{\boldsymbol{n+1}}$, can be determined by first solving for $\Delta\ddot{\boldsymbol{u}}$, as shown below,

$$(\Lambda_6 W_1 \boldsymbol{M} + \Lambda_5 W_2 \Delta t \boldsymbol{C} + \Lambda_3 W_3 \Delta t^2 \boldsymbol{K})\Delta\ddot{\boldsymbol{u}} = -\boldsymbol{M}\ddot{\boldsymbol{u}}_{\boldsymbol{n}} - \boldsymbol{C}(\dot{\boldsymbol{u}}_{\boldsymbol{n}} + \Lambda_4 W_1 \Delta t \ddot{\boldsymbol{u}}_{\boldsymbol{n}}) - \boldsymbol{K}(\boldsymbol{u_n} + \Lambda_1 W_1 \Delta t \dot{\boldsymbol{u}}_{\boldsymbol{n}} + \Lambda_2 W_2 \Delta t^2 \ddot{\boldsymbol{u}}_{\boldsymbol{n}}) + \boldsymbol{F_n} + W_1(\boldsymbol{F_{n+1}} - \boldsymbol{F_n}) \tag{34}$$

Which can then be used to update each variable at the end of every time step as follows,

$$\boldsymbol{u_{n+1}} = \boldsymbol{u_n} + \lambda_1 \dot{\boldsymbol{u}}_{\boldsymbol{n}} \Delta t + \lambda_2 \ddot{\boldsymbol{u}}_{\boldsymbol{n}} \Delta t^2 + \lambda_3 \Delta\ddot{\boldsymbol{u}} \Delta t^2 \tag{35}$$

$$\dot{\boldsymbol{u}}_{\boldsymbol{n+1}} = \dot{\boldsymbol{u}}_{\boldsymbol{n}} + \lambda_4 \ddot{\boldsymbol{u}}_{\boldsymbol{n}} \Delta t + \lambda_5 \Delta\ddot{\boldsymbol{u}} \Delta t^2 \tag{36}$$

$$\ddot{\boldsymbol{u}}_{\boldsymbol{n+1}} = \ddot{\boldsymbol{u}}_{\boldsymbol{n}} + \Delta\ddot{\boldsymbol{u}} \tag{37}$$

Where the two families of algorithms differ in coefficients associated with the matrices.

U0 Family of Algorithms

The U0 family of algorithms is characterized by zero-order overshoot for displacement, and no more than first-order overshoot for velocity. Its coefficients are defined as follows,

$$
\begin{aligned}
\Lambda_1 W_1 &= \frac{1}{1+\rho_\infty^s}, \qquad \lambda_1 = 1 \\
\Lambda_2 W_2 &= \frac{1}{2(1+\ \rho_\infty^s)}, \qquad \lambda_2 = \frac{1}{2} \\
\Lambda_3 W_3 &= \frac{1}{(1+\rho_\infty^{min})(1+\rho_\infty^{max})(1+\rho_\infty^s)}, \\
&\qquad \lambda_3 = \frac{1}{(1+\rho_\infty^{min})(1+\rho_\infty^{max})} \\
\Lambda_4 W_1 &= \frac{1}{1+\rho_\infty^s}, \qquad \lambda_4 = 1 \\
\Lambda_5 W_2 &= \frac{3+\rho_\infty^{min}+\rho_\infty^{max}-\rho_\infty^{min}\rho_\infty^{max}}{2(1+\rho_\infty^{min})(1+\rho_\infty^{max})(1+\rho_\infty^s)}, \\
&\qquad \lambda_5 = \frac{3+\rho_\infty^{min}+\rho_\infty^{max}-\rho_\infty^{min}\rho_\infty^{max}}{2(1+\rho_\infty^{min})(1+\rho_\infty^{max})} \\
\Lambda_6 W_1 &= \frac{2+\rho_\infty^{min}+\rho_\infty^{max}+\rho_\infty^s-\rho_\infty^{min}\rho_\infty^{max}\rho_\infty^s}{(1+\rho_\infty^{min})(1+\rho_\infty^{max})(1+\rho_\infty^s)} \\
W_1 &= \frac{1}{1+\rho_\infty^s}
\end{aligned}
\tag{38}
$$

The parameters ρ_∞^{min}, ρ_∞^{max}, and ρ_∞^s are variables selected by the user and are referred to as the principal roots and spurious root, respectively. They must satisfy the following relation,

$$
0 \leq\ \rho_\infty^s \leq \rho_\infty^{min} \leq\ \rho_\infty^{max} \leq 1 \tag{39}
$$

The algorithms contained in the U0 family of the GS4-2 framework can be seen in the table below,

Table 2. Algorithms contained in U0 family of the GS4-2 framework [15]

Parameters	Algorithms
$\rho_\infty^{min} = \rho_\infty^{max} = \rho_\infty^s = 1$	Mid-point rule
$\rho_\infty^{min} = \rho_\infty^{max} = 1, \quad \rho_\infty^s = 0$	Newmark method
$\rho_\infty^{max} = 1, \quad \rho_\infty^{min} = \rho_\infty^s$	Optimal algorithms with zero-order displacement and zero-order velocity overshoot behaviors and yield only minimal numerical dissipation and dispersion for damped dynamic systems ($U0 - V0_{optimal}$)
$\rho_\infty^{max} = 1, \quad \rho_\infty^s = 0$	Discontinuous acceleration zero-order displacement and zero-order velocity overshoot behaviors ($U0 - V0_{DA}$) family of algorithms
$\rho_\infty^{max} = 1,$ $\rho_\infty^s = \frac{1 - \rho_\infty^{min}\rho_\infty^{max}}{\rho_\infty^{min} + \rho_\infty^{max} + \rho_\infty^{min}\rho_\infty^{max}}$	Continuous acceleration zero-order displacement and zero-order velocity overshoot behavior ($U0 - V0_{CA}$) family of algorithms
$\rho_\infty^{max} = 1, \quad \rho_\infty^s \neq \rho_\infty^{min}, \rho_\infty^s \neq 0$ $\rho_\infty^s \neq \frac{1 - \rho_\infty^{min}\rho_\infty^{max}}{\rho_\infty^{min} + \rho_\infty^{max} + \rho_\infty^{min}\rho_\infty^{max}}$	Family of optimal algorithms with a zero-order displacement and zero-order velocity overshoot behaviors
$\rho_\infty^{min} = \rho_\infty^{max} = \rho_\infty^s$	Optimal numerically dissipative family of algorithms with zero-order displacement and first-order velocity overshoot behaviors ($U0 - V1_{optimal}$), wherein the Optimal U0V1 algorithm is naturally recovered
$\rho_\infty^{min} = \rho_\infty^{max}, \quad \rho_\infty^s = 0$	Discontinuous acceleration zero-order displacement and first-order velocity overshoot behaviors ($U0 - V1_{DA}$), wherein the Optimal U0V1 algorithm is naturally recovered.
$\rho_\infty^{max} = \rho_\infty^{max},$ $\rho_\infty^s = \frac{1 - \rho_\infty^{min}\rho_\infty^{max}}{\rho_\infty^{min} + \rho_\infty^{max} + \rho_\infty^{min}\rho_\infty^{max}}$	Continuous acceleration zero-order displacement and first-order velocity overshoot behaviors ($U0 - V1_{CA}$) family of algorithms, wherein the HHT algorithm is naturally recovered
Otherwise	Family of algorithms with zero-order displacement and first-order velocity overshoot behaviors

V0 Family of Algorithms

To reiterate, the V0 family of algorithms is characterized by zero-order overshoot for velocity, and no more than first-order overshoot for displacement. Its coefficients are defined as follows,

$$
\begin{aligned}
&\Lambda_1 W_1 = \frac{3 + \rho_\infty^{min} + \rho_\infty^{max} - \rho_\infty^{min}\rho_\infty^{max}}{2(1+\rho_\infty^{min})(1+\rho_\infty^{max})}, \quad \lambda_1 = 1 \\
&\Lambda_2 W_2 = \frac{1}{(1+\rho_\infty^{min})(1+\rho_\infty^{max})}, \qquad \lambda_2 = \frac{1}{2} \\
&\Lambda_3 W_3 = \frac{1}{(1+\rho_\infty^{min})(1+\rho_\infty^{max})(1+\rho_\infty^{s})}, \qquad \lambda_3 = \frac{1}{2(1+\rho_\infty^{s})} \\
&\Lambda_4 W_1 = \frac{3 + \rho_\infty^{min} + \rho_\infty^{max} - \rho_\infty^{min}\rho_\infty^{max}}{2(1+\rho_\infty^{min})(1+\rho_\infty^{max})}, \qquad \lambda_4 = 1 \\
&\Lambda_5 W_2 = \frac{2}{(1+\rho_\infty^{min})(1+\rho_\infty^{max})(1+\rho_\infty^{s})}, \qquad \lambda_5 = \frac{1}{1+\rho_\infty^{s}} \\
&\Lambda_6 W_1 = \frac{2 + \rho_\infty^{min} + \rho_\infty^{max} + \rho_\infty^{s} - \rho_\infty^{min}\rho_\infty^{max}\rho_\infty^{s}}{(1+\rho_\infty^{min})(1+\rho_\infty^{max})(1+\rho_\infty^{s})} \\
&W_1 = \frac{3 + \rho_\infty^{min} + \rho_\infty^{max} - \rho_\infty^{min}\rho_\infty^{max}}{2(1+\rho_\infty^{min})(1+\rho_\infty^{max})}
\end{aligned}
\tag{40}
$$

Once again, the parameters ρ_∞^{min}, ρ_∞^{max}, and ρ_∞^{s} are variables selected by the user and are referred to as the principal roots and spurious root, respectively. They must satisfy the following relation,

$$0 \leq \rho_\infty^{s} \leq \rho_\infty^{min} \leq \rho_\infty^{max} \leq 1 \tag{41}$$

The algorithms contained in the V0 family of the GS4-2 framework can be seen in the table below,

Table 3. Algorithms contained in V0 family of the GS4-2 framework [15]

Parameters	Algorithms
$\rho_\infty^{min} = \rho_\infty^{max} = \rho_\infty^s = 1$	Mid-point rule
$\rho_\infty^{min} = \rho_\infty^{max} = 1, \quad \rho_\infty^s = 0$	Velocity based scheme
$\rho_\infty^{max} = 1, \quad \rho_\infty^{min} = \rho_\infty^s$	Optimal algorithms with zero-order displacement and zero-order velocity overshoot behaviors and yield only minimal numerical dissipation and dispersion for damped dynamic systems ($U0 - V0_{optimal}$)
$\rho_\infty^{max} = 1, \quad \rho_\infty^s = 0$	Discontinuous acceleration zero-order displacement and zero-order velocity overshoot behaviors ($U0 - V0_{DA}$) family of algorithms
$\rho_\infty^{max} = 1,$ $\rho_\infty^s = \frac{1 - \rho_\infty^{min}\rho_\infty^{max}}{\rho_\infty^{min} + \rho_\infty^{max} + \rho_\infty^{min}\rho_\infty^{max}}$	Continuous acceleration zero-order displacement and zero-order velocity overshoot behavior ($U0 - V0_{CA}$) family of algorithms
$\rho_\infty^{max} = 1, \quad \rho_\infty^s \neq \rho_\infty^{min}, \rho_\infty^s \neq 0$ $\rho_\infty^s \neq \frac{1 - \rho_\infty^{min}\rho_\infty^{max}}{\rho_\infty^{min} + \rho_\infty^{max} + \rho_\infty^{min}\rho_\infty^{max}}$	Family of optimal algorithms with a zero-order displacement and zero-order velocity overshoot behaviors
Otherwise	Family of algorithms with zero-order displacement and first-order velocity overshoot behaviors

GS4-2 Explicit Formulation

The explicit formulation of GS4-2, taken from [16], of a second order in time transient system defined by (27) with initial conditions (28) is similar to the implicit form show previously. However, three new parameters are introduced, specifically η_1, η_2, and $\eta_3$in such a way to modify the numerical coefficients defined in the U0 and V0 families of algorithms. Specifically,

$$\Lambda_3 W_3 \rightarrow \Lambda_3 W_3 \eta_1, \qquad \Lambda_5 W_2 \rightarrow \Lambda_5 W_2 \eta_2, \qquad \lambda_3 \rightarrow \lambda_3 \eta_3 \tag{42}$$

These parameters are essentially Boolean operators, taking a value of either 0 or 1. It can be seen that if $\eta_1 = \eta_2 = \eta_3 = 1$ this formulation then recovers the implicit form of GS4-2 whereas it is fully explicit if $\eta_1 = \eta_2 = 0$.

Relation between GS4-2 and GS4-1 Frameworks

It was shown by Tamma et. al in [15] that the GS4-2 framework can recover the GS4-1 framework and therefore can be utilized for both first-order and second-order systems, henceforth referred to as an *i*Integrator. This is done by utilizing an 'adapting formula', shown in [15], given in the table below,

Table 4. 'Adapting Formula' utilized when converting the GS4-2 framework, intended for second order systems, to a first order framework equivalent to GS4-1

For Matrices	Assign $\boldsymbol{M}$ in the GS4-2 framework as $\boldsymbol{M}$ in the first order system Assign $\boldsymbol{C}$ in the GS4-2 framework as $\boldsymbol{K}$ in the first order system Set $\boldsymbol{K}$ in the GS4-2 framework equal to $\boldsymbol{0}$ Assign $\boldsymbol{F}$ in the GS4-2 as $\boldsymbol{F}$ in the first order system
For Variables	Treat $\ddot{\boldsymbol{u}}$ in the GS4-2 framework as $\dot{\boldsymbol{u}}$ in the first order system Treat $\dot{\boldsymbol{u}}$ in the GS4-2 framework as $\boldsymbol{u}$ in the first order system Neglect $\boldsymbol{u}$ in the GS4-2 framework
For Parameters	Set ρ_∞^s in the GS4-2 framework as ρ_∞^s in the GS4-1 framework Set ρ_∞^{max} in the GS4-2 framework equal to 1 Set ρ_∞^{min} in the GS4-2 framework as ρ_∞ in the GS4-1 framework
For U0 Family	Requires $\rho_\infty^s = \rho_\infty^{min} \in [0,1]$ and yield the GS4-1 framework without the selective control feature
For V0 Family	Choose $\rho_\infty^s \leq \rho_\infty^{min} \in [0,1]$ and yield the GS4-1 framework with/without the selective control feature

An illustration of how this adapting formula is applied can be seen below,

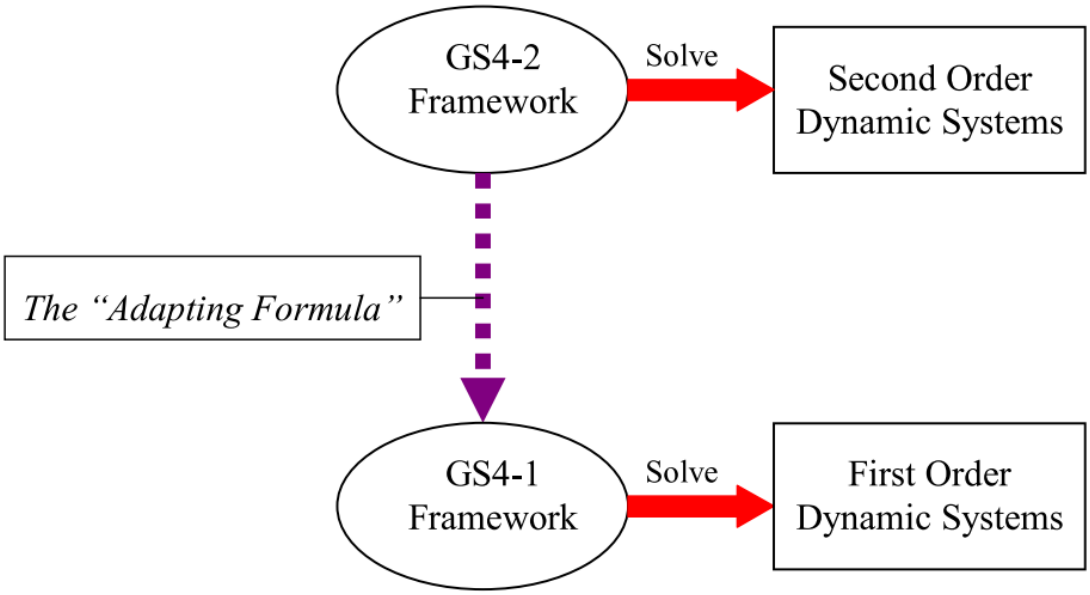


Figure 2. Depiction of the isochronous integration (iIntegrator) framework

It can be seen that for second order systems, one can use the GS4-2 framework without any changes or adjustments, whereas when solving a first order system one must apply the 'adapting formula' show in Table 4. This allows for a universal time stepping framework that can be used for both first order and second order systems.

Chapter 4: Classical C- and F-Processes Heat Conduction Model

The C- and F- Processes model is a novel heat conduction model developed by Tamma et. al [7] that incorporates parabolic heat conduction and hyperbolic heat conduction into a unified theory allowing for both infinite and finite thermal wave speeds as well as any combination therein; it encompasses the Fourier model, the Cattaneo model, and several others that are new.

Fourier Model (Parabolic Heat Conduction)

The Fourier model utilizes Fourier's law as the basis for heat conduction. Fourier's law results in a parabolic partial differential equation and it states that the local heat flux within a specimen is proportional, through the material's thermal conductivity, to the temperature gradient within that specimen,

$$\boldsymbol{q} = -\boldsymbol{k}\nabla T \tag{43}$$

Where $\boldsymbol{q}$ is the heat flux, $\boldsymbol{k} = k\boldsymbol{I}$ where k is the thermal conductivity, and T is the temperature. This law was determined experimentally, not derived from first principles, and is the predominant theory for heat conduction [8]. However, it is generally known that Fourier's law allows for thermal waves to propagate at infinite speeds [9]; which is inaccurate for certain engineering applications.

Cattaneo Model (Hyperbolic Heat Conduction)

The Cattaneo model is a hyperbolic heat conduction model, and is an extension of Fourier's law that incorporates a time dependence on the heat flux term; see Cattaneo [10]. This method extends Fourier's law by incorporating a partial derivative of the heat flux with respect to time that is scaled by a relaxation factor,

$$\boldsymbol{q} + \boldsymbol{\tau}\frac{\partial \boldsymbol{q}}{\partial t} = -\boldsymbol{k}\nabla T \tag{44}$$

Where $\boldsymbol{\tau}$ is the relaxation parameter. Due to the inclusion of the additional terms, the method allows for finite thermal wave speeds allowing for the capture of thermal shock waves and other phenomena. It should be noted that in the steady state, the hyperbolic heat conduction equation above becomes Fourier's law because the time derivative of the heat flux becomes zero. Furthermore, for this model the temperature propagation speed is defined as,

$$c_T = \sqrt{\frac{k}{\rho c \tau}} \tag{45}$$

where c is the heat capacity of the material.

Jeffreys' Model

The Jeffreys' model of heat conduction is a formulation developed by Joseph and Preziosi [17] based upon a formulation that was derived to explain wave propagation in solids, specifically the Earth's mantle. It is given by

$$\sigma + \lambda_1 \frac{\partial \sigma}{\partial t} = -\eta_0 \left(\dot{\gamma} + \lambda_2 \frac{\partial \dot{\gamma}}{\partial t} \right) \tag{46}$$

where σ, $\dot{\gamma}$, η_0, λ_1, and λ_2 are the stress, strain rate tensor, zero shear rate viscosity, relaxation time, and retardation time, respectively. As mentioned, this equation was extended to heat conduction by Joseph and Prezisi in the form of,

$$\boldsymbol{q} + \tau \frac{\partial \boldsymbol{q}}{\partial t} = -k \nabla T - \tau k_1 \frac{\partial (\nabla T)}{\partial t} \tag{47}$$

Where the thermal conductivity k is the summation of the effective thermal conductivity k_1 and the elastic conductivity, k_2, such that $k = k_1 + k_2$. This model is noteworthy because it can recover the Cattaneo model easily by assuming $k_1 = 0$; however it can only recover a Fourier-like diffusive model with relaxation. The above formulation, (47), can also be written as,

$$\boldsymbol{q} + \tau \frac{\partial \boldsymbol{q}}{\partial t} = -k \left(\nabla T - K \frac{\partial (\nabla T)}{\partial t} \right) \tag{48}$$

where $K = \frac{\tau k_1}{k}$. Furthermore it was shown in [18] that the so called Dual-Phase-Lag (DPL) model which is an analogous representation has anamolies and is a physically incorrect model. For lagged parabolic heat conduction, in which there is a relaxation parameter for both the temperature (τ_T) and the flux (τ_q), it is equivalent to the Jeffreys' model if we set,

$$\begin{matrix} \tau \rightarrow \ \tau_q \\ K = \tau F_T \rightarrow \ \tau_T \end{matrix} \tag{49}$$

Classical C- and F- Heat Conduction Model

The C- and F- heat conduction model developed by Tamma et al. [11], is a model developed from first principles via the Boltzmann transport equation (BTE) and it allows for the simultaneous coexistence of infinite speed and finite speed heat conduction admitting both Fourier type (F-processes) and Cattaneo type (C-processes) as described previously. This allows for a generalized framework consisting of a wide variety of models explained above.

Assuming the simultaneous co-existence of C-processes and F-processes, and allowing for a linear combination to determine the total effective heat flux, the generalized C- and F- model can be proposed as follows,

$$\boldsymbol{q_F} = \ -k_f \nabla T = \ -F_t k \nabla T \tag{50}$$

$$\boldsymbol{q_C} + \tau \left(\frac{\partial \boldsymbol{q_C}}{\partial t} \right) = \ -k_C \nabla T = \ -(1 - F_T) k \nabla T \tag{51}$$

$$\boldsymbol{q} = \boldsymbol{q_C} + \boldsymbol{q_F} \tag{52}$$

where $\boldsymbol{q}$ is the total heat flux via heat conduction which is a linear combination of the Cattaneo type and Fourier type heat conduction, and k is the total effective conductivity which is a linear combination of the effective Fourier conductivity k_f and effective Cattaneo conductivity k_C, such that $k = k_f + k_c$. In the equations above a dimensional heat transfer number was utilized which is known as the *Macroscale Heat Conduction Model Number*, F_T and it is defined as,

$$F_T := \frac{Fourier\ Conductivity\ (k_F)}{Fourier\ Conductivity\ (k_F) + Cattaneo\ Conductivity\ (k_c)} \tag{53}$$

Where $F_T \in [0,1]$, such that when $F_T = 1$, Fourier type (F-processes) are recovered and when $F_T = 0$ Cattaneo type (C-processes) are recovered allowing for the C- and F- model to bridge between the infinite and finite wave speed models. Furthermore for $F_T \in (0,1)$ the C- and F- model can also recover the Jeffreys' model.

Derivation from Boltzmann Transport Equation

The derivation of the C- and F- model as described in [11] via the Boltzmann Transport Equation (BTE) is as follows. The general form off the BTE is given as,

$$\frac{\partial f}{\partial t} + \ v \cdot \nabla f + a \cdot \frac{\partial f}{\partial v} = \left(\frac{\partial f}{\partial t}\right)_{scatt} \tag{54}$$

where f is a non-equilibrium distribution; such as Maxwell-Boltzmann for identical individual particles, Bose-Einstein for boson type particles, or Fermi-Dirac for fermion type particles, $v(\omega)$ is the heat carrier velocity which is also known as the first speed of sound for a given medium, $a(v)$ is the particle acceleration, T is the absolute temperature, ω is the frequency, and $E(\omega)$ is the energy state. The three terms on the left-hand side of the equation can be described as the change in particles over time, the convective inflow of particles in physical space, and the convective inflow of particles due to acceleration respectively. Whereas the right-hand term is the change in particles inside the specified control volume due to collisions. Working under the assumption that the heat carrier velocity is constant over a wide range of frequencies, $\frac{\partial f}{\partial v}$ can be approximated as zero removing that term from the equation above. Furthermore, the energy and temperature gradient tend to alter f; because these processes tend to be

opposed processes they attain equilibrium. Therefore, the right-hand term is linearly approximated utilizing the relaxation time approximation which simplifies to,

$$\left(\frac{\partial f}{\partial t}\right)_{scatt} = \frac{\partial (f - f^0)}{\partial t} = \frac{f^0 - f}{\tau} \tag{55}$$

Where f^0 is the distribution at equilibrium, $\frac{\partial f^0}{\partial t} = 0$, and $\tau(\omega, v)$ is the relaxation time.

Derivation of the Fourier Model

For one-dimensional, steady state cases (54) reduced to,

$$v_x \left(\frac{\partial f}{\partial t}\right) = \frac{f_0 - f}{\tau} \tag{56}$$

The theory is derived through the *local thermal equilibrium* (LTE) premise, however in cases where this does not apply the BTE must be solved directly. However, since LTE is valid due to the presence of the temperature gradient, and $\frac{\partial f}{\partial x} \sim \frac{\partial f^0}{\partial x}$, then $\frac{\partial f}{\partial x}$ can be approximated as,

$$\frac{\partial f}{\partial x} = \frac{df^0}{dT}\frac{dT}{dx} \tag{57}$$

Taking (56) and (57), using the relation above and multiplying each side by $v_x \hbar\omega D(\omega)$, and then integrating over the frequency domain one obtains the following expression,

$$\int_0^{\omega_D} v_x^2 \hbar\omega \frac{df^0}{dT}\frac{dT}{dx} D(\omega) d\omega = \int_0^{\omega_D} v_x \hbar\omega D(\omega) \frac{f^0 - f}{\tau} d\omega \tag{58}$$

where ħ is Planck's number divided by 2π, $D(\omega)$ is the density of states, and ω_D is the Debye cutoff angular frequency. Because v_x is an odd function and f^0 is an even function, the first term in the subtraction is equal to zero implying that the net conduction heat flux is zero for f^0. The flux can then be given as,

$$q(x) = \int_0^{\omega_D} v_x f(x) \hbar\omega D(\omega) \tag{59}$$

Using the flux definition above, (58) can be reduced to,

$$q(x) = -\frac{dT}{dx} \int_0^{\omega_D} v_x^2 \tau \frac{df^0}{dT} \hbar\omega D(\omega)\, d\omega \tag{60}$$

And the specific heat is given as,

$$c = \int_0^{\omega_D} \frac{df^0}{dT}\, \hbar\omega D(\omega) d\omega \tag{61}$$

Therefore, based upon the kinetic theory, the overall thermal conductivity is related to the specific heat as follows,

$$k = \int_0^{\omega_D} v_x^2 \tau \frac{df^0}{dT}\, \hbar\omega D(\omega) d\omega = \frac{1}{3}\, cv\lambda \tag{62}$$

where k is the overall thermal conductivity, c is the overall specific heat per unit volume, v is the average speed of the heat carrier and λ is the mean free path. From here, substituting (62) and (61) into the flux definition, one obtains the Fourier model in a single dimension form,

$$q(x) = -k \frac{dT}{dx} \tag{63}$$

Derivation of the Cattaneo Model

Under the relaxation time approximation and temperature gradient approximation the transient one-dimensional BTE equation becomes,

$$\frac{\partial f}{\partial t} + v_x \frac{df^0}{dT} \frac{dT}{dx} = \frac{f^0 - f}{\tau} \tag{64}$$

Then following the same procedure as for the Fourier derivation, (64) is multiplied by $v_x \hbar\omega D(\omega)$ and then integrated over the entire frequency range to yield,

$$\int_0^{\omega_D} v_x \hbar\omega D(\omega)\left(\frac{\partial f}{\partial t} + v_x \frac{df^0}{dT}\frac{dT}{dx} - \frac{f^0 - f}{\tau}\right) = 0 \tag{65}$$

From which, following the Fourier derivation, the Cattaneo model is readily derived as,

$$q + \tau\frac{dq}{dt} = -k\frac{dT}{dx} \tag{66}$$

Derivation of the C- and F- Model

The only difference between the Fourier model and the Cattaneo model is the presence of the $\frac{\partial f}{\partial t}$ term. The importance of this term is dependent on the distribution function f being utilized. The main three functions utilized are,

1. **Maxwell-Boltzmann:**

$$f\big(E(\omega)\big) = f_{MB}\big(E(\omega)\big) \coloneqq \frac{1}{Ae^{\frac{E(\omega)}{T\kappa_B}}} \tag{67}$$

2. **Bose-Einstein:**

$$f\big(E(\omega)\big) = f_{BE}\big(E(\omega)\big) \coloneqq \frac{1}{Ae^{\frac{E(\omega)}{T\kappa_B}} - 1} \tag{68}$$

3. **Fermi-Dirac**

$$f\big(E(\omega)\big) = f_{FD}\big(E(\omega)\big) \coloneqq \frac{1}{Ae^{\frac{E(\omega)-E_F}{T\kappa_B}} + 1} \tag{69}$$

where A is a normalized constant, κ_B is the Boltzmann's constant, and E_F is the Fermi energy. It can be seen that for significantly large enough frequencies the above functions behave as,

$$f\big(E(\omega)\big) \approx 0, \qquad \forall\omega \in [\omega_T, \omega_D] \tag{70}$$

Which means that,

$$\frac{\partial f\big(E(\omega)\big)}{\partial t} \approx 0, \qquad \forall \omega \in [\omega_T, \omega_D] \tag{71}$$

Under these approximations, we obtain the total heat flux as,

$$q = \int_0^{\omega_T} v_x \hbar\omega D(\omega) d\omega + \int_{\omega_T}^{\omega_D} v_x f \hbar\omega D(\omega) d\omega = q_C + q_F \tag{72}$$

It is assumed that till a certain frequency threshold ω_T, the flux are Cattaneo type processes, and that from ω_T to ω_D they are Fourier type processes. This is reiterated by the left hand side of the equation where the terms q_C and q_F represent the Cattaneo type and Fourier type processes, respectively. If the entire conduction process is considered, the BTE can be multiplied by $v_x \hbar\omega D(\omega)$ and integrated over the entire frequency domain to yield,

$$\begin{aligned} \int_0^{\omega_T} v_x \hbar\omega D(\omega) \frac{\partial f}{\partial t} d\omega + \int_0^{\omega_T} v_x^2 \hbar\omega D(\omega) \frac{\partial f}{\partial t} d\omega + \int_{\omega_T}^{\omega_D} v_x^2 \hbar\omega D(\omega) \frac{\partial f}{\partial t} d\omega \\ = \int_0^{\omega_T} v_x \hbar\omega D(\omega) \frac{f^0 - f}{\tau} d\omega + \int_{\omega_T}^{\omega_D} v_x \hbar\omega D(\omega) \frac{f^0 - f}{\tau} d\omega \end{aligned} \tag{73}$$

However instead of using the equation above, two equations will be developed independently from the BTE associated with the C-type and F-type processes as follows,

$$\int_0^{\omega_T} \left(v_x \hbar\omega D(\omega) \frac{\partial f}{\partial t} + v_x^2 \hbar\omega D(\omega) \frac{\partial f}{\partial t} \right) d\omega = \int_0^{\omega_T} v_x \hbar\omega D(\omega) \frac{f^0 - f}{\tau} d\omega \tag{74}$$

for C-type processes and,

$$\int_{\omega_T}^{\omega_D} v_x^2 \hbar\omega D(\omega) \frac{\partial f}{\partial t} d\omega = \int_{\omega_T}^{\omega_D} v_x \hbar\omega D(\omega) \frac{f^0 - f}{\tau} d\omega \tag{75}$$

for F-type processes. The total thermal conductivity can then be defined as,

$$k = \int_0^{\omega_T} v_x^2 \tau \frac{df^0}{dT} \hbar\omega D(\omega) d\omega + \int_{\omega_T}^{\omega_D} v_x^2 \tau \frac{df^0}{dT} \hbar\omega D(\omega) d\omega = k_C + k_F \tag{76}$$

A non-dimensional *heat conduction model number* can be defined as the following ratio,

$$F_T = \frac{\int_{\omega_T}^{\omega_D} v_x^2 \tau \frac{df^0}{dT} \hbar\omega D(\omega) d\omega}{\int_0^{\omega_D} v_x^2 \tau \frac{df^0}{dT} \hbar\omega D(\omega) d\omega} \tag{77}$$

For the C- and F- heat conduction model, for the one dimensional case we have,

$$q_F = -F_T k_F \frac{dT}{dx} \tag{78}$$

$$q_C + \tau \frac{\partial q_C}{\partial t} = -(1 - F_T) k \frac{dT}{dx} \tag{79}$$

$$q = q_F + q_C \tag{80}$$

Two-Field Form and GS4 Implementation

When implementing this model a two-field form; where both temperature and the heat fluxes are independent variables, is utilized. This is done for two reasons. Firstly, to allow for easier implementation of flux boundary conditions, because even though heat fluxes are a natural boundary condition in the single field form they can considered essential boundary conditions as the fluxes are independent variables. Second, because it allows the system to be only first-order in time, whereas a single field form; where temperature would be the only independent variable, would require first and second order considerations. Therefore, when considering the governing equation for a moving heat source,

$$\rho c_p \frac{\partial \boldsymbol{T}}{\partial t} - \boldsymbol{v} \cdot \nabla T = -\nabla \cdot \boldsymbol{q} + \boldsymbol{S} \tag{81}$$

where ρ is the density and c_p is the constant pressure specific heat. Along with the C- and F- model, utilizing the two-field form one can define a first order in time, multi-dimensional space system of equations as follows,

$$\boldsymbol{M}\dot{\boldsymbol{u}} + \boldsymbol{K}\boldsymbol{u} = \boldsymbol{F} \tag{82}$$

Where the matrices are defined as follows,

$$\boldsymbol{M} = \begin{bmatrix} \rho c_p \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{\tau} \end{bmatrix} \tag{83}$$

$$\boldsymbol{K} = \begin{bmatrix} -\boldsymbol{v}\cdot\boldsymbol{\nabla} & \boldsymbol{\nabla}\cdot & \boldsymbol{\nabla}\cdot \\ \boldsymbol{k}F_T\boldsymbol{\nabla} & \boldsymbol{I} & \boldsymbol{0} \\ (1-F_T)\boldsymbol{k}\boldsymbol{\nabla} & \boldsymbol{0} & \boldsymbol{\tau}(\boldsymbol{v}\cdot\boldsymbol{\nabla}) \end{bmatrix} \tag{84}$$

$$\boldsymbol{F} = \begin{bmatrix} \boldsymbol{S} + (1-F_T)\boldsymbol{\tau}\left(\dfrac{\boldsymbol{dS}}{\boldsymbol{dt}}\right) \\ \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix} \tag{85}$$

where $\boldsymbol{I}$ and $\boldsymbol{0}$ are the identity and null matrix, respectively, and are $n \times n$ matrices, where n is the length of a variable matrix for a specific particle. These variable vectors are,

$$\boldsymbol{u} = \begin{bmatrix} \boldsymbol{T} \\ \boldsymbol{q_F} \\ \boldsymbol{q_C} \end{bmatrix} \tag{86}$$

$$\dot{\boldsymbol{u}} = \begin{bmatrix} \dot{\boldsymbol{T}} \\ \dot{\boldsymbol{q}}_{\boldsymbol{F}} \\ \dot{\boldsymbol{q}}_{\boldsymbol{C}} \end{bmatrix} \tag{87}$$

where T is a scalar, and both $\boldsymbol{q_C}$ and $\boldsymbol{q_F}$ are $m \times 1$ vectors; where m is the number of spatial dimensions. It can be seen that (*82*) is a first order in time, linear system of equations, so the application of first-order solution contained within the GS4 *i*Integration framework can be utilized.

Chapter 5: MPS Framework with Implementation of C- and F- Model

Introduction

This chapter will cover the basic theory, implementation, and numerical validation, for the C- and F- model with regards to problems not involving internal heat generation nor external source terms; stationary or moving, as well as it provides additional numerical examples,. More complex formulations involving internal heat generation and external source terms will be presented in the chapters to follow. The C- and F- model will be compared with the Fourier Model, Cattaneo Model, and Jeffreys' Model; as to validate the model with respect to various types of heat conduction; Fourier-type parabolic heat conduction, Cattaneo-type hyperbolic heat conduction, and Jeffreys'-type lagged parabolic heat conduction. Additional examples will be shown as well to further illustrate

the method's capability. The formulation presented herein covers a broad range of practical engineering applications attributable to the C- and F- model's versatility.

Theory

Numerical heat conduction utilizing the C- and F- model without the presence of internal heat generation or an external heat source can be described with constitutive equations (78)-(80) and the governing equation (81). As noted previously, this type of problem can be written in a two-field form as seen in equations (82)-(87), where the source matrix ($\boldsymbol{S}$) in equation (85) is a zero vector. For the transient examples, the GS4 *i*Integration framework for first order systems; as shown previously, will be implemented for the time stepping.

Implementation

The algorithm for the problems in this chapter is described in Figure 3. After the initial inputs are taken from the user the weight matrix, as prescribed by the weight function, is determined; from which the stiffness ($\boldsymbol{K}$) and mass ($\boldsymbol{M}$) can be computed. It should be noted that if this simulation involved moving particles, the weight matrix would need to be computed at the beginning of every time step. However since all the following simulations involve stationary media, the weight matrix will only have to be calculated once.

Once the initial temperatures and fluxes are computed, the boundary conditions are applied, and the algorithmic values are calculated as per the GS4 *i*Integration framework.

The actual values are determined and then updated, and then the loop is performed until the end of the simulation.

Furthermore, because a two-field form is being utilized in the constitutive equations, both the temperature and heat fluxes can be set as Type 1, or essential, boundary conditions; meaning that they can be directly applied as constraints on the Jacobian matrix during the numerical simulation.

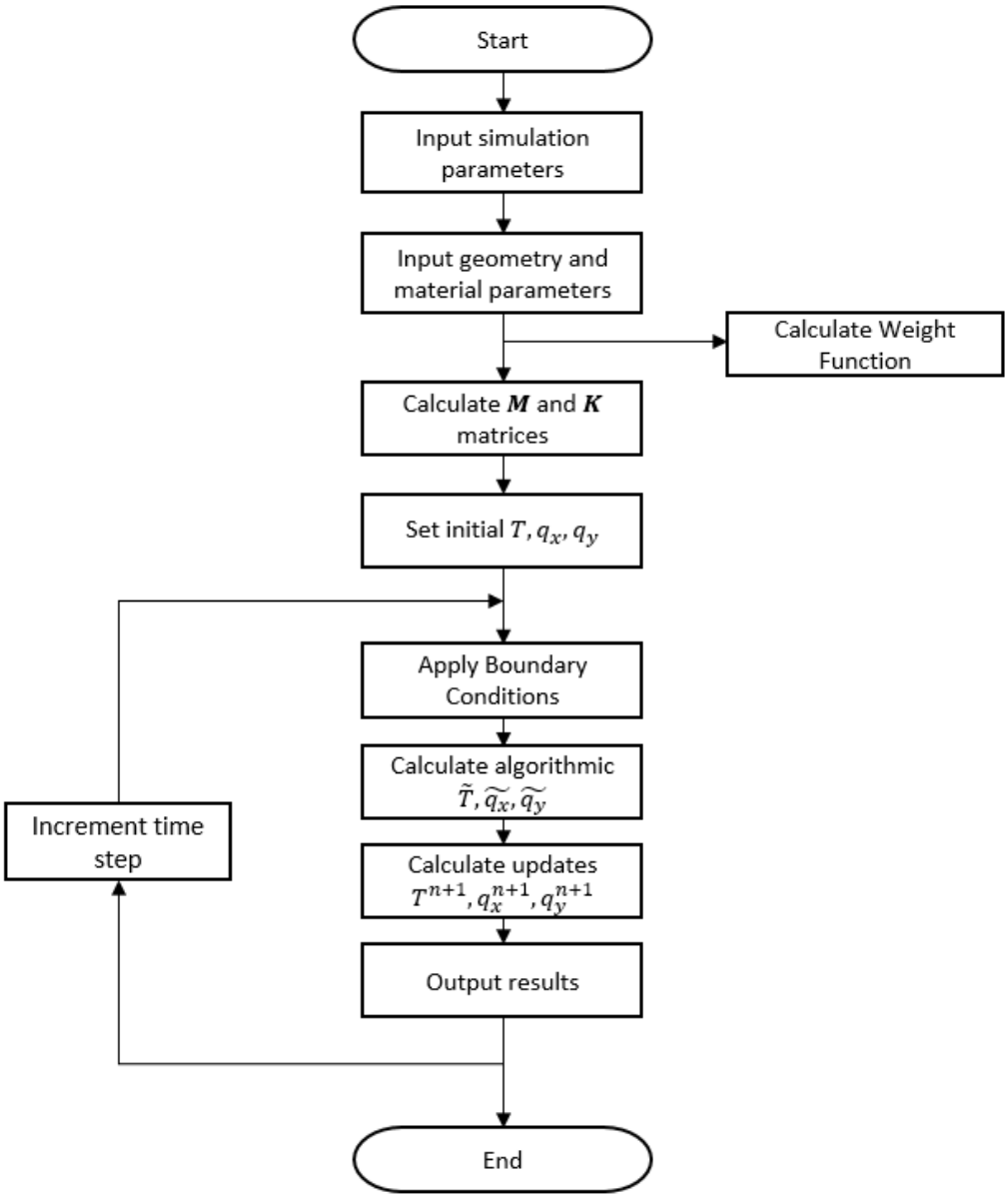


Figure 3. Computational algorithm of proposed framework encompassing Chapter 5 examples

Numerical Validation

Various numerical examples are illustrated to validate the temperature distributions resulting from the Fourier-type parabolic heat conduction, Cattaneo-type hyperbolic heat conduction, and Jeffreys'-type lagged parabolic conduction, respectively.

For the steady state cases, a square domain with specified boundary conditions that has been solved analytically will be used as a basis for comparison. For the transient cases, quasi-one-dimensional and two-dimensional problems will be solved and then results will be compared against analytical results.

Fourier-type Heat Conduction (Parabolic Heat Conduction [$F_T = 1$])

To validate the model for parabolic heat conduction, both a steady state problem and a transient problem will be presented. The results of both problems will be compared to known analytical solutions.

Steady State Validation

Consider a 2D diffusion problem in a square domain with side lengths of $1\ [m]$ subject to the boundary conditions [19],

$$\nabla^2 T(x,y) = 0; \tag{88}$$

$$T(x,0) = x, \qquad T(x,1) = T(0,y) = T(1,y) = 0 \tag{89}$$

An illustration of the problem can be seen in Fig. 4,

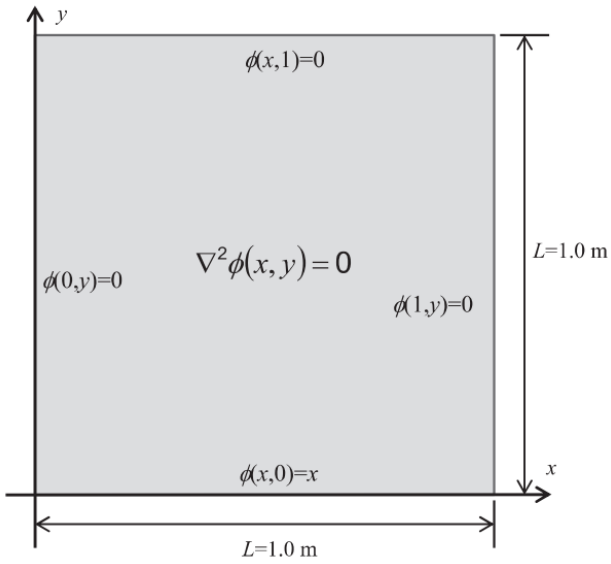


Figure 4. Illustration of 2D Steady State Diffusion Problem, (note: $\phi = T$)

with the corresponding exact solution as given as,

$$T(x,y) = \sum_{n=1}^{\infty} \left(\frac{2(-1)^{n+1}}{(n\pi)\, sinh(n\pi)} sinh\big(n\pi(1-y)\big)\, sin(n\pi x) \right) \tag{90}$$

This problem is solved using the C- and F- model with $F_t = 1$ which corresponds to Fourier-type processes with the computational domain being discretized into 1089, 2601, and 4489 particles; alternatively it is described by 32, 50, and 66 divisions in each dimension respectively, to illustrate spatial convergence of the model. For the weight function the cutoff radius utilized is $r_e = 1.05 \left(\frac{L}{Divisions}\right)$. Also the temperature values for all particles at $x = 0.5\ [m]$as well as for all particles at $y = 0.5\ [m]$ will be compared to their corresponding exact solutions. Plots of the comparisons discussed above can be seen below Figs. 5-6,

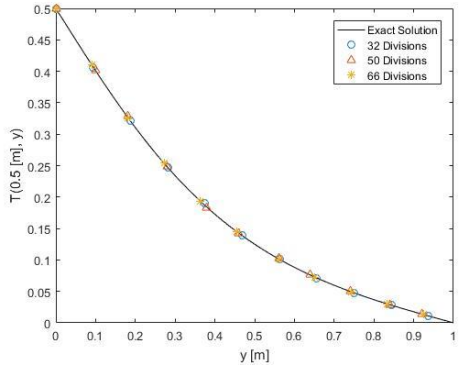


Figure 5. Comparison with exact solution for particles at x = 0.5 [m]

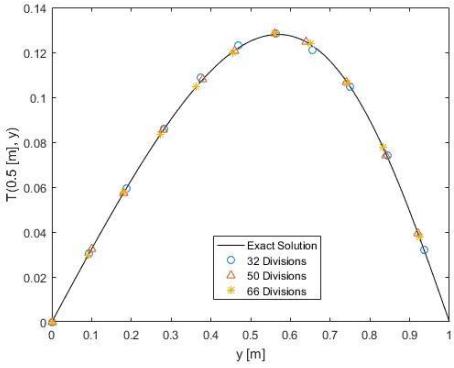


Figure 6. Comparison with exact solution for particles at y = 0.5 [m]

It can be seen qualitatively that there is significant agreement between the simulations and the exact solution. Furthermore, the temperature distribution resulting from this framework matches that from the results from the same simulation conducted utilizing FEM; the contour plots of the temperature distribution follow next (Figs. 7 and 8).

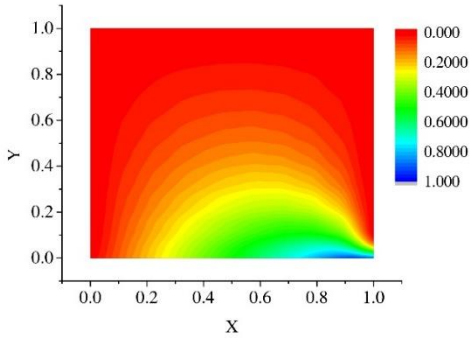


Figure 7. Temperature distribution resulting from proposed framework for steady state Fourier-type processes

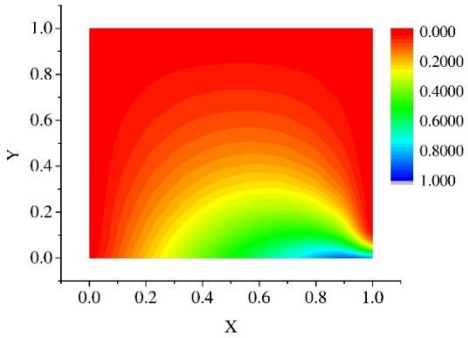


Figure 8. Temperature distribution resulting from FEM for steady state Fourier-type processes

For a better illustration of the error versus spatial discretization, the error for the central particle in the square; located at $(x, y) = (0.5, 0.5)\ [m]$, was determined and can be seen below in Fig. 9,

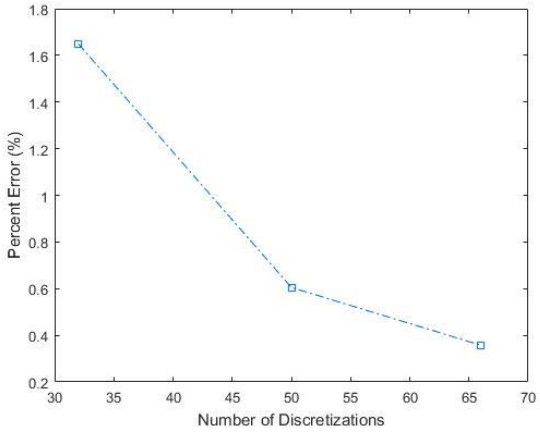


Figure 9. Percent error at center point

From the plots above, it is shown that there is significant agreement between the numerical simulation results and the exact solution, validating the particle-based method utilized in this work can be used to achieve promising spatial approximation required in the numerical simulations.

Transient Validation

Consider a quasi-one dimensional semi-infinite solid with a constant surface temperature boundary condition. This problem is being utilized because there exists an analytical solution and is therefore a good comparison for validation. The analytical solution for the temperature and surface heat flux, from [20] are as follows,

$$\frac{T(x,t)-T_s}{T_i-T_s}=\operatorname{erf}\left(\frac{x}{2\sqrt{\alpha t}}\right) \tag{91}$$

$$q_s''(t)=\frac{k(T_s-T_i)}{\sqrt{\pi\alpha t}} \tag{92}$$

where T_i and T_s are the initial temperature and constant surface temperature, respectively, erf( ) is the Gaussian error function, and α is the thermal diffusivity which is given as,

$$\alpha = \frac{k}{\rho c_p} \tag{93}$$

For the simulation, we will utilize a 2D slender rectangular domain, with geometry and material properties as seen in the Table 5 below,

Table 5. Geometry and material parameters for Fourier quasi-one-dimensional transient simulation

Parameter	Value
Length	100 mm
Width	1 mm
Specific Heat (c_p)	1 J/kg/°K
Density (ρ)	1 kg/m^3
Thermal Conductivity (k)	10 W/m/°K

boundary condition,

$$T_{BC} = 100\ [°K] \tag{94}$$

For this problem the explicit form of GS4-2 was utilized with,

$$\eta_1 = \eta_2 = \eta_3 = 1 \tag{95}$$

time stepping parameter,

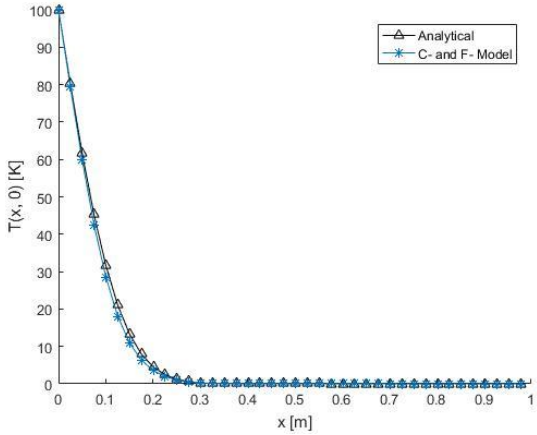
$$dt = 0.0001\ [s] \tag{96}$$

and,

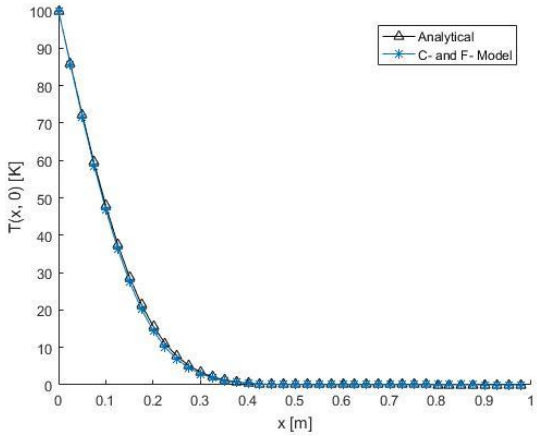
$$\rho_s = 0, \qquad \rho_{min} = 0, \qquad \rho_{max} = 1 \tag{97}$$

which is equivalent to $(\rho_s = 0,\ \rho = 0)$ in GS4-1, resulting in Gear's Method for time stepping. This was chosen due to its damping characteristics. To validate the results of the model, they are compared to the analytical solution at various time steps,

(a)



(b)



(c)

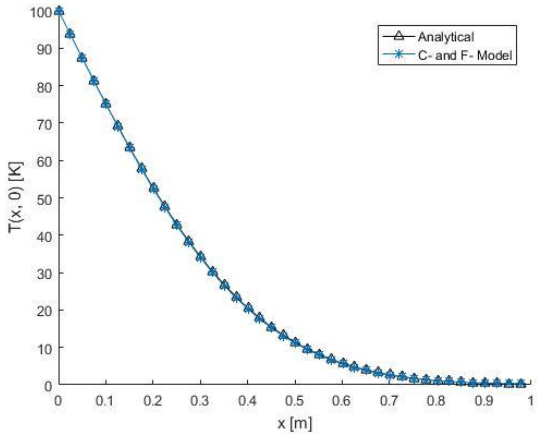


Figure 10. Results comparing simulation results to analytical results for (a) 0.0005 s, (b) 0.0010 s, and (c) 0.005 s

In the figure above, it can be seen that there is good agreement between the numerical results and the analytical solution. Furthermore, a time history for three particles located at $x = 0.2005, 0.4010, 0.6015\ [m]$ can be seen below Fig. 11,

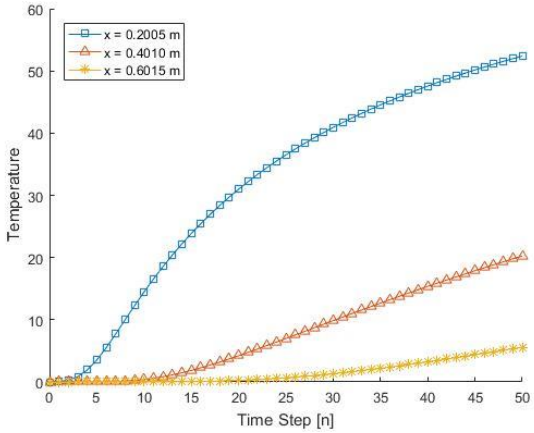


Figure 11. Time history for Fourier process heat conduction transient simulation for particles at x = 0.2005, 0.4010, 0.6015 [m]

To further illustrate the transient results, the contour plots for both the temperature and the flux in the x-direction can be seen below in Fig. 12, Figure 12 shows that this problem is in fact quasi-one dimensional because of the isothermal and isoflux vertical sections present in the figures. It should be noted that the contour plots are not scaled the same in the x and y directions; this is done to facilitate the illustration of the data.

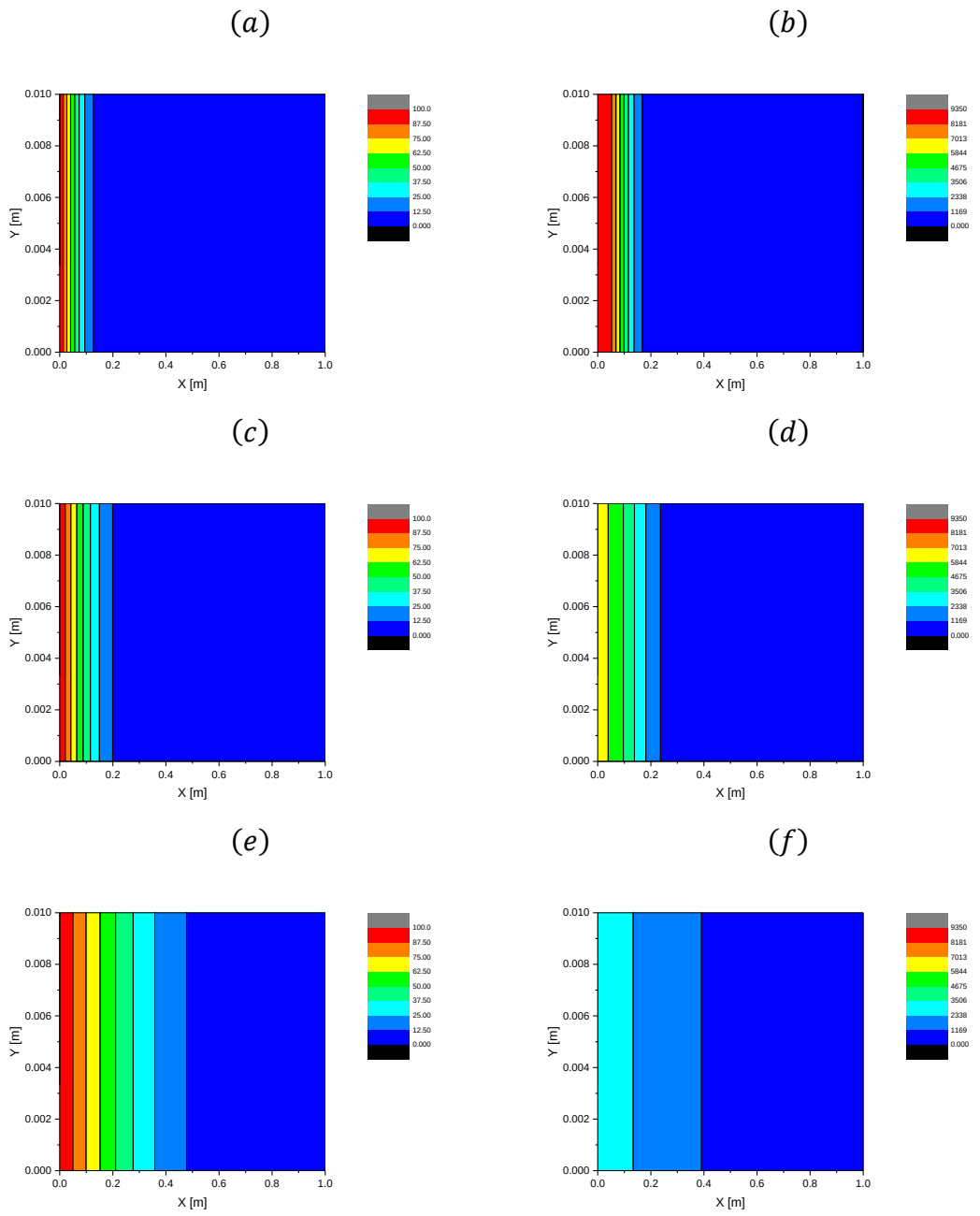


Figure 12. Contour plots for temperature; (a), (c), (e) at time steps n = 5, 10, 50 respectively and heat flux in the x-direction (b), (d), (f) at time steps n = 5, 10, 50 respectively

For an error comparison, the average percent error between the analytical and simulated results were compared for all particles with temperatures greater than 10^{-3}, this was done to prevent the influence of large errors for numerically negligible values, see Fig. 13,

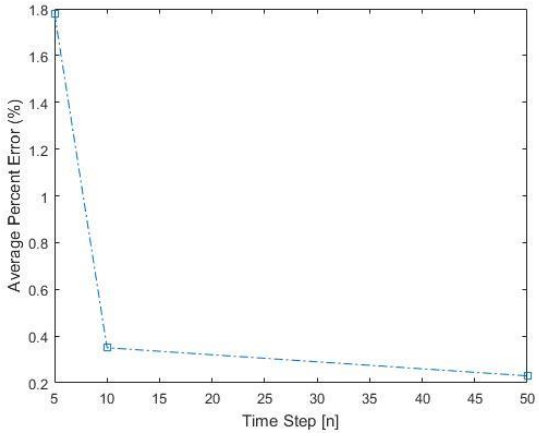


Figure 13. Average percent error for the 5th, 10th, and 50th time steps

It can be seen in Figure 10 that there is significant agreement between the C- and F-model implementation and the analytical solution for temperature distribution at various time steps.

For further corroboration, the simulated surface heat flux at the boundary can be compared to the flux calculated by (92). A time history of the flux for both the analytical solution and the simulation results can be seen below in Fig. 14,

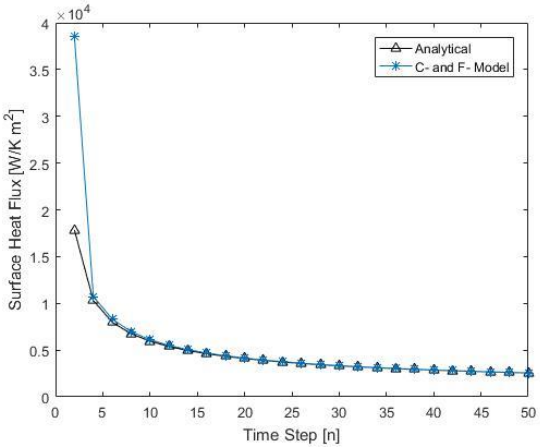


Figure 14. Time history of surface heat flux

The error comparison starting at the fifth time step, in order to distance the results from the large gradients present in the initial few time steps, can be seen in the Fig, 15 below.

The first four time steps were ignored in the error comparison due to the extremely high gradients. However, it can be seen that the flux stabilizes after the fourth time step allowing for such considerations.

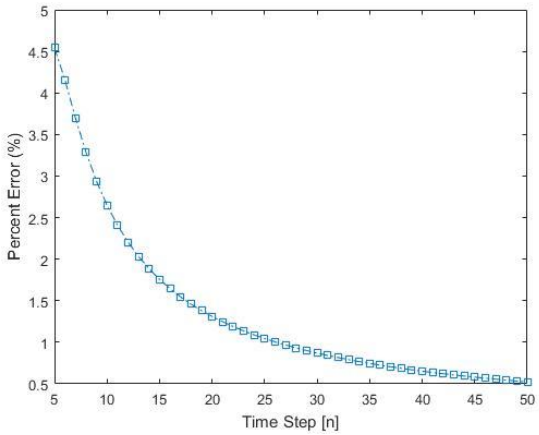


Figure 15. Percent error for heat flux

Once again, when comparing the surface heat flux we see that there is significant agreement between the simulation results and the analytical results. Therefore the implementation of the C- & F- model within this framework can accurately recover Fourier-type conduction problems.

Cattaneo-type Heat Conduction (Hyperbolic Heat Conduction [$F_T = 0$])

To validate the model with respect to hyperbolic heat conduction, both a steady state and a transient problem will be presented. The steady state problem will be the same one utilized in the parabolic heat conduction validation. For the transient problem, a quasi-one dimensional problem with the corresponding analytical solution presented in [21] will be simulated.

Steady State Validation

To validate the model's ability to capture hyperbolic heat conduction in the steady state, the same problem used for the Fourier validation will be simulated. Theoretically these two formulations should produce the same output because the models only differ in the temporal sense. Therefore the results should look identical to those previously presented for the Fourier-type validation. Once again, for an illustration of the geometry for this problem see Figure 4; the boundary conditions for which are,

$$\nabla^2 T(x,y) = 0; \tag{98}$$

$$T(x,0) = x, \qquad T(x,1) = T(0,y) = T(1,y) = 0 \tag{99}$$

with the corresponding the exact solution,

$$T(x,y) = \sum_{n=1}^{\infty} \left(\frac{2(-1)^{n+1}}{(n\pi)\, sinh(n\pi)} sinh\big(n\pi(1-y)\big)\, sin(n\pi x) \right) \tag{100}$$

This problem is solved using the C- and F- model with $F_t = 0$; which corresponds to Cattaneo-type processes. The computational domain for this problem is discretized into 4489 particles; 66 divisions in each dimension respectively. For the weight function the cut-off radius utilized is $r_e = 1.05 \left(\frac{L}{Divisions}\right)$. For comparison, the temperature values for all particles located at $x = 0.5\ [m]$as well as for all particles located at $y = 0.5\ [m]$ will be compared to the exact solution as well as the Fourier solution for the same level of discretization. Plots of the comparisons discussed above can be seen below in Figs. 16-17,

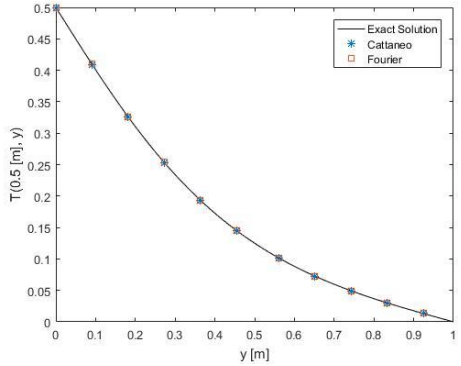


Figure 16. Comparison of steady state Cattaneo solution with exact solution and Fourier solution for particles at x = 0.5 [m]

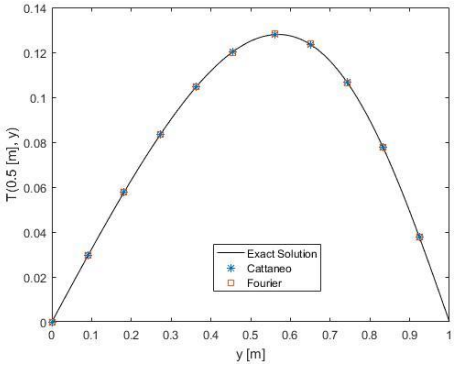


Figure 17. Comparison of steady state Cattaneo solution with exact solution and Fourier solution for particles at y = 0.5 [m]

The Cattaneo steady state results shown above match with both the exact solution and Fourier-type numerical results, corroborating the previous claim that the steady state results are the same for both the Fourier-type process and the Cattaneo-type process. Therefore, the implementation of the C- & F- model within this framework can accurately recover Cattaneo-type conduction problems.

Transient Validation

To validate the implementation for Cattaneo-type transient problems, consider a quasi-one-dimensional beam with geometry and material parameters as follows [21],

Table 6. Geometry and Material parameters for Quasi-one-dimensional Cattaneo Example

Parameter	Value
Length	1 m
Width	0.002 m
Specific Heat (c_p)	1 J/kg/°K
Material Density (ρ)	1 kg/m^3
Thermal Conductivity (k)	0.5 W/m/°K
Relaxation Parameter (τ)	0.5 s

It is worth noting that the relaxation time is usually at the magnitude of 1e-12s in the engineering practices. Herein, we use a large value for better illustrating the computational performance of the proposed method. The specific initial and boundary conditions are given as follows,

$$T_i = 0\ [°K], \qquad T_{BC} = 1\ [°K] \tag{101}$$

For this problem the explicit form of GS4-2 was utilized with,

$$\eta_1 = \eta_2 = \eta_3 = 1 \tag{102}$$

and time stepping parameter,

$$dt = 0.001\ [s] \tag{103}$$

This example will illustrate the model's ability to capture the wave propagation along the length of the beam, moving with a wave speed as shown below,

$$c_T = \sqrt{\frac{k}{\rho c_p \tau}} \tag{104}$$

It should be noted that this framework only possesses first order spatial convergence, which can lead to significant numerical oscillations. These types of oscillations can be damped within the GS4 *i*Integration framework by varying the selectable parameters. To illustrate this capability, results will be shown both with and without controllable numerical dissipation. Specifically, for no numerical dissipation,

$$\rho_\infty^s = 1, \qquad \rho_\infty^{min} = 1, \qquad \rho_\infty^{max} = 1 \tag{105}$$

and for maximum numerical dissipation,

$$\rho_\infty^s = 0, \qquad \rho_\infty^{min} = 0, \qquad \rho_\infty^{max} = 1 \tag{106}$$

which are equivalent to $(\rho_s = 1,\ \rho = 1)$ and $(\rho_s = 0,\ \rho = 0)$ in GS4-1, respectively. Resulting in Crank-Nicolson and Gear's Method for time stepping, respectively. These were selected to highlight different damping characteristics present in different time stepping algorithms. To validate the model, it will be shown that for various time steps the wave front matches wave location predicted by the wave speed calculated in (104). Based on the material parameters provided, the calculated wave speed is $1\ \left[\frac{m}{s}\right]$. Therefore the wave front location along the beam should be equal, in nominal value, to that of the time step; the times shown are 0.10, 0.20, and 0.50 seconds.

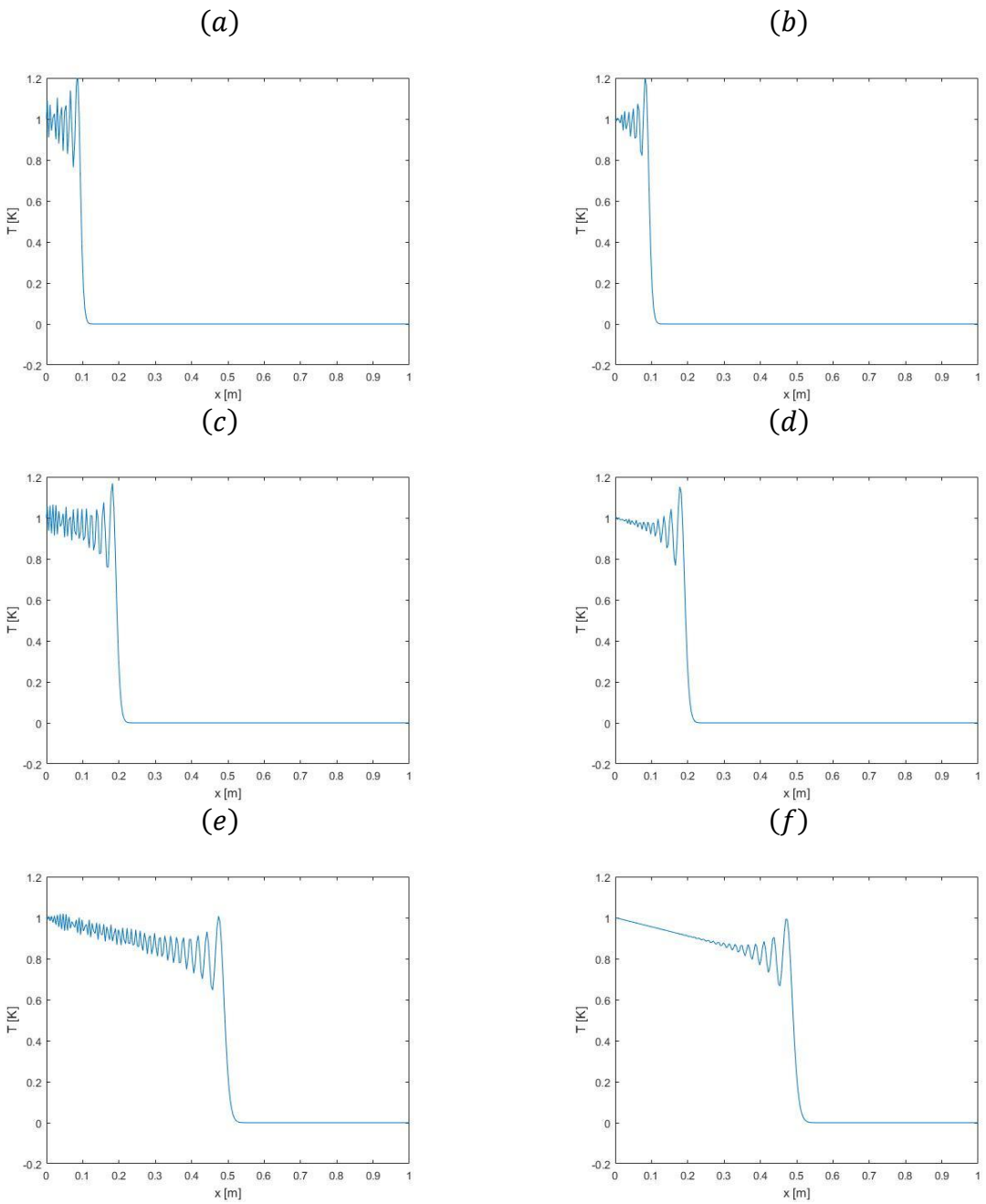


Figure 18. Cattaneo quasi-one-dimensional transient solution, at (a) & (b) 0.10 seconds, (c) & (d) 0.20 seconds, (e) & (f) 0.50 seconds, where (a),(c),(e) correspond to the undamped solution and (b),(d),(f) correspond to the damped solution

It can be seen in Figure 18 how the numerical dissipation introduced by the GS4 parameters (0,1,0); ρ_∞^{min} , ρ_∞^{max}, and ρ_∞^{s} can minimize, and almost eliminate, the numerical oscillations caused by the wave. Furthermore, the model, both with and

without numerical dissipation, can capture the wave front accurately. To further validate the numerical results, the numerical solution will be compared to the analytical solution, provided in [21], for $t = 0.5\ [s]$,

(a) (b)

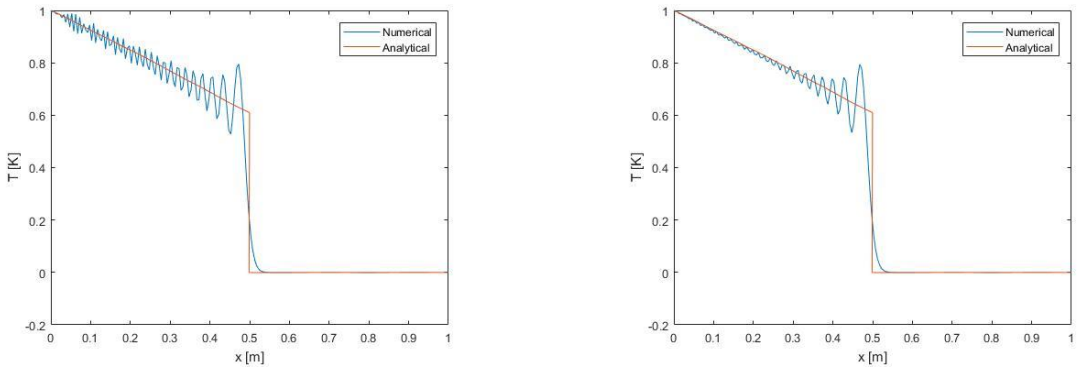


Figure 19. Comparison of numerical and analytical results for both damped and undamped solutions for t = 0.5 seconds

In the Figs. 18-19 above, it can be seen that the numerical results match the analytical results; albeit with varying degrees of oscillatory behavior attributable to the spatial convergence rate of the framework, validating the model in regard to Cattaneo-type conduction processes.

Jeffreys'-type Heat Conduction [Lagged Parabolic Heat Conduction $F_T \in (0, 1)$]

The C- and F- model degenerates to the Jeffery-type lagged parabolic heat conduction; the discontinuities existing in the Jeffreys'-type model are smoothed by the diffusion effects. It is worth noting that the acknowledged Jeffreys'-type heat conduction is also considered as a generalized model, however, it is not able to exactly recover Fourier-type heat conduction.

To illustrate the model's ability to capture lagged parabolic heat conduction, [Ft ∈ (0,1)], a transient problem will be illustrated, with its results compared to an analytical solution provided in [22]. The solutions provided for both Fourier-type and Cattaneo-type heat conduction suffice since the Jeffreys'-type is reducible to the Cattaneo-type form.

Transient Validation

To validate the implementation for Jeffreys'-type transient problems, consider a quasi-one-dimensional beam with geometry and material parameters as follows [22],

Table 7. Geometry and Material parameters for Quasi-one-dimensional Jeffreys'-type validation

Parameter	Value
Length	1 m
Width	0.002 m
Specific Heat (c_p)	1 J/kg/°K
Material Density (ρ)	1 kg/m^3
Thermal Conductivity (k)	0.5 W/m/°K
Temperature Relaxation (τ_T)	0.0, 0.001, & 0.05 s
Flux Relaxation (τ_q)	0.05 s

It was mentioned earlier that the DPL model is incorrect under certain conditions; and for correct conditions it is nothing but the Jeffreys' model which is contained within the C- and F- model. Therefore the numerical results derived from the C- and F- model will be compared to the analytical solutions of an example in [22]. To illustrate the versatility of the model results for various temperature relaxations, the combination of the temperature relaxation and the flux relaxation will be utilized to calculate the dimensionless parameter F_T, using the relation shown Chapter 4,

$$F_T = \frac{\tau_T}{\tau_q} \tag{107}$$

The values provided the corresponding dimensionless parameters are as follows,

Table 8. Dimensionless parameters based upon relaxation parameters

Temperature relaxation (τ_T) [s]	Flux relaxation (τ_q) [s]	F_T
0.0	0.05	0
0.001	0.05	0.2
0.05	0.05	1

The domain is subjected to a constant temperature boundary condition at its left boundary, with initial and boundary conditions as follows,

$$T_i = 0\ [°K], \qquad T_{BC} = 1\ [°K] \tag{108}$$

For this problem the explicit form of GS4-2 was utilized with,

$$\eta_1 = \eta_2 = \eta_3 = 1 \tag{109}$$

and time stepping parameter,

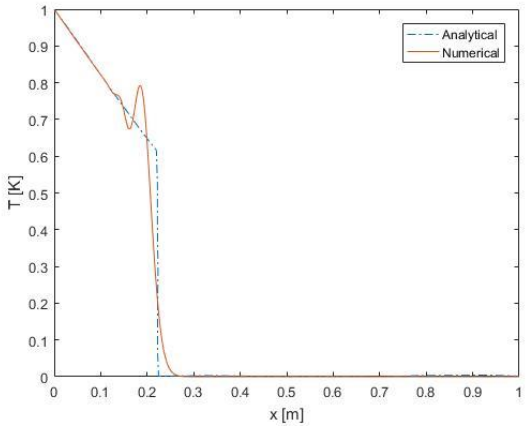
$$dt = 0.001\ [s] \tag{110}$$

This constitutive equation will result in numerical oscillations for $F_T \neq 1$. Numerical dissipation will be employed via the GS4 *i*Integration framework, with the selectable parameters being,

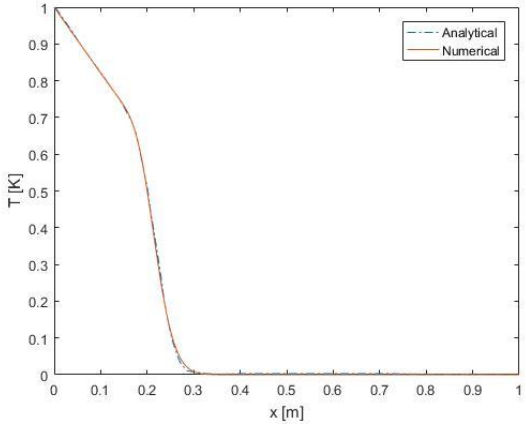
$$\rho_\infty^s = 0, \qquad \rho_\infty^{min} = 0, \qquad \rho_\infty^{max} = 1 \tag{111}$$

which is equivalent to $(\rho_s = 0,\ \rho = 0)$ in GS4-1, resulting in Gear's Method for time stepping. This was chosen due to its damping characteristics.

(a)



(b)



(c)

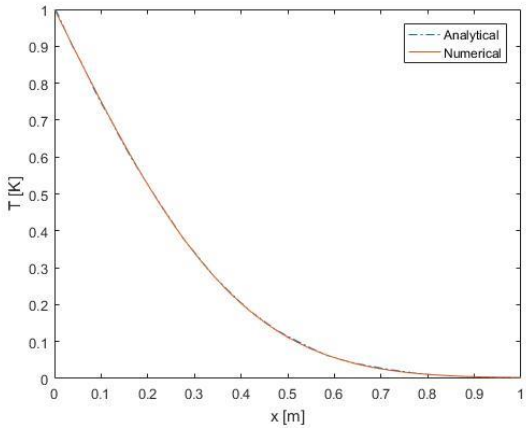


Figure 20. Results comparison for Jeffreys'-type heat conduction for (a) F_T = 0.0, (b) F_T = 0.20, and (c) F_T = 1

It can be seen in the Fig. 20 above, that there is significant agreement between the numerical results and analytical solution. For Figure 20 (a) corresponding to an $F_T = 0$; which was previously stated to be equivalent to hyperbolic heat conduction, the wave front is present with its corresponding oscillations which is characteristic of the hyperbolic form. For Figure 20 (b) corresponding to an $F_T = 0.2$ which is in between purely hyperbolic and purely parabolic heat conduction, there is still the semblance of a wave front, but it is significantly smoother than the purely hyperbolic result. Lastly for Figure 20 (c) corresponding to an $F_T = 1$; which is equivalent to parabolic heat conduction, we can see the wave like nature no longer exists and we have a smooth temperature distribution along the length of the computational domain, characteristic of Fourier-type processes. Therefore, we can claim that the C- & F- model can accurately recover Jeffreys'-type heat conduction.

Additional Examples

To further illustrate the versatility of the model additional numerical examples for parabolic, hyperbolic, and lagged parabolic heat conduction will be presented. These examples will utilize geometries and material properties that vary from the validation examples shown previously.

Fourier-type Process Examples

The first additional example will be a steel circular disc with constant radial temperature. The disc has material and geometry properties as follows,

Table 9. Geometry and Material properties for parabolic heat conduction in a circular disc with constant radial temperature

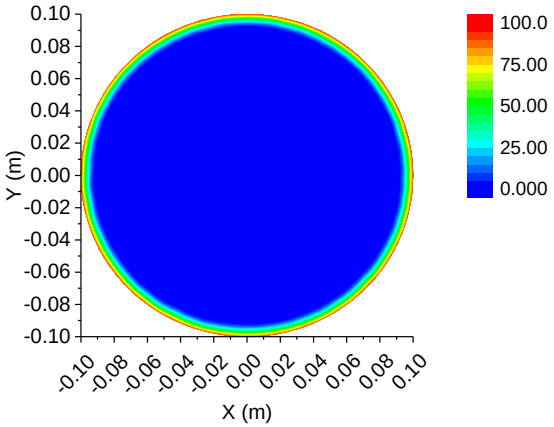
Parameter	Value
Radius	0.1 m
Specific Heat	486 J/kg/°K
Material Density	7870 kg/m^3
Thermal Conductivity	51.9 W/m/°K

It should be noted that the material properties for this example were taken from Ref [20]. The initial condition and boundary condition can be seen below,

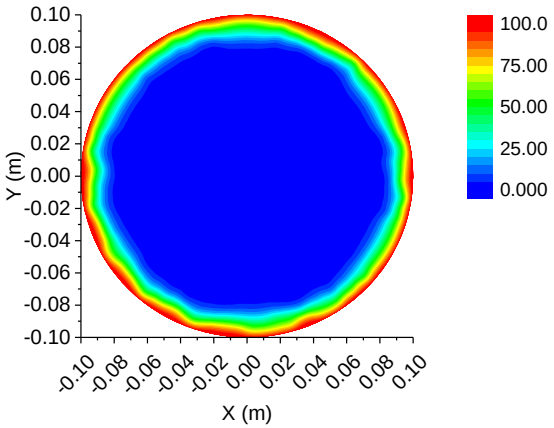
$$T_i = 0\ [K], \qquad T_{BC} = 100\ [K] \tag{112}$$

To illustrate the results from this simulation, temperature contour plots will be shown at various time steps as in Fig. 21 below,

(a)



(b)



(c)

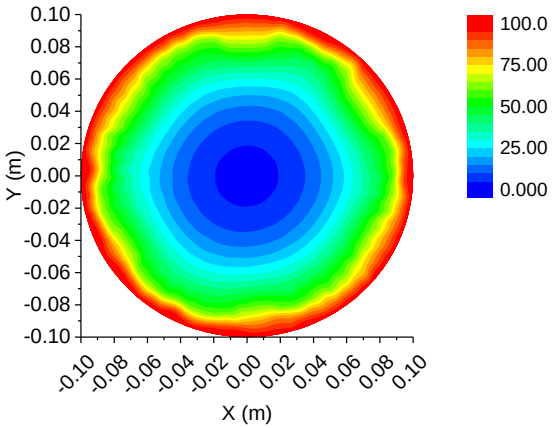


Figure 21. Temperature contour plots for circular disc example at 0.5, 5.0, and 50..0 seconds; (a), (b), and (c) respectively

It can be seen that there are isothermal lines radially moving inward from the boundary towards the disc's center, as is to be expected from the simulation. The irregularities in the contours are due to nonuniformity in the initial particle discretization. For this problem, a code designed for meshing developed in [23] was utilized for the initial particle discretization. To further illustrate the heat propagation, time histories for particles at various radial distances can be seen in Fig. 22 below,

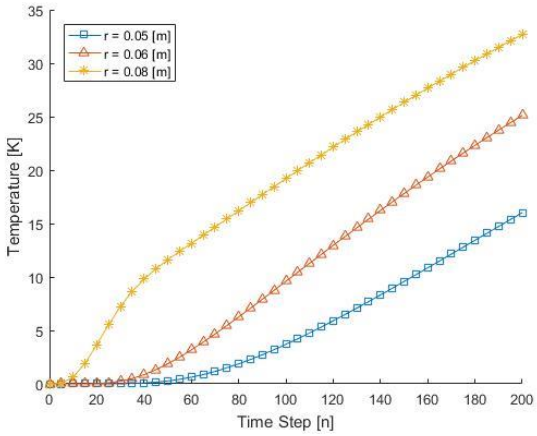


Figure 22. Temperature time histories for particles at, approximately, r = 0.05, 0.06, and 0.08 (m) from the center of the disc

As another two-dimensional example, a square domain with two sets of specified boundary conditions will be simulated, as presented in [24]. It is a two-dimensional square domain with side lengths with parameters as shown below,

Table 10. Geometry and material parameters for 2D square transient Fourier example

Parameter	Value
Length	1 m
Width	1 m
Specific Heat	1 J/kg/°K
Material Density	1 kg/m^3
Thermal Conductivity	1 W/m/°K

The boundary conditions for the two cases are shown below,

$$Case\ A: T(x,0) = T(x,L) = T(0,y) = T(L,y) = 1\ °C \tag{113}$$

$$Case\ B: T(x,0) = T(x,L) = T(0,y) = 1\ °C, \frac{\partial T}{\partial y} = 0 \tag{114}$$

With the following time stepping parameters,

$$t_i = 0\ [sec], \qquad dt = 10^{-3}\ [sec], \qquad n_{steps} = 100 \tag{115}$$

An illustration of each boundary condition case can be seen in the Fig. 23 below,

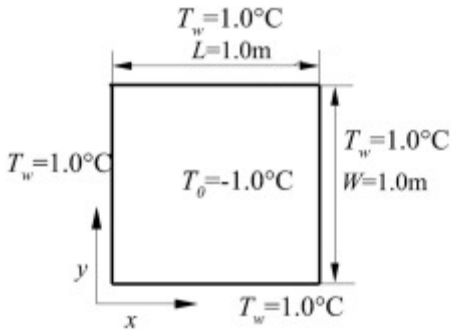


Figure 23. Two-dimensional transient heat conduction numerical example, Case A

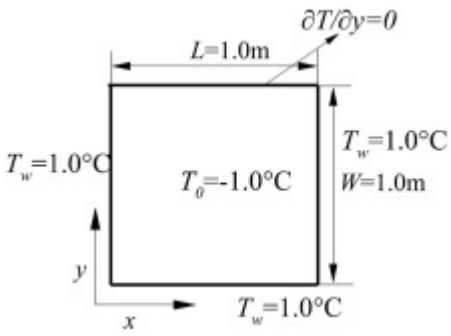


Figure 24. Two-dimensional transient heat conduction numerical example, Case B

The temperature distribution at various time steps for both cases can be seen in Fig. 25 below,

(a) (b)

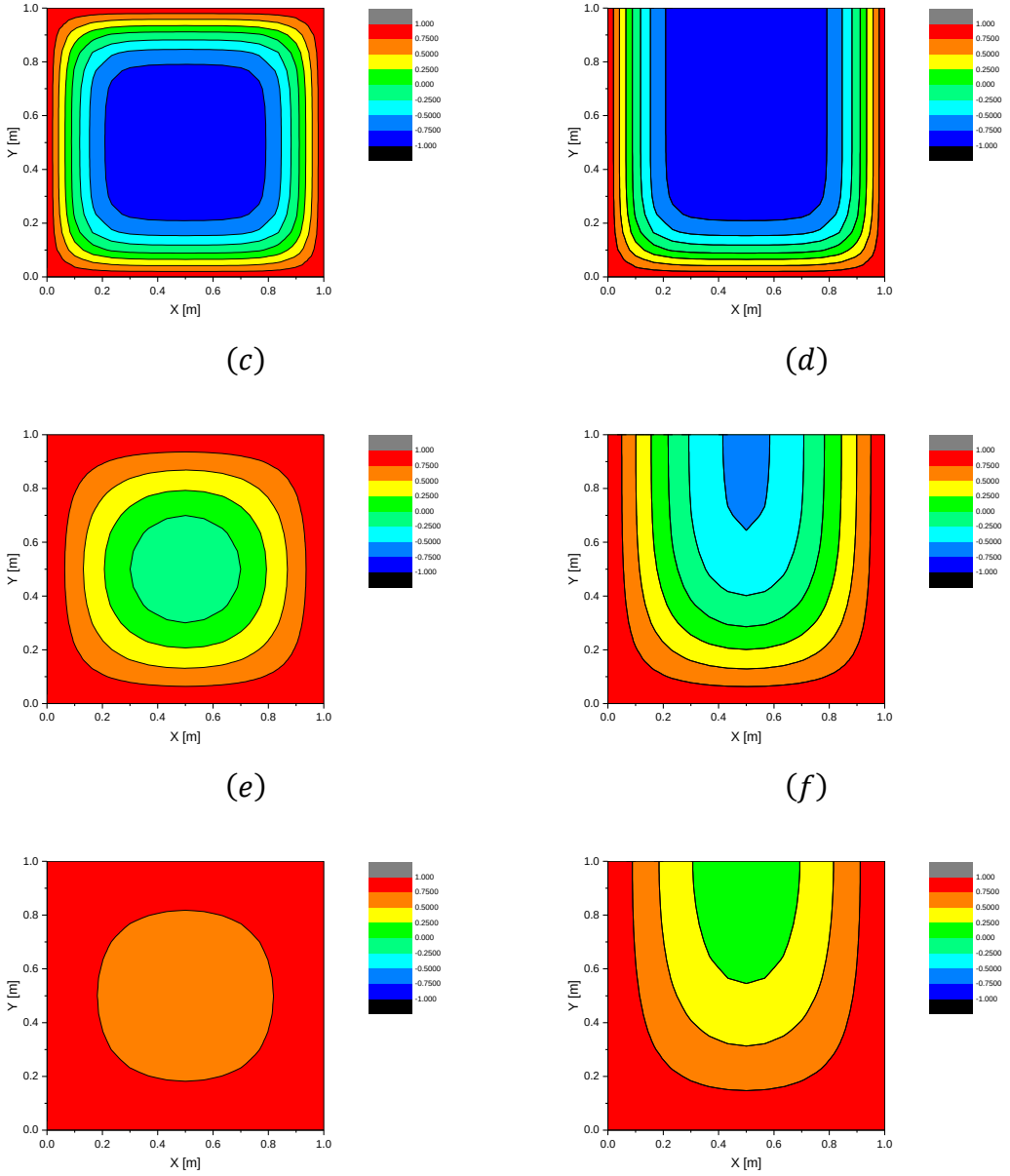


Figure 25. Case A temperature distribution plots; (a), (c), (e), at 0.01, 0.05, and 0.1 seconds respectively, Case B temperature distribution plots;(b), (d), (f), at 0.01, 0.05, and 0.1 seconds respectively

It can be seen in the temperature distribution plots, for each of the boundary case examples that the physics are captured correctly. Furthermore, from the numerical validation shown previously we can also claim, that since the operators utilized have been shown to work previously that these numerical results are also quantitively correct.

Cattaneo-type Process Examples

A circular steel disc with constant radial temperature has material and geometry properties as follows,

Table 11. Geometry and Material properties for hyperbolic heat conduction in a circular disc with constant radial temperature

Parameter	Value
Radius	0.1 m
Specific Heat	486 J/kg/°K
Material Density	7870 kg/m^3
Thermal Conductivity	51.9 W/m/°K
Relaxation Parameter (τ)	5 s

It should be noted that the material properties for this example were taken from [20], except for the relaxation parameter which was chosen arbitrarily. The initial condition and boundary condition can be seen below,

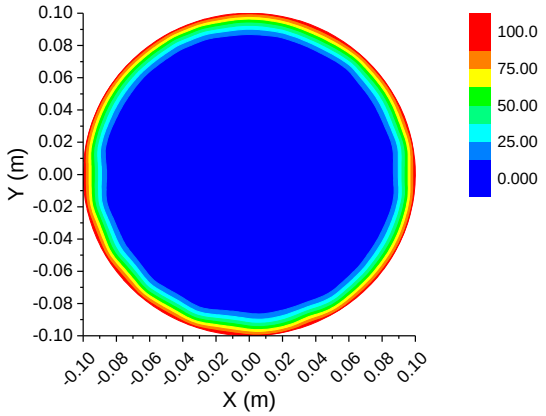
$$T_i = 0\ [K], \qquad T_{BC} = 100\ [K] \tag{116}$$

Because it leads to numerical oscillations in this model, numerical dampening will be employed via the GS4 *i*Integration framework, with the selectable parameters being,

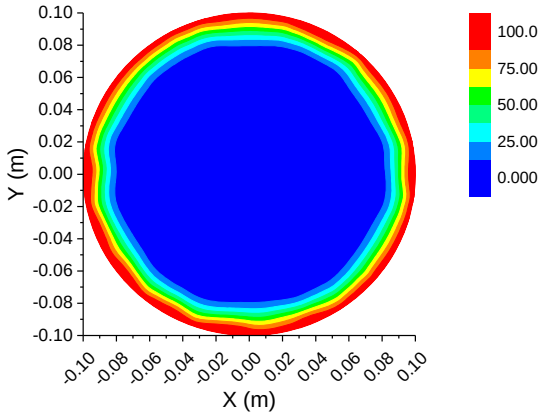
$$\rho_\infty^s = 0, \qquad \rho_\infty^{min} = 0, \qquad \rho_\infty^{max} = 1 \tag{117}$$

To illustrate the results from this simulation, temperature contour plots will be shown at various time steps as in Fig. 26 below,

(a)



(b)



(c)

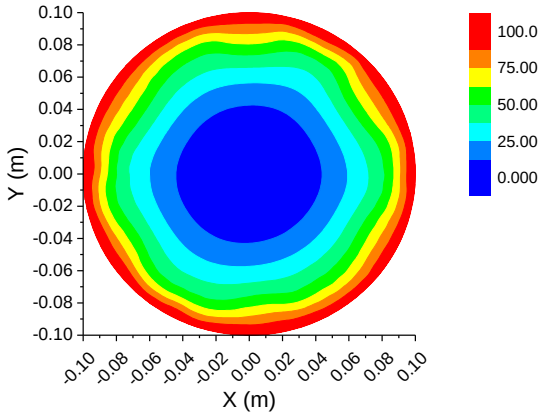


Figure 26. Temperature contour plots for hyperbolic heat conduction in steel disc at (a) t = 5 seconds, (b) t = 10 seconds, and (c) t = 50 seconds

We still observe the isothermal lines propagating inward as the time increases. In the later time steps it can be seen that there is a larger section at the center that is yet to be affected, attributable to the wave propagation not yet reaching the center of the disc.

A second additional example is an annular disc with a specified outer radius temperature boundary condition. The disc has material and geometry properties as follows,

Table 12. Geometry and Material properties for hyperbolic heat conduction in an annular disc with constant temperature boundary condition

Parameter	Value
Inner Radius	0.05 m
Outer Radius	0.1 m
Specific Heat	486 J/kg/°K
Material Density	7870 kg/m^3
Thermal Conductivity	51.9 W/m/°K
Relaxation Parameter (τ)	1 s

It should be noted that the material properties for this example were taken from [20]. The initial condition and boundary condition for this problem can be seen below,

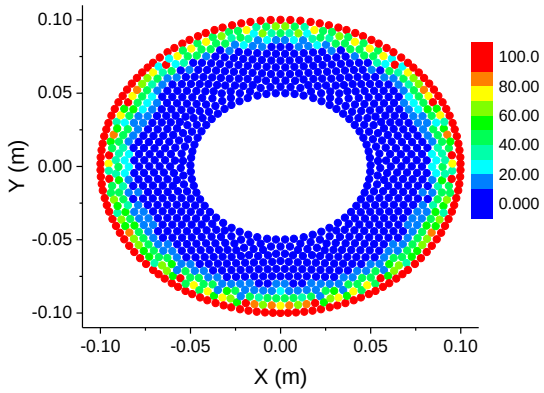
$$T_i = 0\ [K], \qquad T_{BC} = 100\ [K] \tag{118}$$

Because it was shown that hyperbolic heat conduction leads to numerical oscillations in this model, numerical dampening will be employed via the GS4 *i*Integration framework, with the selectable parameters being,

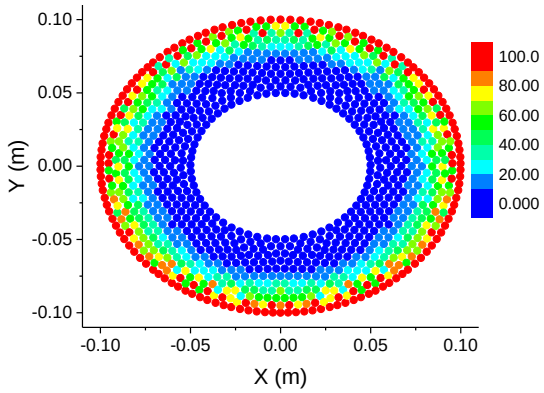
$$\rho_\infty^s = 0, \qquad \rho_\infty^{min} = 0, \qquad \rho_\infty^{max} = 1 \tag{119}$$

To illustrate the results from this simulation, temperature contour plots will be shown at various time steps as in Fig. 27 below,

(a)



(b)



(c)

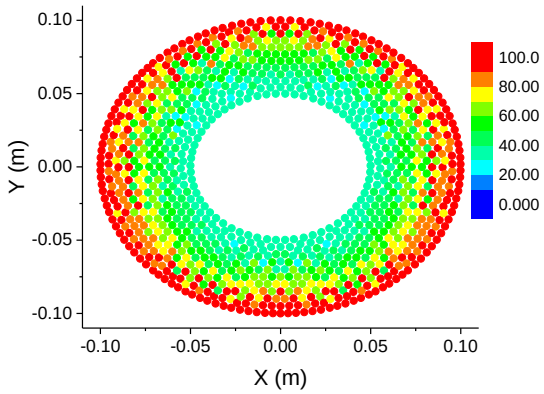


Figure 27. Temperature contour plots for hyperbolic heat conduction in annular steel disc at (a) t = 5 seconds, (b) t = 10 seconds, and (c) t = 50 seconds

Due to the discretized nature of the plots above, the numerical oscillations corresponding with hyperbolic heat conduction are more apparent. When comparing Figure 27 (a) and Figure 27 (b), the wave propagating from the outer radius inward is apparent in the sharp discontinuity between the temperature upstream of the wave and the temperature downstream of the wave.

Jeffreys'-type Process Examples

The first additional example for will be the same the one presented for both Fourier-type and Cattaneo-type processes. A circular steel disc with constant radial temperature has material and geometry properties as follows,

Table 13. Geometry and Material properties for lagged parabolic conduction in a circular disc with constant radial temperature

Parameter	Value
Inner Radius	0.05 m
Outer Radius	0.1 m
Specific Heat	486 J/kg/°K
Material Density	7870 kg/m^3
Thermal Conductivity	51.9 W/m/°K
Temperature Relaxation (τ_T)	0.005s
F_T	0.2

It should be noted that the material properties for this example were taken from [20], except for the relaxation parameters; the flux relaxation was chosen arbitrarily and the temperature relaxation was chosen such that $F_T = 0.2$ via (107). The initial condition and boundary condition can be seen below,

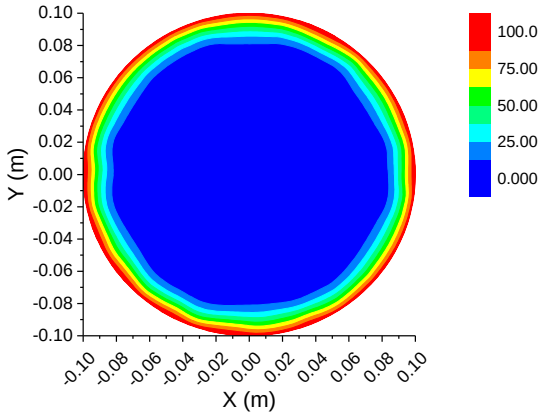
$$T_i = 0\ [K], \qquad T_{BC} = 100\ [K] \tag{120}$$

Because it was shown that lagged parabolic conduction leads to numerical oscillations in this model, numerical dampening will be employed via the GS4 *i*Integration framework, with the selectable parameters being,

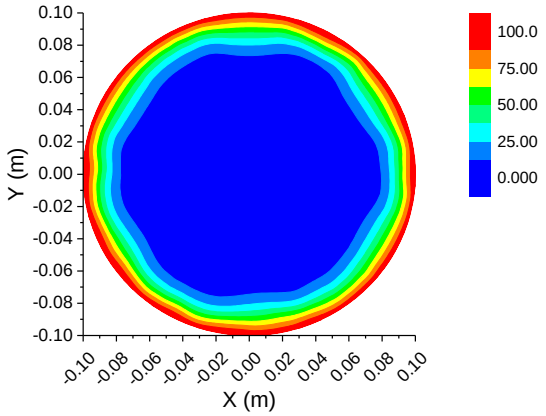
$$\rho_\infty^s = 0, \qquad \rho_\infty^{min} = 0, \qquad \rho_\infty^{max} = 1 \tag{121}$$

To illustrate the results from this simulation, temperature contour plots will be shown at various time steps as in Fig. 28 below,

(a)



(b)



(c)

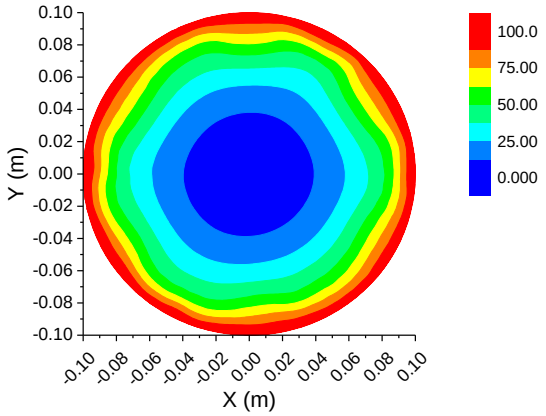


Figure 28. Temperature contour plots for lagged parabolic conduction in steel disc at (a) t = 5 seconds, (b) t = 10 seconds, and (c) t = 50 seconds

Similar to when this problem was solved using a Cattaneo-type process, we still observe the isothermal lines propagating inward as the time increase. However when comparing with the previously mentioned example, in the later time steps it can be seen that there is a smaller section at the center that is yet to be affected, attributable to the smoothed wave propagation characteristic of this type of lagged parabolic conduction.

A second additional example is another two-dimensional example of an annular disc with a specified outer radius temperature boundary condition. The disc has material and geometry properties as follows,

Table 14. Geometry and Material properties for lagged parabolic conduction in an annular disc with constant temperature boundary condition

Parameter	Value
Inner Radius	0.05 m
Outer Radius	0.1 m
Specific Heat	486 J/kg/°K
Material Density	7870 kg/m^3
Thermal Conductivity	51.9 W/m/°K
Temperature Relaxation (τ_T)	0.005s
Flux Relaxation (τ_q)	0.1 s

It should be noted that the material properties for this example were taken from [20], except for the relaxation parameters; the flux relaxation was chosen arbitrarily and the temperature relaxation was chosen such that $F_T = 0.2$ via (107). The initial condition and boundary condition for this problem can be seen below,

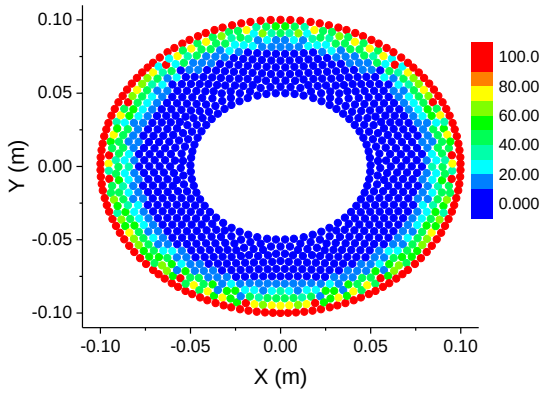
$$T_i = 0\ [K], \qquad T_{BC} = 100\ [K] \tag{122}$$

Because it was shown that hyperbolic heat conduction leads to numerical oscillations in this model, numerical dampening will be employed via the GS4 *i*Integration framework, with the selectable parameters being,

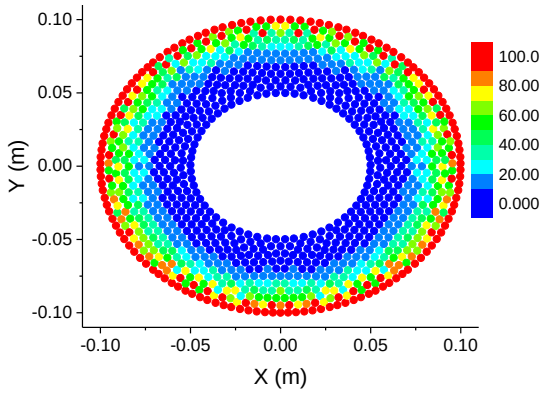
$$\rho_\infty^s = 0, \qquad \rho_\infty^{min} = 0, \qquad \rho_\infty^{max} = 1 \tag{123}$$

To illustrate the results from this simulation, temperature contour plots will be shown at various time steps as in Fig. 29 below,

(a)



(b)



(c)

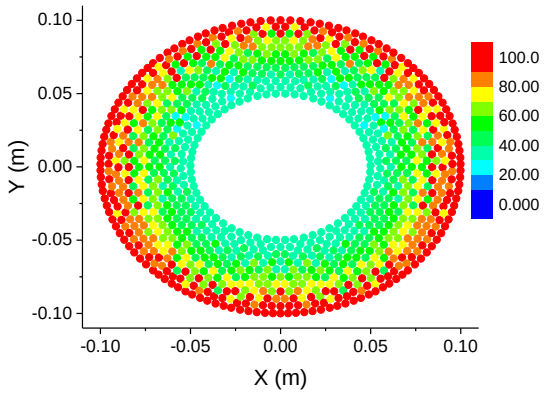


Figure 29. Temperature contour plots for lagged parabolic heat conduction ($F_T = 0.2$) in annular steel disc at (a) t = 5 seconds, (b) t = 10 seconds, and (c) t = 50 seconds

Once again, due to the discretized nature of the plots above, the numerical oscillations corresponding with lagged parabolic heat conduction are more apparent. Furthermore, when comparing Figure 29 (a) and Figure 29 (b), the wave propagating from the outer radius inward is still apparent, as in the hyperbolic example. However the discontinuity of the wave is smoother and less sharp between the temperature upstream of the wave and the temperature downstream of the wave.

Conclusions

In this chapter the numerical implementation of the C- & F- model with respect to Fourier-type parabolic, Cattaneo-type hyperbolic, and Jeffreys'-type lagged parabolic heat conduction without source terms was presented. It was also verified against both analytical solutions and previously published numerical results for the various conduction types, corroborating the operators and methods used in this framework. Additional examples for each of the conduction types were also provided to illustrate the flexibility and versatility of the model.

Furthermore, for the hyperbolic and lagged parabolic heat conduction problems it was shown that there exists significant numerical oscillations attributable to the wave like nature of the problems. It was also seen that due to the GS4 *i*Integration time integration framework, these oscillations could be dissipated although they could not be eliminated entirely without additional operations being applied to the data.

Chapter 6: MPS Framework with Implementation of C- and F- Model with Stationary Source

Introduction

This chapter will cover the basic theory, implementation, and numerical validation, as well as provide additional numerical examples, for the C- and F- model with regards to problems involving internal heat generation or a stationary external source in the proposed framework. This formulation of the C- and F- model will be validated against analytical and published results for Fourier-type and Cattaneo-type heat conduction with internal heat generation and source terms in both the steady and transient states. It is claimed that if the addition of the source term is validated with regards to Fourier-type and Cattaneo-type processes, in unison to the validation provided in Chapter 5, then by extension the addition of source term would be valid for Jeffreys'-type heat conduction as

well. Additional examples will be shown as well to further illustrate the method's capability. By extending the base formulation, discussed in the Chapter 5, with the addition of heat generation or a stationary external source, the range of practical applications that can be analyzed with this method is broadened.

Theory

Numerical heat conduction utilizing the C- and F- model with the presence of stationary internal heat generation or an external heat source can be described with constitutive equations (78) -(80) and the governing equation (81). As noted previously, this type of problem can be written in a two-field form as seen in equations (82)-(87). In the cases encompassed by this chapter, the source matrix (***S***) in equation (85) is now a non-zero vector representing the external heat source, or internal heat generation/sink. For the transient examples, the GS4 *i*Integration framework for first order systems (GS4-1) as shown in Chapter 3, will be implemented for time integration.

Implementation

The algorithm for the problems in this chapter is described in Figure 30. In contrast to the algorithm presented in Chapter 5 and as mentioned in the previous, a source term is added due to the inclusion of heat generation and external source terms in this chapter, specifically in equation (85).

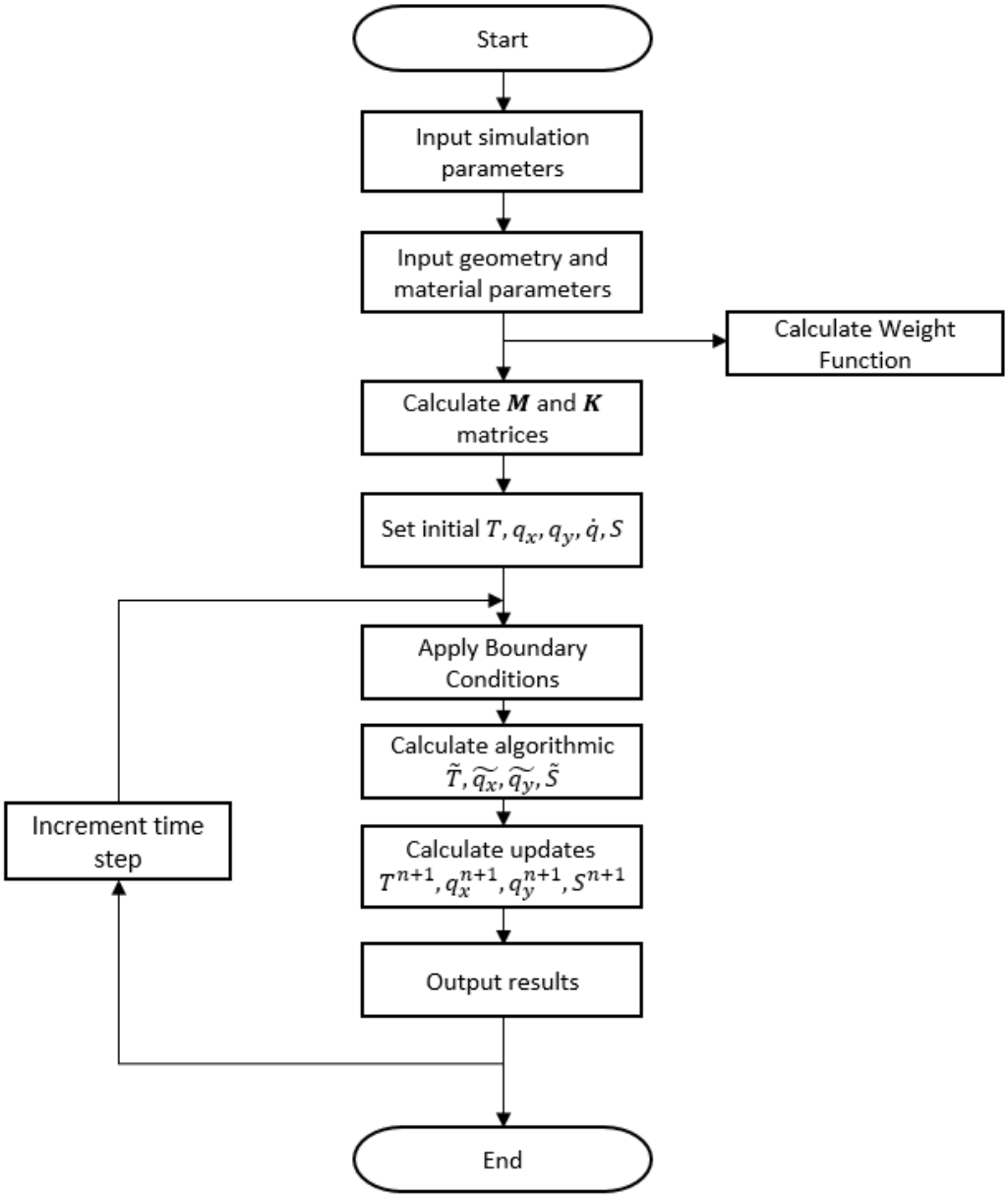


Figure 30. Computational algorithm of proposed framework encompassing Chapter 6 examples

Numerical Validation

Various numerical examples are utilized to validate that the temperature distributions resulting from the model are derived and implemented correctly for both Fourier type processes and Cattaneo type processes. For Fourier type processes, the model will be validated in both the steady state and transient state. For Cattaneo type processes, the model will be validated in the transient state.

Fourier Heat Conduction (Parabolic Heat Conduction [F_T = 1])

To validate the model with regard to Fourier-type heat conduction with heat generation, or a stationary external heat source, problems will be presented for both the steady and transient states in which numerical results will be compared against analytical solutions as feasible. For the steady state problem, a one-dimensional bar with internal heat generation will be simulated and its results will be compared against the analytical solution presented. For the transient case, a time dependent external source problem will be presented in which a one-dimensional bar will experience an instantaneous heat source at one end of the bar. Numerical results for this problem will then be compared against a provided analytical solution to the problem.

Steady State Validation

Consider in time-independent, one-dimensional space with the presence of a source term, under these assumptions the governing equation reduces to,

$$k\frac{d^2T}{dx^2} + \dot{q} = 0 \tag{124}$$

When solved using separation of variables the steady state solution becomes,

$$T(x) = -\frac{\dot{q}x^2}{2k} + Ax + B \tag{125}$$

Where A and B are constants dependent upon the specified boundary conditions for the problem. For simplicity consider a one-dimensional bar of length L, subject to constant temperature boundary conditions at each end. Under these considerations, the constants in the analytical solution become,

$$B = T_0, \qquad A = \frac{\frac{\dot{q}L^2}{2k} - T_0 + T_L}{L} \tag{126}$$

where T_0 and T_L are the boundary conditions at the beginning and end of the beam respectively. The given boundary conditions are,

$$T_0 = T(x = 0) = 1, \qquad T_L = T(x = L) = 0 \tag{127}$$

The problem parameters are as given in the table below,

Table 15. Problem parameters for steady state heat generation validation

Parameter	Value
Length	1
Thermal Conductivity	1
Internal Heat Generation	1

and the corresponding analytical solution becomes,

$$T(x) = -\frac{x^2}{2} - 0.5x + 1 \tag{128}$$

This will be used as the basis of validation for the numerical simulation. For this simulation, a quasi-one-dimensional slender bar geometry with given geometry and material properties as seen in the table above were utilized.

A plot of the numerical solution and the analytical solution can be seen below, in which it can be seen that there is excellent agreement between the analytical solution and the numerical results from the C- and F- model.

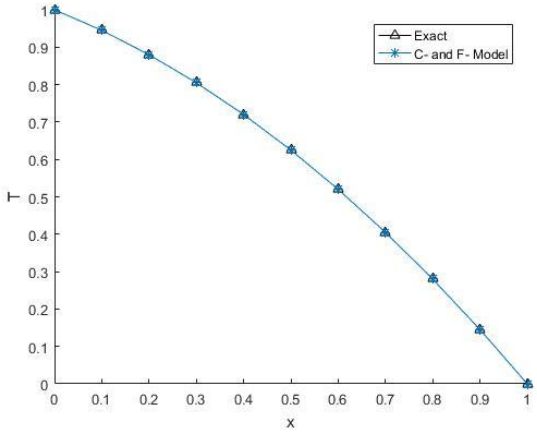


Figure 31. Plot of numerical results versus analytical results for 1D steady state heat generation problem

To once again illustrate the spatial convergence of the model a plot of the average percent error for the numerical results against the level of spatial discretization can be seen below. Unlike the two-dimensional problem utilized in Chapter 5, the problem presented here in Chapter 6 is one-dimensional allowing for greater levels of discretization without causing undue strain on the computational resources.

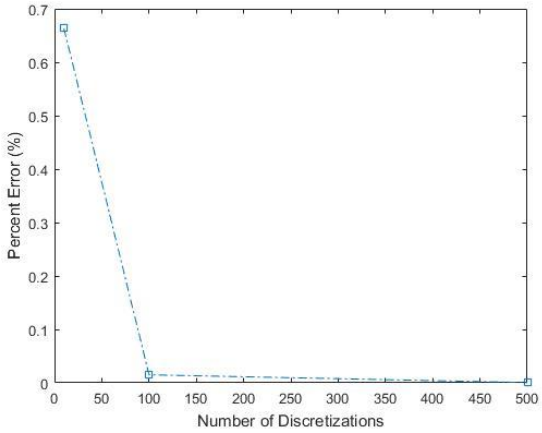


Figure 32. Steady state heat generation average percent error for various spatial discretization

Transient Validation

Consider a three-dimensional semi-infinite solid that is governed by the three-dimensional diffusivity equation,

$$\frac{1}{\alpha}\frac{\partial T}{\partial t} = \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2} + Q \tag{129}$$

initially at $T(x, y, z, 0) = 0$. However if we consider heat released instantaneously over an entire plane; whether it be the $xy - plane, xz - plane,$ or $yz - plane$, instead of at a single point, then the problem reduces spatially to a one-dimensional problem; this is because we need to only consider the direction orthogonal to planar heat source. Therefore, consider a semi-infinite beam, initially at $T(x, 0) = 0$, that has a specified amount of energy $Q\rho c_p$ released at its $x = 0$ at time $t = 0$, then the corresponding analytical solution is,

$$T(x, t) = \frac{Q}{\sqrt{4\pi\alpha t}} e^{-\frac{x^2}{4\alpha t}} \tag{130}$$

It should be noted that in this instance $Q\rho c_p$ represents the heat released per unit area. The parameters for this problem are as seen in the table below,

Table 16. Geometry and material parameters for Fourier quasi-one-dimensional transient simulation

Parameter	Value
Length	0.5
Thermal Conductivity (k)	1
Specific Heat (c_p)	1
Material Density (ρ)	1
Heat Source (Q)	1

A quasi-one-dimensional domain will be utilized to simulate this problem. For this problem the explicit form of GS4-2 was utilized with,

$$\eta_1 = \eta_2 = \eta_3 = 1 \tag{131}$$

with the time stepping parameter,

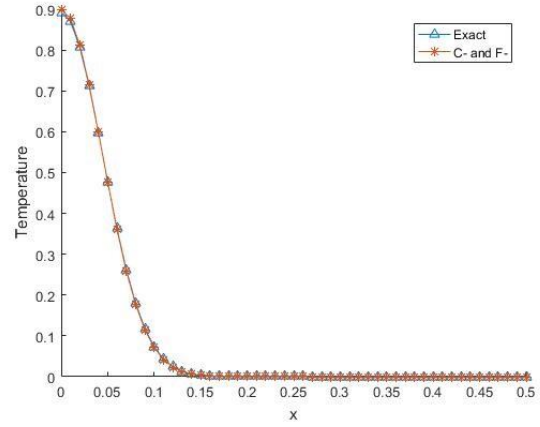
$$dt = 10^{-5}\,[s] \tag{132}$$

and,

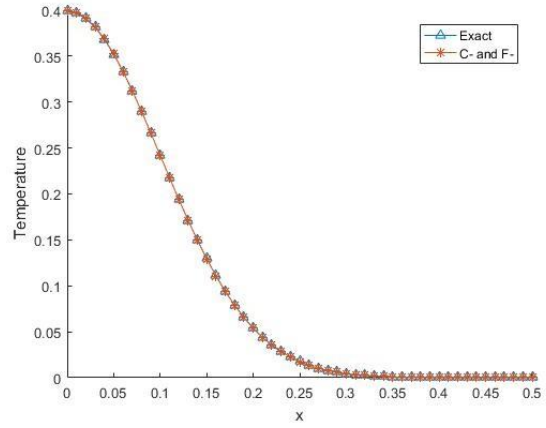
$$\rho_s = 0, \qquad \rho_{min} = 0, \qquad \rho_{max} = 1 \tag{133}$$

which is equivalent to $(\rho_s = 0,\ \rho = 0)$ in GS4-1, resulting in Gear's Method for time stepping. This was chosen due to its damping characteristics. Results will be compared against the exact solution; (130). Temperature versus position for three different time steps; at 0.001, 0.005, and 0.01 seconds can be seen below in Fig. 33,

(a)



(b)



(c)

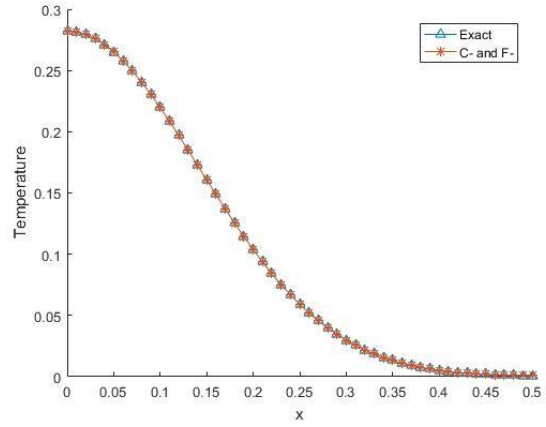


Figure 33. Results for 1D instantaneous source at (a) 0.001, (b) 0.005, and (c) 0.01 seconds, respectively

It can be seen in the Fig. 33 above that there is significant agreement between the numerical results and the exact solution. The average absolute value percent error for the duration of the simulation can be seen below,

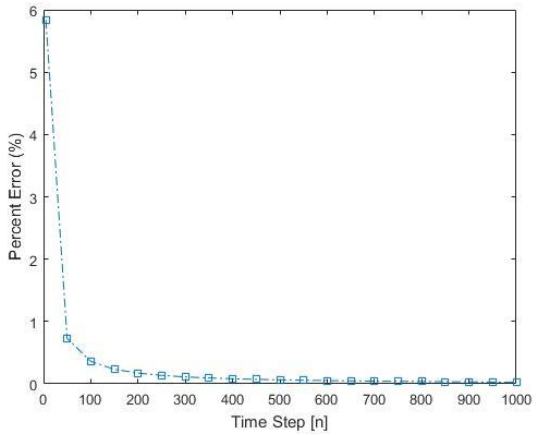


Figure 34. Average percent error for each time step for instantaneous heat source numerical example

After the first set of initial time steps that the average percent error drastically decreases, corroborating the claim that there is significant agreement between the numerical results and the analytical solution.

Cattaneo-type Heat Conduction (Hyperbolic Heat Conduction [$F_T = 0$])

To validate the model with regard to Cattaneo-type hyperbolic heat conduction with the inclusion of heat generation, or a stationary external heat source, a problem will be presented in the transient state from which numerical results will be compared against previously published results.

Transient Validation

It has been shown in [25] and [26] that a heat flux boundary condition is equivalent to an external source acting on a boundary. This is achieved by equating the heat flux at the boundary to a source term of equivalent energy input. Specifically for a constant cross-

section part one can take the heat flux ($\boldsymbol{q}$) and the spatial discretization (Δ) and compute the equivalent volumetric heat generation ($\tilde{Q}$),

$$\tilde{Q} = \frac{\dot{Q}}{V_f} = -\frac{\int_{S_f} \boldsymbol{q}\cdot\boldsymbol{n} dS}{V_f} = -\frac{(\boldsymbol{q}\cdot\boldsymbol{n})S_f}{S_f\Delta} = -\frac{(\boldsymbol{q}\cdot\boldsymbol{n})}{dx} \tag{134}$$

Using the relation above we can validate the C- and F- model with the inclusion of a source term with regard to Cattaneo type hyperbolic heat conduction by reproducing the results of a problem utilizing a heat flux boundary condition.

Therefore consider the one dimensional beam problem, of constant cross section, from [27] with the follow geometry and material properties,

Table 17. Geometry and material parameters for Fourier quasi-one-dimensional transient simulation

Parameter	Value
Length	1
Thermal Conductivity (k)	1
Specific Heat (c_p)	1
Material Density (ρ)	1
Relaxation Parameter (τ)	1

With the applied heat flux boundary condition,

$$q_{BC} = 1$$

This is equivalent to the same problem utilizing a source term applied at the boundary, specifically,

$$\tilde{q} = \frac{q_{BC}}{dx} \tag{135}$$

For this problem the explicit form of GS4-2 was utilized with,

$$\eta_1 = \eta_2 = \eta_3 = 1 \tag{136}$$

with the time stepping parameter,

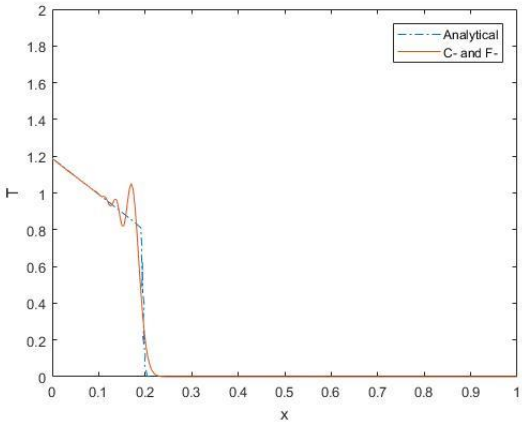
$$dt = 10^{-5}\,[s] \tag{137}$$

and,

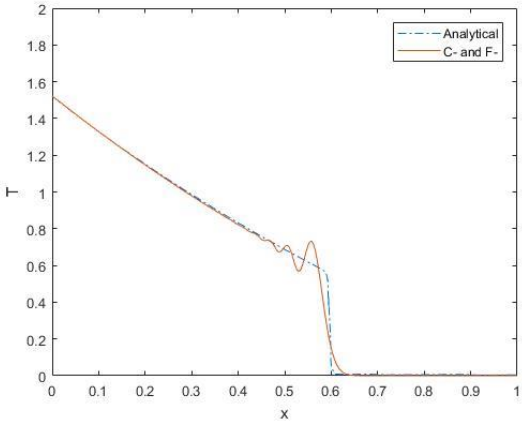
$$\rho_s = 0, \qquad \rho_{min} = 0, \qquad \rho_{max} = 1 \tag{138}$$

which is equivalent to $(\rho_s = 0,\ \rho = 0)$ in GS4-1, resulting in Gear's Method for time stepping. This was chosen due to its damping characteristics. The published results from this paper will be utilized as the basis of validation. The results from the simulation, in comparison to the published results, can be seen below,

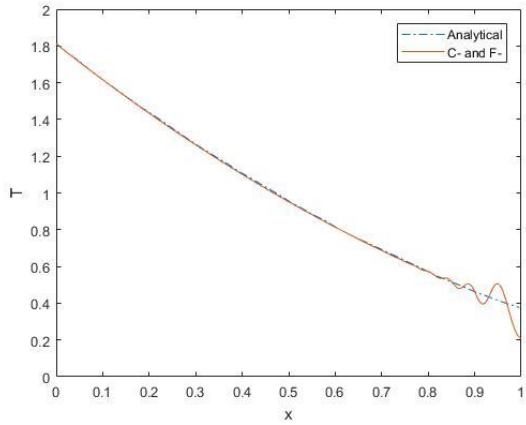
(a)



(b)



(c)



.

Figure 35. Results for 1D instantaneous source at (a) 0.02, (b) 0.06, and (c) 1.00 seconds, respectively

It can be seen in the figures above that there is significant agreement between the previously published results, albeit with the numerical oscillations similar to what was seen with the transient hyperbolic heat conduction results in Chapter 5. Further validating this models ability to simulate hyperbolic heat conduction with the presence of heat generation or external source terms.

Additional Examples

To further illustrate the versatility of the model, additional numerical examples for Fourier-type parabolic, Cattaneo-type hyperbolic, and Jeffreys'-type lagged parabolic heat conduction will be presented. These examples will utilize geometries and material properties that vary from the validation examples shown previously.

Fourier-type Process Examples

The first additional example presented is a steel circular disc with an external source continuously acting on the center of the disc and a constant temperature radius boundary condition. The material and geometry properties are as follows,

Table 18. Geometry and material properties for Fourier-type additional example

Parameter	Value
Radius	0.1 m
Specific Heat	486 J/kg/°K
Material Density	7870 kg/m^3
Thermal Conductivity	51.9 W/m/°K

To approximate a point source a Gaussian distribution will be utilized, with the maximum energy level being,

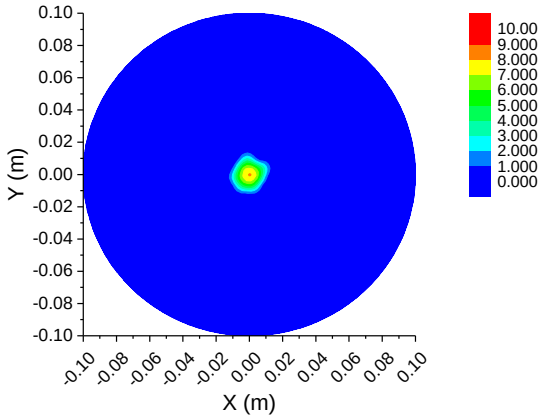
$$Q_{max} = 10\ [W], \qquad r_{source} = 0.01\ [m] \tag{139}$$

The initial condition and boundary condition can be seen below,

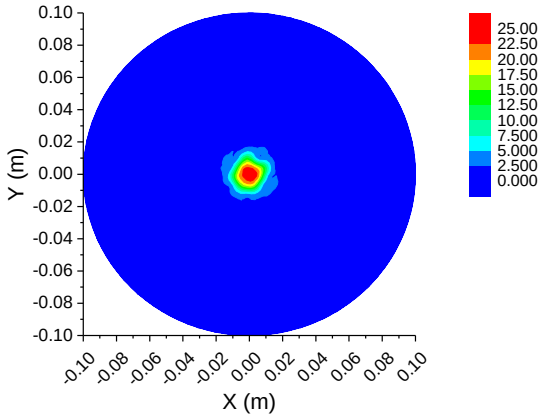
$$T_i = 0\ [K], \qquad T_{BC} = 0\ [K] \tag{140}$$

The results of this simulation can be seen in Fig. 35 below,

(a)



(b)



(c)

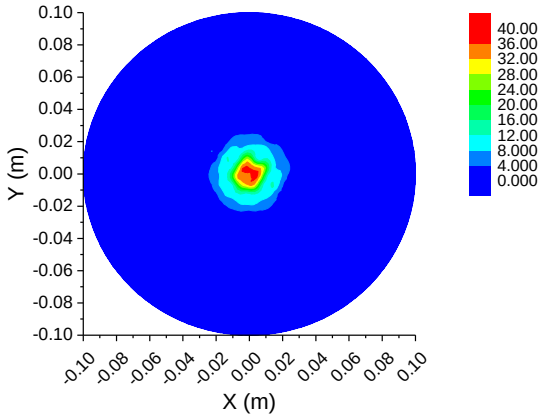


Figure 36. Temperature contour plots for Fourier-type heat conduction example with continuous external source at (a) t = 1 second, (b) t = 5 seconds, (c) t = 15 seconds

Notice in the plots above that as time progresses the temperature at the center of the disc is increasing, as energy is being continually input by the external source. Furthermore, the temperature is slowly propagating outward in concentric circles, as is expected based upon the physics governing the problem. Also, it should be noted that smoothing was utilized in the plotting software, this was simply to create a better illustration of the results.

The second additional example presented is a steel square with a time dependent external source acting within the lower left hand quadrant and a constant temperature edge boundary condition. The material and geometry properties are as follows,

Table 19. Geometry and material properties for Fourier-type additional example

Parameter	Value
Length	0.2 m
Width	0.2 m
Specific Heat	486 J/kg/°K
Material Density	7870 kg/m^3
Thermal Conductivity	51.9 W/m/°K

To approximate a point source a Gaussian distribution will be utilized, with the maximum energy level being,

$$Q_{max} = 10\ [W], \qquad r_{source} = 0.02\ [m] \tag{141}$$

This external source will be applied for the first 5 seconds, after that the source will be turned off. The initial condition and boundary condition can be seen below,

$$T_i = 0\ [K], \qquad T_{BC} = 0\ [K] \tag{142}$$

The results from this simulation can be seen below in Figure 37.

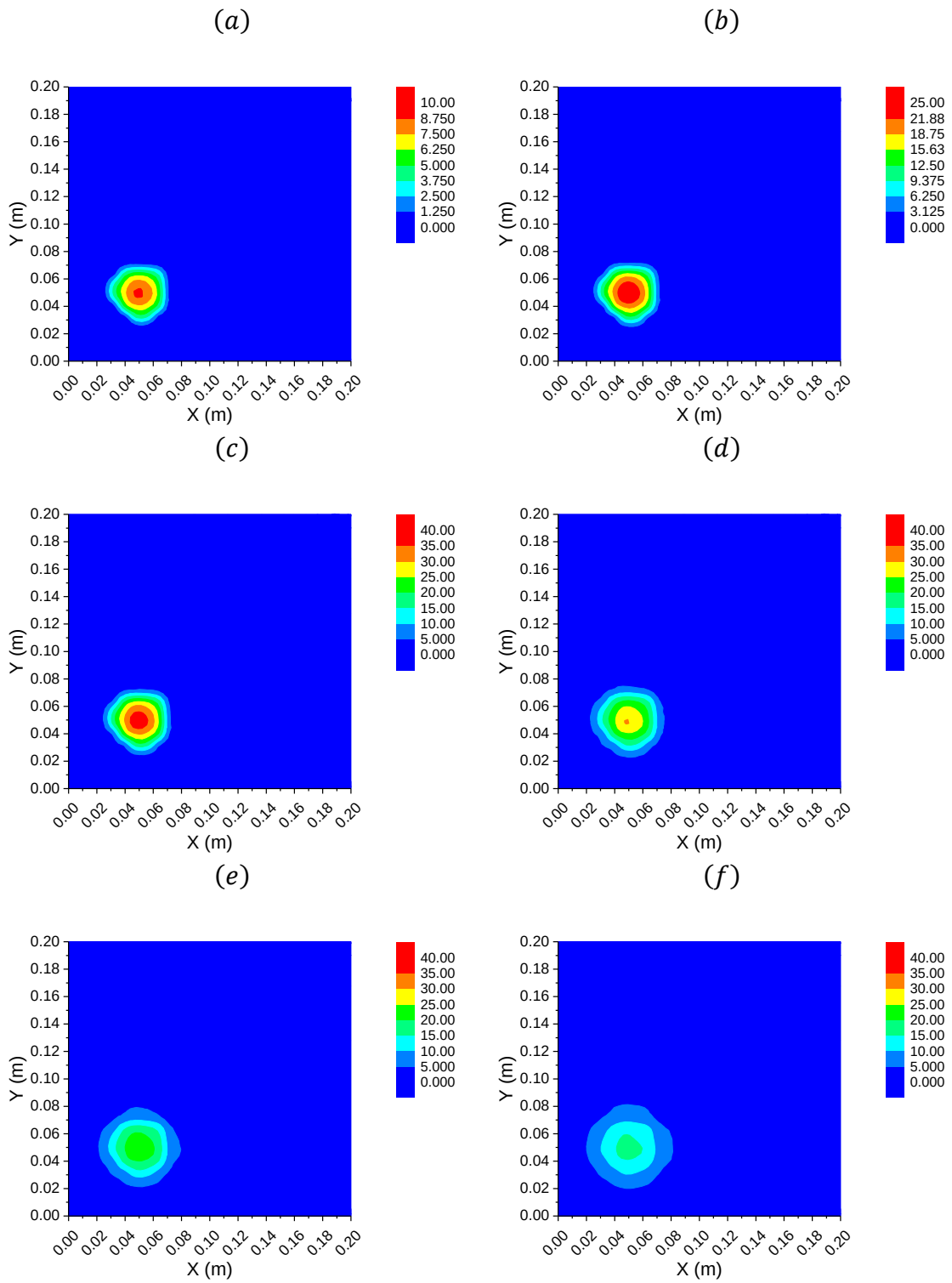


Figure 37. Temperature contour plots for Fourier-type heat conduction example with time dependent external source at (a) t = 1 second, (b) t = 3 seconds, (c) t = 5 seconds, (d) t = 7.5 seconds, (e) t = 10 seconds, and (f) t = 15 seconds

Similar to the first example, one can see that the heat being applied by the external source is causing a temperature increase that is radiating out from the point source. Then once the source stops acting on the domain, at $t = 5\ [s]$, it can be observed that the plate starts cooling, as indicated by the trend at the point $(0.05, 0.05)$; where the source was initially acting (the colors are shifting from red to blue).

Cattaneo-type Process Examples

The first additional example presented for Cattaneo-type heat conduction will be a steel circular disc with an external source continuously acting on the center of the disc and a constant temperature radius boundary condition. The material and geometry properties are as follows,

Table 20. Geometry and material properties for Fourier-type additional example

Parameter	Value
Radius	0.1 m
Specific Heat	486 J/kg/°K
Material Density	7870 kg/m^3
Thermal Conductivity	51.9 W/m/°K
Relaxation	5 s

As stated previously in the examples shown in Chapter 5, the relaxation parameter chosen, $\tau = 5$, is significantly higher than what one would expect in reality; this is to illustrate the differences between this model and the Fourier-type model. To approximate a point source a Gaussian distribution will be utilized, with the maximum energy level being,

$$Q_{max} = 10\ [W]\ , \qquad r_{source} = 0.01\ [m] \tag{143}$$

The initial condition and boundary condition can be seen below,

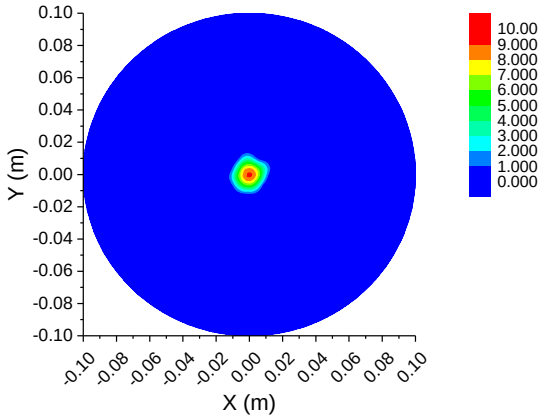
$$T_i = 0\ [K], \qquad T_{BC} = 0\ [K] \tag{144}$$

As shown previously in Chapter 5, Cattaneo-type hyperbolic heat conduction leads to numerical oscillations near the wave front. In order to minimize this, the maximum numerical dissipation was utilized via the GS4 *i*Integration framework, specifically,

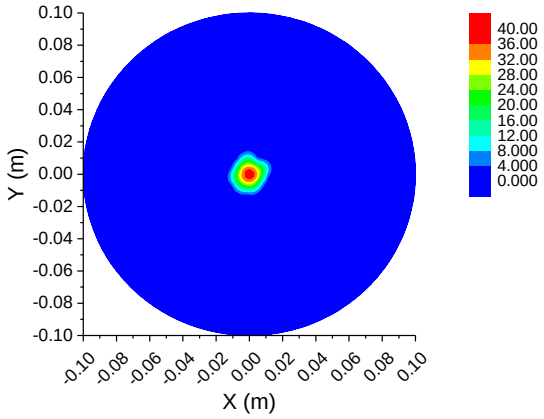
$$\rho_\infty^s = 0, \qquad \rho_\infty^{min} = 0, \qquad \rho_\infty^{max} = 1 \tag{145}$$

The results of this simulation can be seen below in Fig. 37,

(a)



(b)



(c)

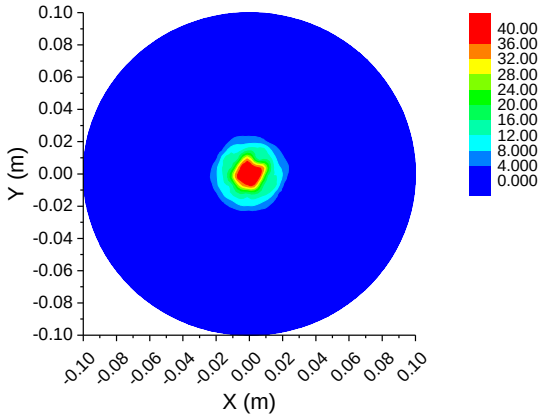


Figure 38. Temperature contour plots for Cattaneo-type heat conduction example with continuous external source at (a) t = 1 second, (b) t = 5 seconds, (c) t = 15 seconds

The second additional example presented will be a steel square with a time dependent external source acting within the lower left hand quadrant and a constant temperature edge boundary condition. The material and geometry properties are as follows,

Table 21. Geometry and material properties for Fourier-type additional example

Parameter	Value
Length	0.2 m
Width	0.2 m
Specific Heat	486 J/kg/°K
Material Density	7870 kg/m^3
Thermal Conductivity	51.9 W/m/°K
Relaxation	5 s

To approximate a point source a Gaussian distribution will be utilized, with the maximum energy level being,

$$Q_{max} = 10\ [W], \qquad r_{source} = 0.02\ [m] \tag{146}$$

This external source will be applied for the first 5 seconds, after that the source will be turned off. The initial condition and boundary condition can be seen below,

$$T_i = 0\ [K], \qquad T_{BC} = 0\ [K] \tag{147}$$

As shown previously in Chapter 5, Cattaneo-type hyperbolic heat conduction leads to numerical oscillations near the wave front. In order to minimize this, the maximum numerical dissipation was utilized via the GS4 *i*Integration framework, specifically,

$$\rho_\infty^s = 0, \qquad \rho_\infty^{min} = 0, \qquad \rho_\infty^{max} = 1 \tag{148}$$

The results from this simulation can be seen below in Figure 39.

(a) (b)

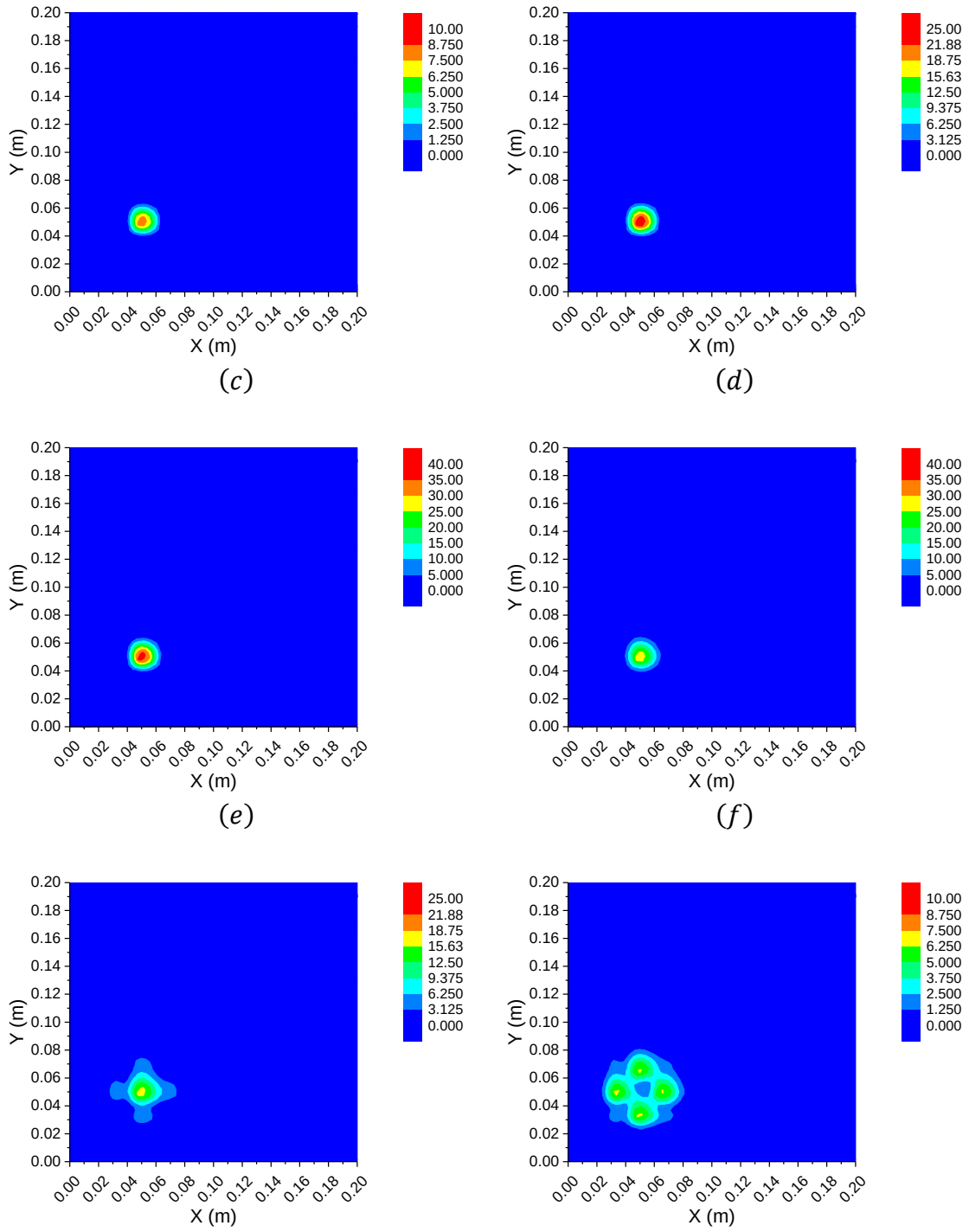


Figure 39. Temperature contour plots for Cattaneo-type heat conduction example with time dependent external source at (a) t = 1 second, (b) t = 3 seconds, (c) t = 5 seconds, (d) t = 7.5 seconds, (e) t = 10 seconds, and (f) t = 15 seconds

Similar to the first example one can see that the heat being applied by the external source is causing a temperature increase that is radiating out from the point source. Once the

source stops acting on the domain, at $t = 5\ [s]$, it can be observed that the plate starts cooling, as indicated by the trend at the point $(0.05, 0.05)$; where the source was initially acting, and the colors are shift from red to blue. Although unlike the Fourier-type process shown previously, it can be seen in Figure 39 (e) and (f) there is an oscillatory behavior indicated by the temperature peaks not located at the point source, it should also be noted that there was no smoothing applied within the simulation.

Jeffreys'-type Process Examples

The first additional example presented for Jeffreys'-type heat conduction is similar to the Fourier-type and Cattaneo-type examples. A steel circular disc with an external source continuously acting on the center of the disc and a constant temperature radius boundary condition is presented. The material and geometry properties are as follows,

Table 22. Geometry and material properties for Fourier-type additional example

Parameter	Value
Radius	0.1 m
Specific Heat	486 J/kg/°K
Material Density	7870 kg/m^3
Thermal Conductivity	51.9 W/m/°K
Relaxation	5 s
F_T	0.5

As stated previously in Chapter 5, the relaxation parameter chosen is significantly higher than what one would expect. This is to illustrate the differences between this model and

the two previously discussed models. To approximate a point source a Gaussian distribution will be utilized, with the maximum energy level being,

$$Q_{max} = 10\ [W], \qquad r_{source} = 0.01\ [m] \tag{149}$$

The initial condition and boundary condition can be seen below,

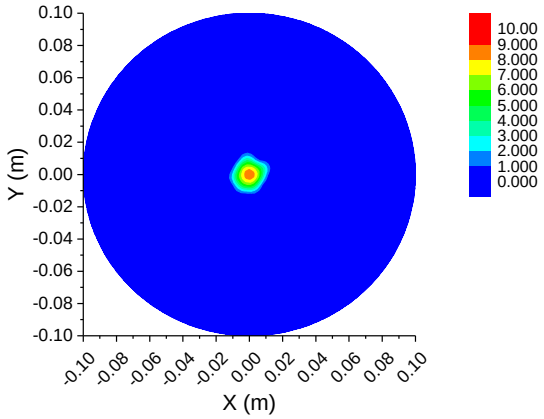
$$T_i = 0\ [K], \qquad T_{BC} = 0\ [K] \tag{150}$$

As shown previously in Chapter 5, Jeffreys'-type lagged parabolic heat conduction can lead to numerical oscillations. In order to minimize this, the maximum numerical dissipation was utilized via the GS4 *i*Integration framework, specifically,

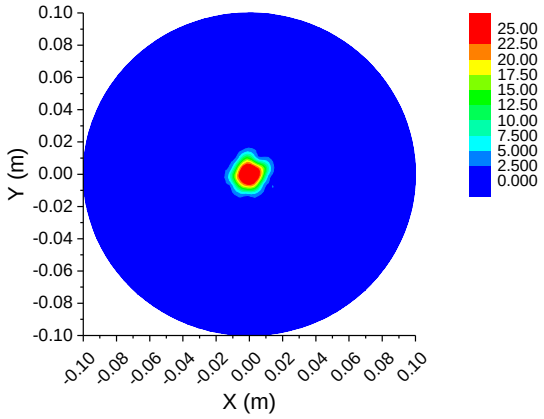
$$\rho_\infty^s = 0, \qquad \rho_\infty^{min} = 0, \qquad \rho_\infty^{max} = 1 \tag{151}$$

The results of this simulation can be seen below in Fig. 39,

(a)



(b)



(c)

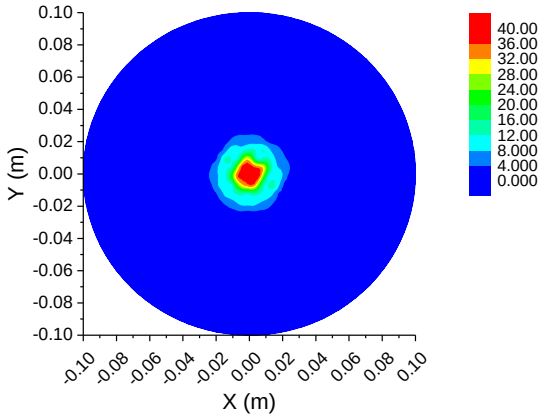


Figure 40. Temperature contour plots for Jeffreys'-type heat conduction example with continuous external source at (a) t = 1 second, (b) t = 5 seconds, (c) t = 15 seconds

Notice in the plots above that as time progresses the temperature at the center of the disc is increasing, as energy is being continually input by the external source. Furthermore, the temperature is slowly propagating outward in concentric circles, as is expected based upon the physics governing the problem. The results from this lagged parabolic heat conduction is very similar to the results presented previously for the hyperbolic heat conduction model. However there are differences in the rate of propagation due to the differences in constitutive equations.

The second additional example presented will be a steel square with a time dependent external source acting within the lower left hand quadrant and a constant temperature edge boundary condition. The material and geometry properties are as follows,

Table 23. Geometry and material properties for Fourier-type additional example

Parameter	Value
Length	0.2 m
Width	0.2 m
Specific Heat	486 J/kg/°K
Material Density	7870 kg/m^3
Thermal Conductivity	51.9 W/m/°K
Relaxation	5 s
F_T	0.5

To approximate a point source a Gaussian distribution will be utilized, with the maximum energy level being,

$$Q_{max} = 10\ [W], \qquad r_{source} = 0.02\ [m] \tag{152}$$

This external source will be applied for the first 5 seconds, after that the source will be turned off. The initial condition and boundary condition can be seen below,

$$T_i = 0\ [K], \qquad T_{BC} = 0\ [K] \tag{153}$$

As shown previously in Chapter 5, Cattaneo-type hyperbolic heat conduction leads to numerical oscillations near the wave front. In order to minimize this, the maximum numerical dissipation was utilized via the GS4 *i*Integration framework, specifically,

$$\rho_\infty^s = 0, \qquad \rho_\infty^{min} = 0, \qquad \rho_\infty^{max} = 1 \tag{154}$$

The results from this simulation can be seen below in Fig. 40,

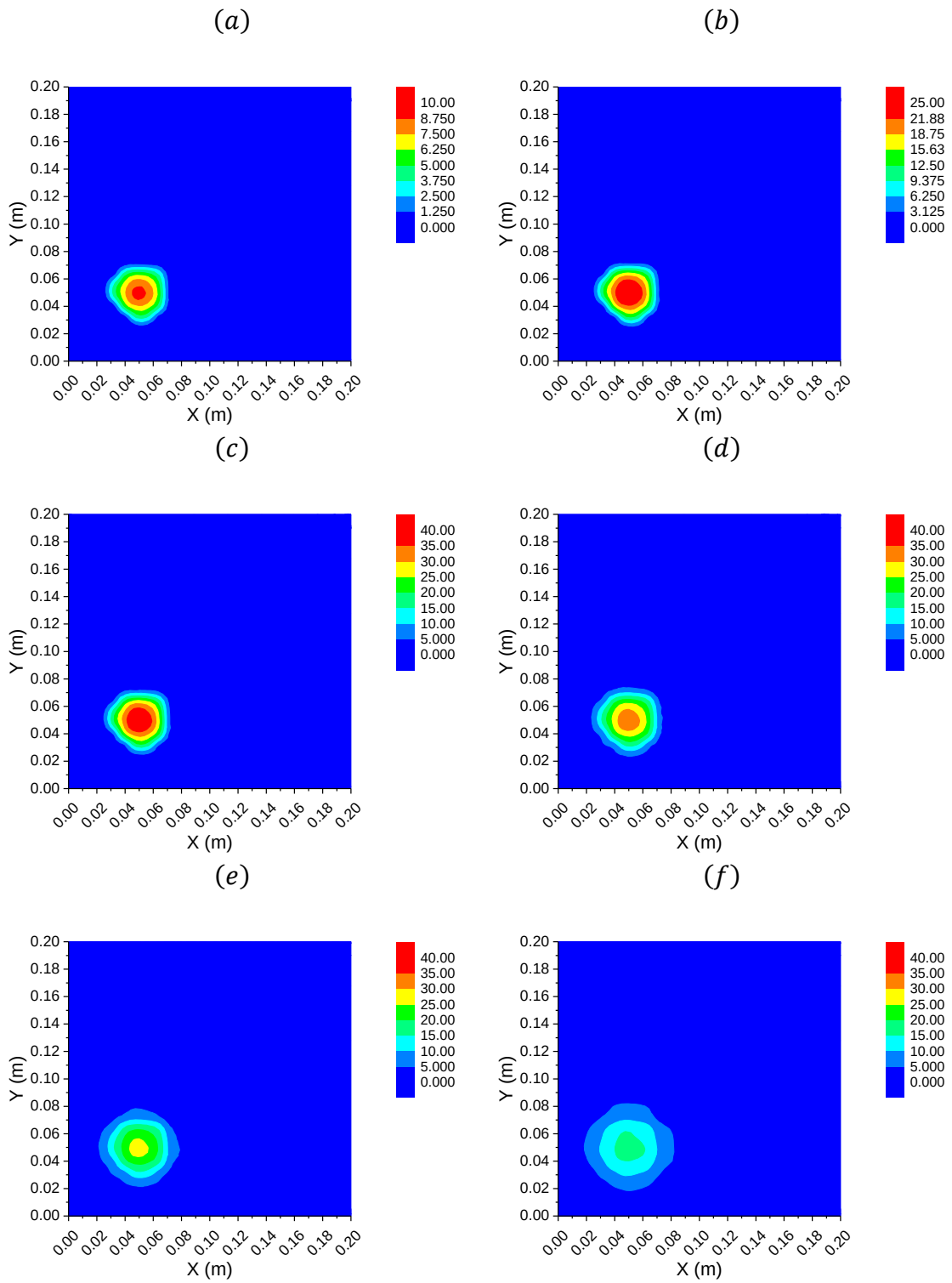


Figure 41. Temperature contour plots for Jeffreys'-type heat conduction example with time dependent external source at (a) t = 1 second, (b) t = 3 seconds, (c) t = 5 seconds, (d) t = 7.5 seconds, (e) t = 10 seconds, and (f) t = 15 seconds

Similar to the first example, one can see that the heat being applied by the external source is causing a temperature increase that is radiating out from the point source. Once the source stops acting on the domain, at $t = 5\ [s]$, it can be observed that the plate starts cooling, as indicated by the trend at the point $(0.05, 0.05)$; where the source was initially acting, and the colors are shifting from red to blue. Unlike the Cattaneo-type example shown previously, there appears to be no oscillatory behavior in the results after the external source stops acting on the plate. This is to be expected since the oscillations seen in Chapter 5 for the Jefferys'-type problems were significantly less than that of the Cattaneo-type processes.

Conclusions

In this chapter the numerical implementation of the C- and F- model with respect to Fourier-type parabolic, Cattaneo-type hyperbolic, and Jeffreys'-type lagged parabolic heat conduction with the presence of heat generation and stationary external source terms was presented. Results for Fourier-type heat conduction with both heat generation in the steady state and a time dependent external source in the transient state were validated against their respective analytical solutions. It is claimed that in addition to the validation provided in Chapter 5, the introduction of heat generation, or stationary external source terms, need only be validated for a single heat conduction process because if the implementation was correct for that process, then it would be assumed to be implemented correctly for the other processes.

Also, additional examples were presented for all three types of heat conduction models, to further illustrate the versatility of the model. However, the same types of numerical oscillations seen in Chapter 5 for Cattaneo-type heat conduction were present again in this chapter, even with the use of maximum numerical dissipation provided within the GS4 *i*Integration framework.

Chapter 7: MPS Model with Implementation of C- and F- Model with Moving Heat Source

Introduction

This chapter will cover the theory, implementation, and numerical validation, as well as provide additional numerical examples for the C- and F- model with the addition of a moving source in the proposed framework. This differs from the previous chapters in that the forcing vector is changing based upon the location of the source. It should be noted that the stationary, time-dependent examples from the Chapter 6 can be considered a special case of the more generalized formulation and algorithm presented in this chapter. Numerical results will be validated against published results for Fourier-type heat conduction with a moving source term. Similar to the validation in Chapter 6, it is

claimed that if the extension of the base algorithm is implemented correctly for Fourier-type processes then the algorithm would also be valid for Cattaneo-type and Jefferys'-type processes. Also, additional examples will be presented for all types of heat conduction to further illustrate the method's versatility. By extending the base formulation, discussed in the previous chapter, with the addition of a moving external source, the range of practical applications is further broadened. Practical applications of such simulations include, but are not limited to, the modeling of laser processes, welding, 3D printing, etc..

Theory

Numerical heat conduction utilizing the C- and F- model with the presence of a moving external heat source can be described with constitutive equations (78)-(80) and with the governing equation (81) As noted previously this type of problem is a single order in time set of differentiable equations that can be written in a two-field form as seen in equations (82)-(87), where the source matrix (**$\boldsymbol{S}$**) in equation (85) is a non-zero vector indicating the external heat source, or internal generation/sink. However, whereas in the previous chapter the source matrix (**$\boldsymbol{S}$**) was only calculated a single time, for a moving source the forcing vector will need to be recalculated at each time step based upon the movement of the source in the spatial domain utilizing the form,

$$\boldsymbol{S}(\boldsymbol{t}) = Q * \boldsymbol{W_S} \tag{155}$$

where Q is the magnitude of the moving source and $\boldsymbol{W_S}$ is a source weight vector, similar to, which can be defined as,

$$W_S(r) = \begin{cases} e^{-\frac{r^2}{r_s^2}} & (0 \leq r < r_s) \\ 0 & (r_s < r) \end{cases} \tag{156}$$

where r is the distance between a particle and the external source position and r_S is the radius of influence for the external source. This formulation allows for the source term to be dispersed to various particles based upon it's region of influence; it is assumed that the source is circular in two-dimensions and that the source is distributed based upon the weight function. For the transient examples, the GS4 *i*Integration framework for first order systems (GS4-1); as shown in Chapter 3, will be implemented for time integration.

Implementation

The algorithm for the problems in this chapter is described in Figure 30. In contrast to the algorithm presented in Chapter 6, the source is now moving so it's position and region of influence; represented by the source weight vector, need to be calculated at each time step.

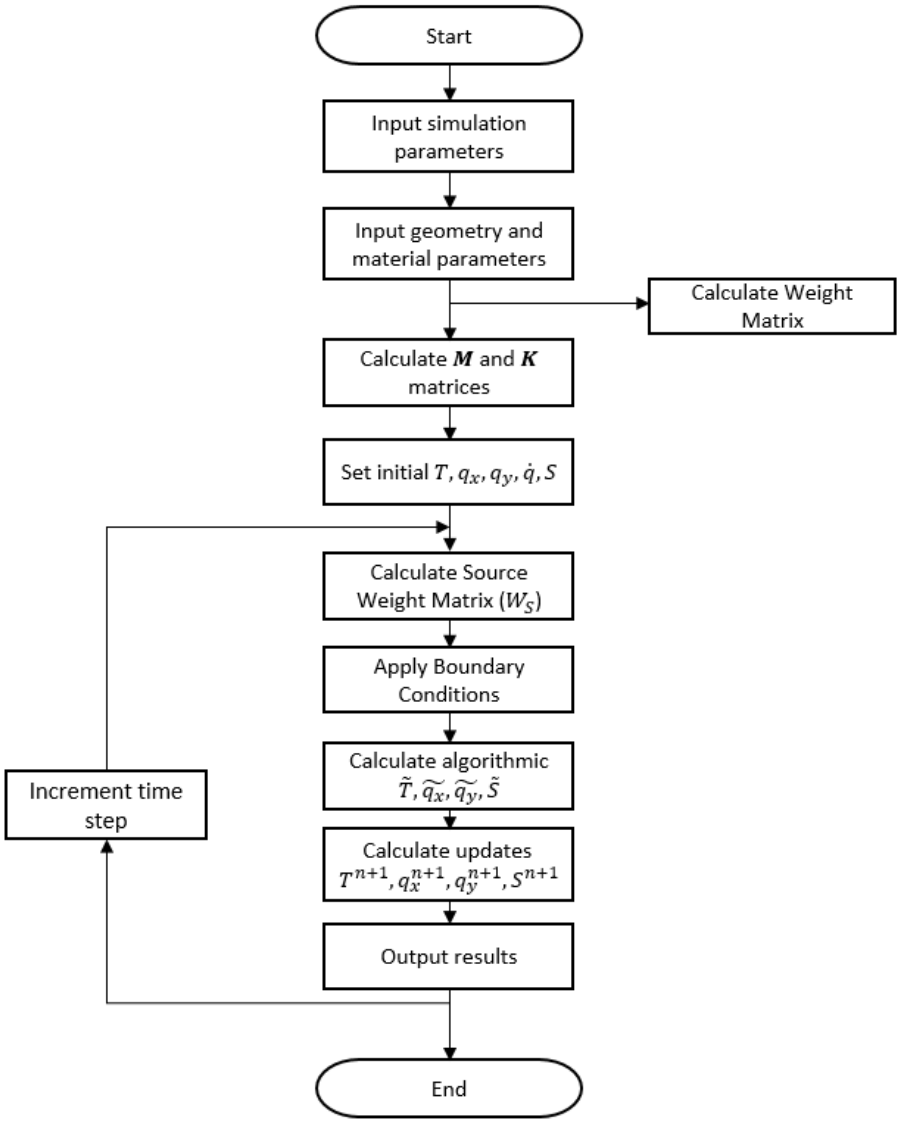


Figure 42. Computational algorithm of proposed framework encompassing Chapter 6 examples

Numerical Validation

For the purpose of validating the implementation of a moving source term, numerical results will be compared to published results for a Fourier-type process. In addition to the validation work done conducted in Chapters 5 and 6, it will be assumed that if the numerical results for Fourier-type processes presented in this chapter match the published

results, then by extension the results from the model for Cattaneo-type and Jeffreys'-type processes will also be valid.

Fourier Heat Conduction (Parabolic Heat Conduction [F_T= 1])

Consider a point source moving across a rectangular plate. An illustration of the problem can be seen below,

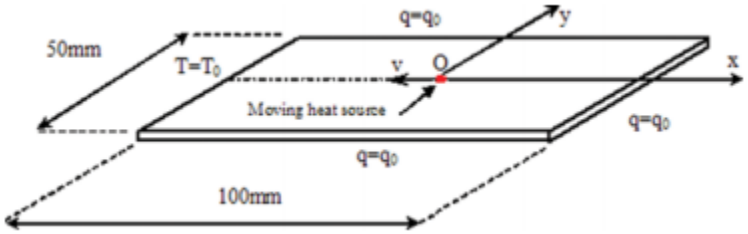


Figure 43. Illustration of moving heat source numerical example

With the following boundary conditions,

$$T_0 = 20\ [°C], \qquad q_0 = 0.001\left[\frac{W}{mm^2}\right] \tag{157}$$

The parameters taken from [28] for this example are as follows,

Table 24. Geometry, Material, and Source parameters for moving source numerical validation

Parameter	Value
Length	100 mm
Width	50 mm
Specific Heat (c_p)	658 J/kg/°C
Material Density (ρ)	$7.6 \cdot 10^{-6}$ kg/mm^3
Thermal Conductivity (k)	0.025 W/mm/°C
Maximum Heat Source (Q_{max})	5 W/mm^3
Radius of Heat Source (r_{source})	2 mm
Velocity of Heat Source (v_{source})	2 mm/s

In the paper, the simulation was run for a total of 45 seconds, due to the length of this simulation an explicit time integration scheme Specifically the explicit form of GS4-2 was utilized with,

$$\eta_1 = \eta_2 = \eta_3 = 1 \tag{158}$$

which as previously mentioned recovers the implicit form of GS4-2 with the time stepping parameter,

$$dt = 0.1\ [s] \tag{159}$$

and,

$$\rho_s = 0, \qquad \rho_{min} = 0, \qquad \rho_{max} = 1 \tag{160}$$

which is equivalent to $(\rho_s = 0,\ \rho = 0)$ in GS4-1, resulting in Gear's Method for time stepping. This was chosen due to its damping characteristics. With an acceptable tolerance per iteration of 0.001. Temperature contour plots at $t = 5, 15,$ and 25 seconds

steps taken from [28] will be used a basis comparison, those can be seen below with the results from the numerical simulation as well,

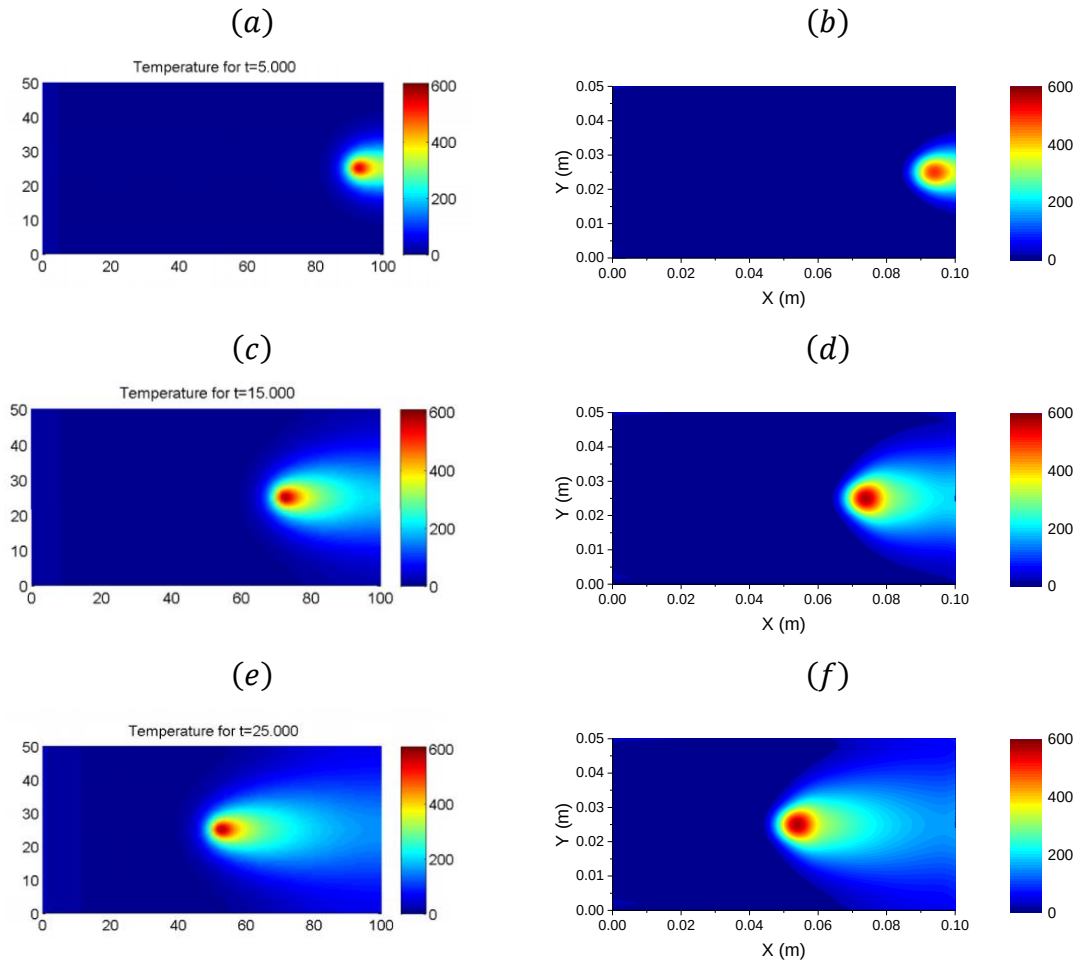
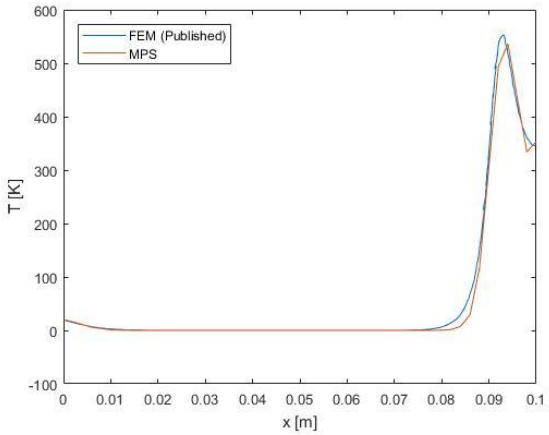


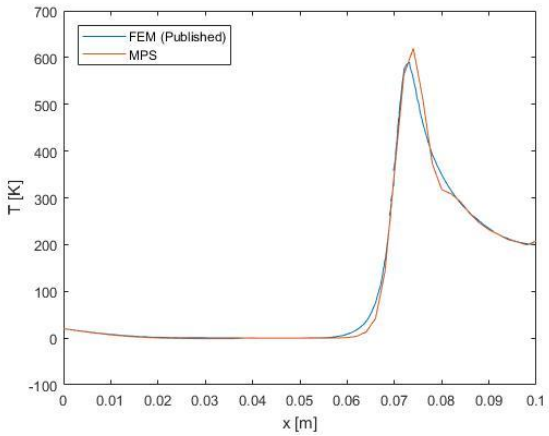
Figure 44. Temperature contour plots at t = 5, 15, and 25 seconds for the numerical example where (a), (c), (e) are the previously published results and (b), (d), and (f) are the numerical results from this model

Qualitatively, it can be seen when comparing $(a), (c)$, and (f) to $(b), (d)$, and (f), respectively, that the published results and numerical results match. A more quantitative comparison can be seen by comparing the temperature distribution along the centerline for the same time steps as shown above, those plots can be seen below,

(a)



(b)



(c)

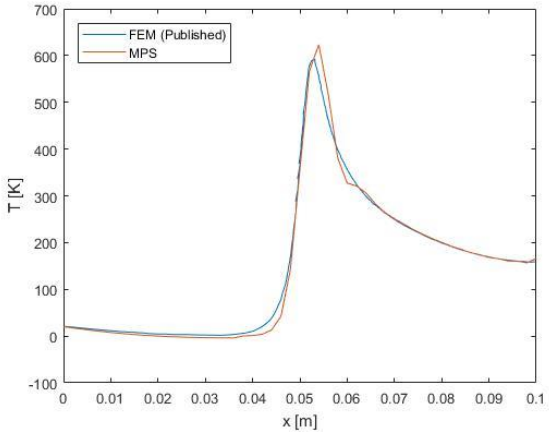


Figure 45. Comparison plots of centerline temperature at t = 5, 15, and 25 seconds respectively

From the plots above in Figure 45, there is significant agreement between the numerical solutions at the different time steps and the published solutions, the differences are due to the first order nature of the particle method in comparison to the second do order nature of the meshed method. To minimize the sharp gradients present in the data, smoothing could be applied, however it was determined it would be better to show the data without smoothing as to illustrate the base capability of the proposed framework.

Additional Examples

To further illustrate the versatility of the model additional numerical examples for Fourier-type parabolic, Cattaneo-type hyperbolic, and Jeffreys'-type lagged parabolic heat conduction will be presented. These examples will utilize geometries and material properties that vary from the validation example shown previously.

Fourier-type Process Examples

The first additional example presented will be a square steel plate with a source moving and changing direction; specifically transcribing a square pattern starting in the bottom right hand corner of the domain, moving in a clockwise direction to illustrate a motion similar to that of cutting a square hole into a steel plate. The geometry, material, and source parameters are as follows,

Table 25. Geometry, Material, and Source parameters for Fourier-type heat conduction with moving source numerical example

Parameter	Value
Length	50 mm
Width	50 mm
Specific Heat (c_p)	658 J/kg/°C
Material Density (ρ)	$7.6 \cdot 10^{-6}$ kg/mm^3
Thermal Conductivity (k)	0.025 W/mm/°C
Maximum Heat Source (Q_{max})	50 W/m^3

The initial condition is a uniform temperature, with the flux in the normal direction at each boundary being zero for the boundary conditions and,

$$T_i = 0\ [K], \qquad q_{BC} = \begin{cases} q_x = 0\ \left[\frac{W}{m^2}\right] \forall\ x = 0\ or\ 50\ [mm] \\ q_y = 0\ \left[\frac{W}{m^2}\right] \forall\ y = 0\ or\ 50\ [mm] \end{cases} \tag{161}$$

The simulation will be conducted over a four second period, with time step,

$$dt = 0.005\ [s] \tag{162}$$

in which the source's trajectory will be governed by its starting position and the velocity of the source depending on the temporal position. The source parameters can be seen below,

$$p_{source,initial} = (0.0375, 0.0125)\ [m]$$

$$v_{source} = \begin{cases} (-0.025, 0)\ \left[\frac{m}{s}\right], & t < 1\ [s] \\ (0, 0.025)\ \left[\frac{m}{s}\right], & 1 \le t < 2\ [s] \\ (0.025, 0)\ \left[\frac{m}{s}\right], & 2 \le t < 3\ [s] \\ (0, -0.025)\ \left[\frac{m}{s}\right], & 3 \le t\ [s] \end{cases} \tag{163}$$

The temperature distributions resulting from the simulation for various time steps can be seen in the figure below. Notice that as the source makes its way clockwise around the path the heat being accumulated in the center of the plate.

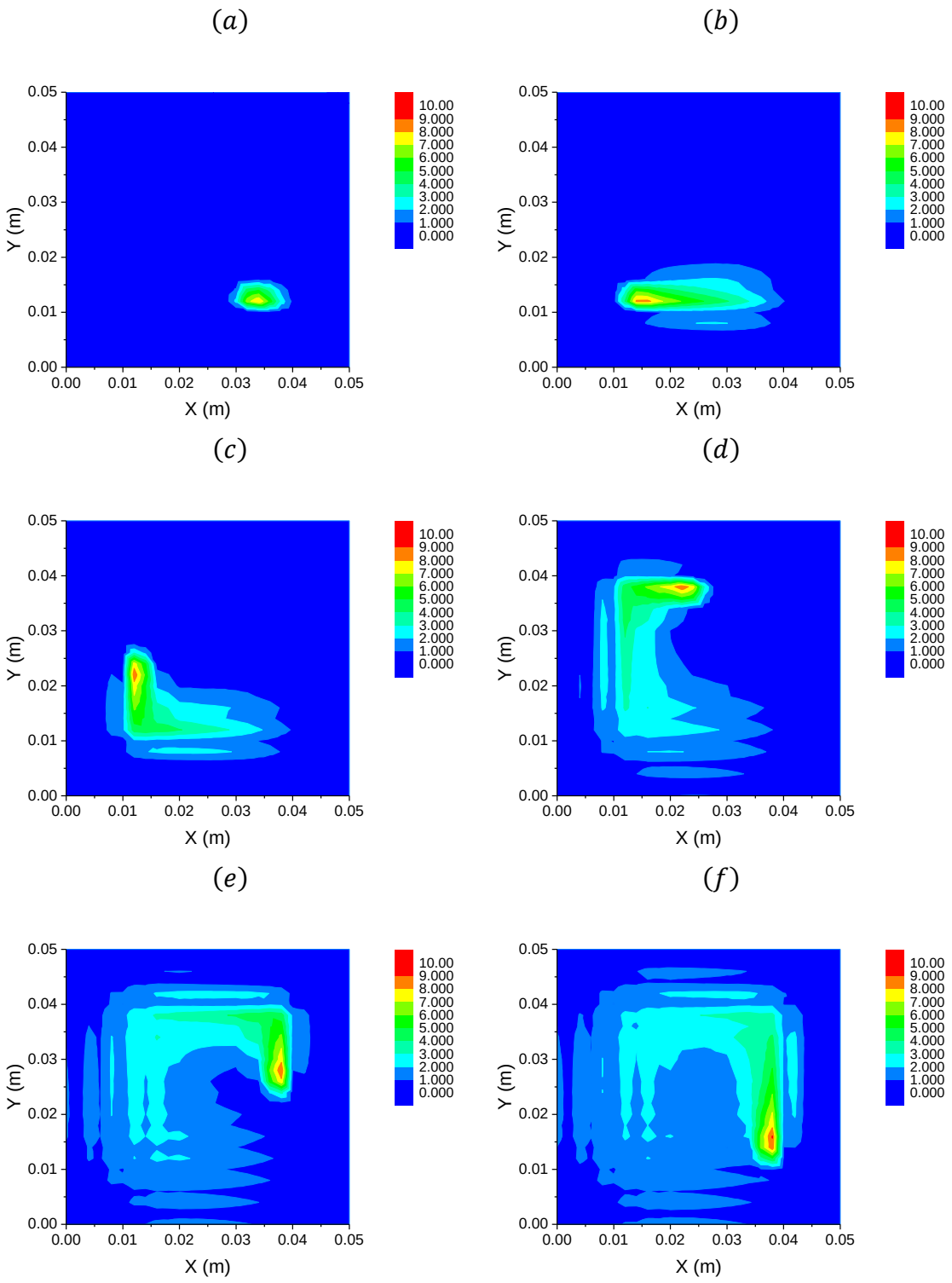


Figure 46. Temperature contour plots for Fourier-type heat conduction example with time moving external source at (a) t = 0.25 seconds, (b) t = 1.0 seconds, (c) t = 1.5 seconds, (d) t = 2.5 seconds, (e) t = 3.5 seconds, and (f) t = 4 seconds

Also, no data smoothing was utilized in the data representation above, which is why there are sharp changes in the temperature bands along the pathway of the source.

Similar to the problem shown above, we could instead model the same process but instead of a single external source transcribing the entire path, we could utilize two sources and split the path between them. For this simulation we will utilize the same geometry, material, and simulation parameters, as well as, initial conditions, boundary conditions, and time step. However since we are now using two individual sources, those parameters are as follows,

$$
\begin{gathered}
p_{source\ 1,initial} = (0.0375, 0.0125)\ [m] \\
v_{source\ 1} = \begin{cases} (-0.025, 0)\ \left[\frac{m}{s}\right], & t < 1\ [s] \\ (0, 0.025)\ \left[\frac{m}{s}\right], & 1 \leq t < 2\ [s] \end{cases} \\
p_{source\ 2,initial} = (0.0125, 0.0275)\ [m] \\
v_{source\ 2} = \begin{cases} (0.025, 0)\ \left[\frac{m}{s}\right], & t < 1\ [s] \\ (0, -0.025)\ \left[\frac{m}{s}\right], & 1 \leq t < 2\ [s] \end{cases}
\end{gathered}
\tag{164}
$$

Results from the processes now using multiple heat sources can be seen below. As expected, the brighter colors at the various time steps, in comparison to the single source problem, indicate the extra heat being injected into the steel plate, however the overall maximum temperature is about the same between the two simulations.

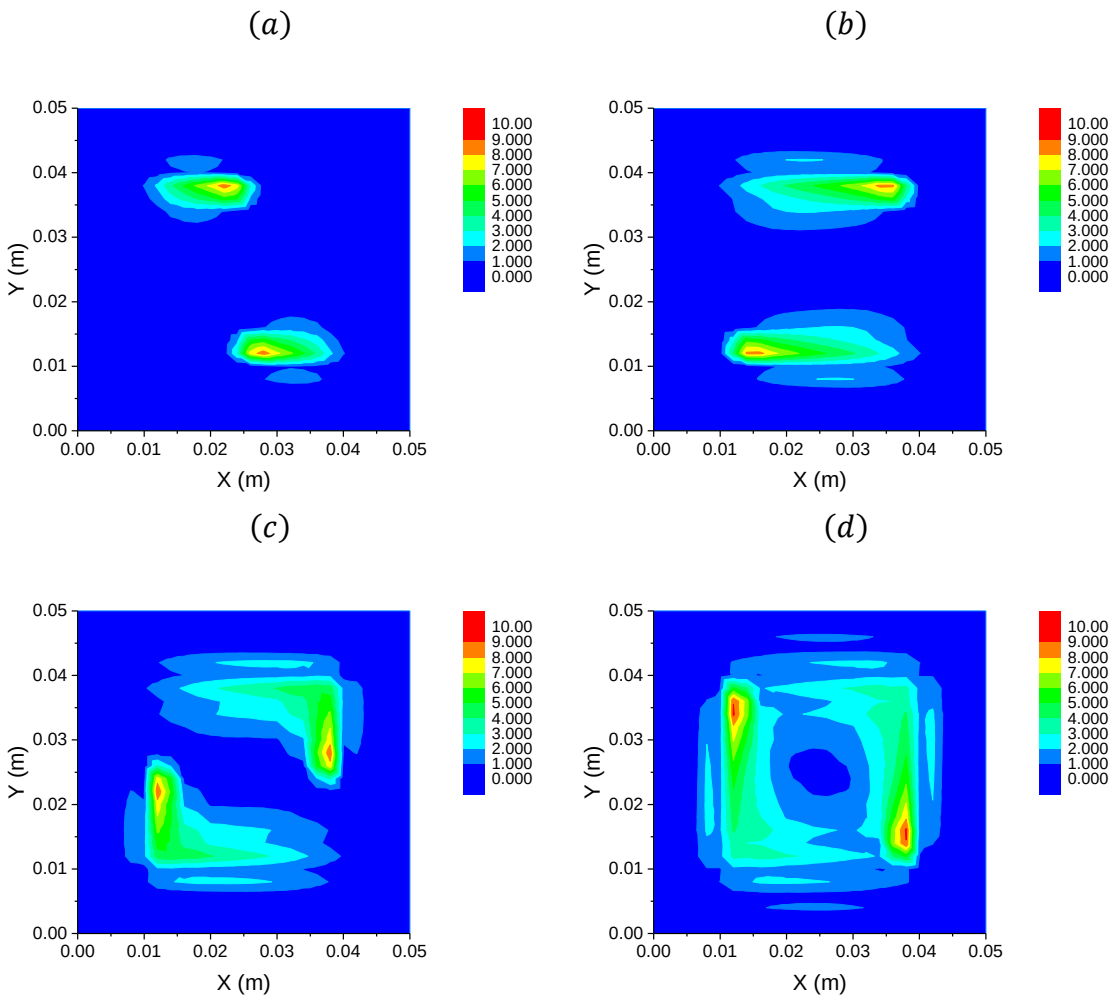


Figure 47. Numerical results for Fourier-type example with multiple sources at (a) t = 0.5 seconds, (b) t = 1.0 seconds, (c) t = 1.5 seconds, and (d) t = 2.0 seconds

Cattaneo-type Process Examples

The first additional example presented will be a square steel plate with a source moving and changing direction; specifically transcribing a square pattern starting in the bottom right hand corner of the domain, moving in a clockwise direction to illustrate a motion similar to that of cutting a square hole into a steel plate. The geometry, material, and source parameters are as follows,

Table 26. Geometry, Material, and Source parameters for Cattaneo-type heat conduction with moving source numerical example

Parameter	Value
Length	50 mm
Width	50 mm
Specific Heat (c_p)	658 J/kg/°C
Material Density (ρ)	$7.6 \cdot 10^{-6}$ kg/mm^3
Thermal Conductivity (k)	0.025 W/mm/°C
Relaxation (τ)	5 s
Maximum Heat Source (Q_{max})	50 W/m^3

The initial condition is a uniform temperature, with the flux in the normal direction at each boundary being zero for the boundary conditions and,

$$T_i = 0\ [K], \qquad q_{BC} = \begin{cases} q_x = 0\ \left[\frac{W}{m^2}\right] \forall\ x = 0\ or\ 50\ [mm] \\ q_y = 0\ \left[\frac{W}{m^2}\right] \forall\ y = 0\ or\ 50\ [mm] \end{cases} \tag{165}$$

The simulation will be conducted over a four second period, with time step,

$$dt = 0.005\ [s] \tag{166}$$

in which the source's trajectory will be governed by its starting position and the velocity of the source depending on the temporal position. The source parameters can be seen below,

$$p_{source,initial} = (0.0375, 0.0125)\ [m] \tag{167}$$

$$v_{source} = \begin{cases} (-0.025, 0)\ \left[\frac{m}{s}\right], & t < 1\ [s] \\ (0, 0.025)\ \left[\frac{m}{s}\right], & 1 \leq t < 2\ [s] \\ (0.025, 0)\ \left[\frac{m}{s}\right], & 2 \leq t < 3\ [s] \\ (0, -0.025)\ \left[\frac{m}{s}\right], & 3 \leq t\ [s] \end{cases}$$

The temperature distributions resulting from the simulation for various time steps can be seen in the figure below. Notice that as the source makes its way clockwise around the path the heat being accumulated in the center of the plate.

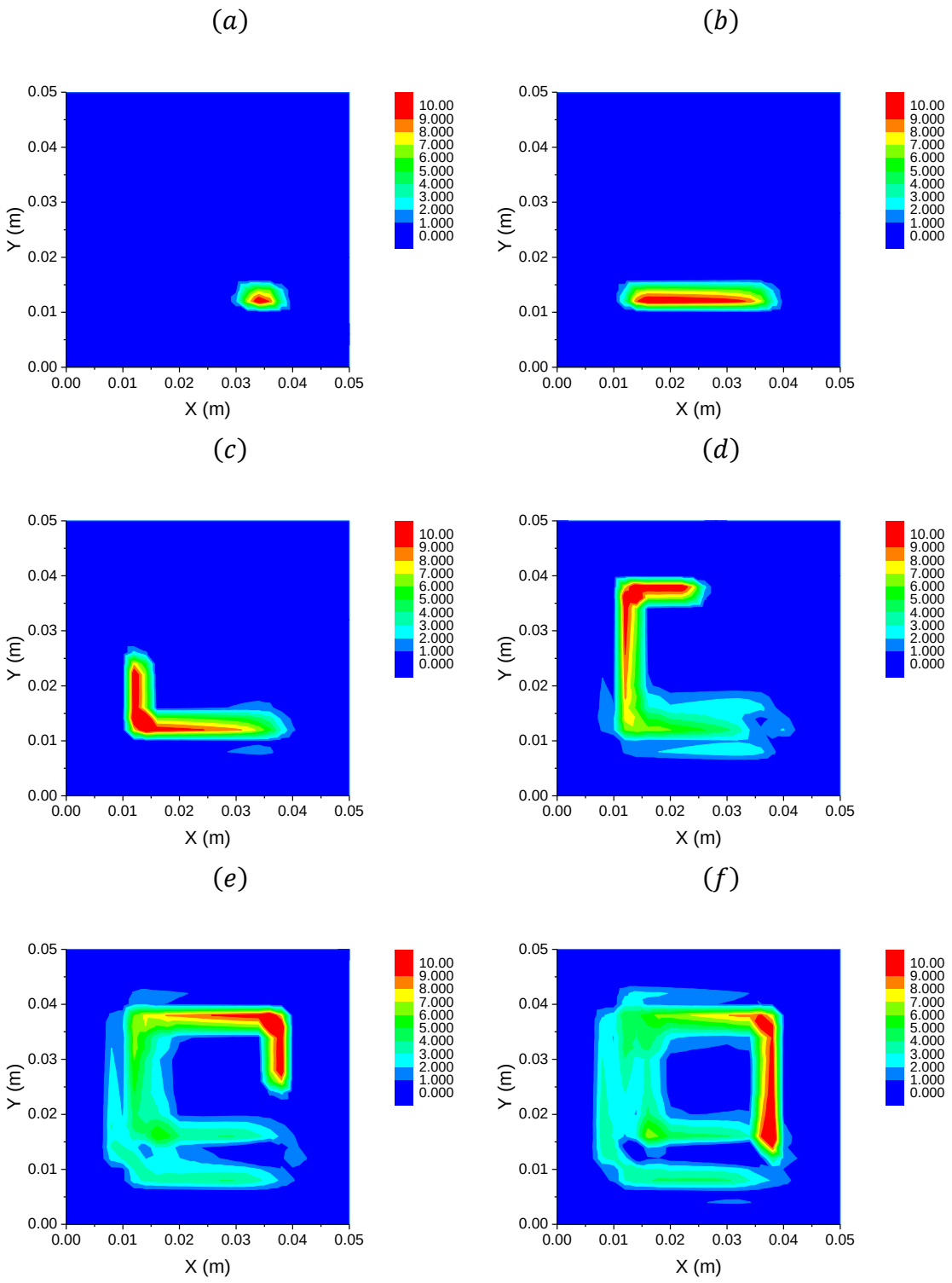


Figure 48. Temperature contour plots for Cattaneo-type heat conduction example with time moving external source at (a) t = 0.25 seconds, (b) t = 1.0 seconds, (c) t = 1.5 seconds, (d) t = 2.5 seconds, (e) t = 3.5 seconds, and (f) t = 4 seconds

Similar to the problem shown above, we could instead model the same process but instead of a single external source transcribing the entire path, we could utilize two sources and split the path between them. For this simulation we will utilize the same geometry, material, and simulation parameters, as well as, initial conditions, boundary conditions, and time step. However since we are now using two individual sources, those parameters are as follows,

$$p_{source\ 1,initial} = (0.0375, 0.0125)\ [m]$$
$$v_{source\ 1} = \begin{cases} (-0.025, 0)\ \left[\frac{m}{s}\right], & t < 1\ [s] \\ (0, 0.025)\ \left[\frac{m}{s}\right], & 1 \leq t < 2\ [s] \end{cases}$$
$$p_{source\ 2,initial} = (0.0125, 0.0275)\ [m] \tag{168}$$
$$v_{source\ 2} = \begin{cases} (0.025, 0)\ \left[\frac{m}{s}\right], & t < 1\ [s] \\ (0, -0.025)\ \left[\frac{m}{s}\right], & 1 \leq t < 2\ [s] \end{cases}$$

Results from the processes now using multiple heat sources can be seen below. Similar to as what was seen for the Fourier-type examples, the brighter colors at the various time steps, in comparison to the single source problem, indicate the extra heat being injected into the steel plate, however the overall maximum temperature is about the same between the two simulations.

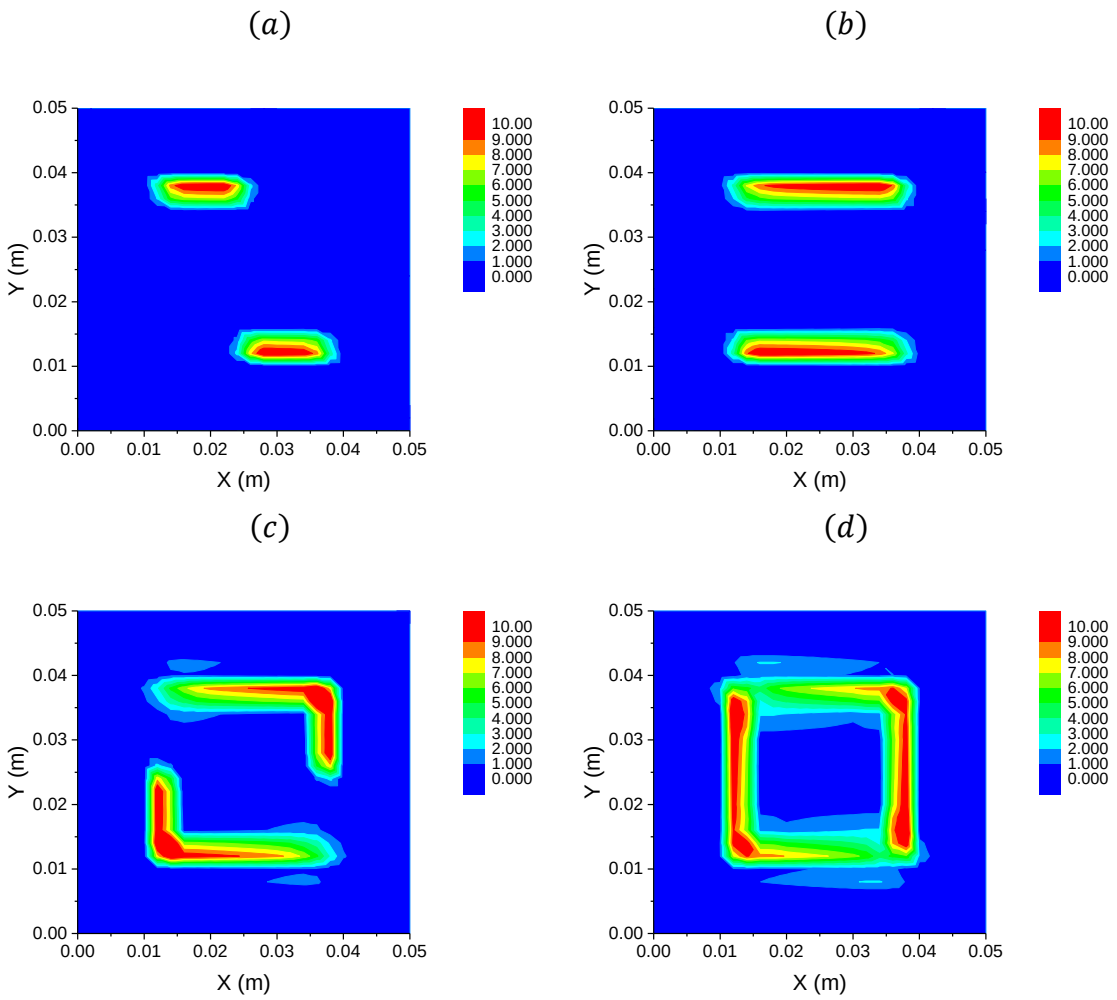


Figure 49. Numerical results for Cattaneo-type example with multiple sources at (a) t = 0.5 seconds, (b) t = 1.0 seconds, (c) t = 1.5 seconds, and (d) t = 2.0 seconds

Lastly let's consider a problem similar to the Fourier validation problem where there is a single heat source moving horizontally across a plate towards an edge. The geometry, material, and source parameters are as follows,

Table 27. Geometry, Material, and Source parameters for Cattaneo-type heat conduction with moving source numerical example

Parameter	Value
Length	50 mm
Width	25 mm
Specific Heat (c_p)	100
Material Density (ρ)	100
Thermal Conductivity (k)	100
Relaxation (τ)	10
Maximum Heat Source (Q_{max})	1

The initial condition is a uniform temperature, with the flux in the normal direction at each boundary being zero for the boundary conditions and,

$$T_i = 0\,, \qquad T_{BC} = \begin{cases} T = 0\ \left[\frac{W}{m^2}\right] \forall\ x = 0\ or\ 50 \\ T = 0\ \left[\frac{W}{m^2}\right] \forall\ y = 0\ or\ 25 \end{cases} \tag{169}$$

The simulation will be conducted over a four second period, with time step,

$$dt = \ 0.0005\ [s] \tag{170}$$

with,

$$\rho_s = 0, \qquad \rho_{min} = 0, \qquad \rho_{max} = 1 \tag{171}$$

which is equivalent to $(\rho_s = 0,\ \rho = 0)$ in GS4-1, resulting in Gear's Method for time stepping. This was chosen due to its damping characteristics. Furthermore, the source's trajectory will be governed by its starting position and the velocity of the source depending on the temporal position. The source parameters can be seen below,

$$p_{source,initial} = (25, 12.5)$$
$$v_{source} = -10 \tag{172}$$

numerical results from this problem can be seen below, specifically at $t = 0.1\ [s]$,

(a) (b)

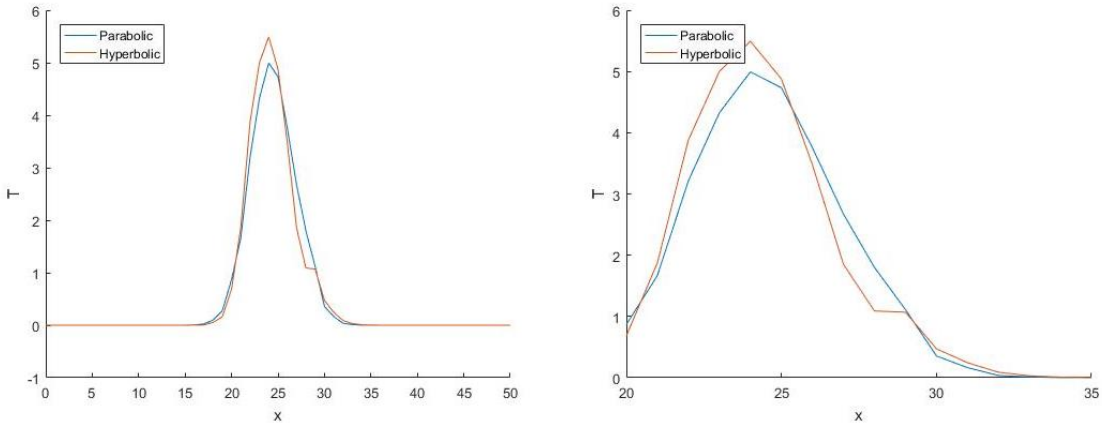


Figure 50. Numerical results from the simulation, specifically the temperature distribution along the centerline, y = 12.5 mm, for (a) the entire centerline and (b) a zoomed spatial discretization

Note the difference in shape between the parabolic and hyperbolic results. In Figure 50 (b), it can be seen that there is a wave like nature moving in the positive x direction, illustrating the difference between the two constitutive models.

Jeffreys'-type Process Examples

The first additional example presented will be a square steel plate with a source moving and changing direction; specifically transcribing a square pattern starting in the bottom right hand corner of the domain, moving in a clockwise direction to illustrate a motion similar to that of cutting a square hole into a steel plate. The geometry, material, and source parameters are as follows,

Table 28. Geometry, Material, and Source parameters for Jeffreys'-type heat conduction with moving source numerical example

Parameter	Value
Length	50 mm
Width	50 mm
Specific Heat (c_p)	658 J/kg/°C
Material Density (ρ)	$7.6 \cdot 10^{-6}$ kg/mm^3
Thermal Conductivity (k)	0.025 W/mm/°C
Relaxation (τ)	5 s
F_T	0.5
Maximum Heat Source (Q_{max})	50 W/m^3

The initial condition is a uniform temperature, with the flux in the normal direction at each boundary being zero for the boundary conditions and,

$$T_i = 0\ [K], \qquad q_{BC} = \begin{cases} q_x = 0\ [W/m^2\ ] \ \forall\ x = 0\ or\ 50\ [mm] \\ q_y = 0\ [W/m^2\ ] \ \forall\ y = 0\ or\ 50\ [mm] \end{cases} \tag{173}$$

The simulation will be conducted over a four second period, with time step,

$$dt = \ 0.005\ [s] \tag{174}$$

in which the source's trajectory will be governed by its starting position and the velocity of the source depending on the temporal position. The source parameters can be seen below,

$$p_{source,initial} = (0.0375, 0.0125)\ [m]$$
$$v_{source} = \begin{cases} (-0.025, 0)\ [m/s], & t < 1\ [s] \\ (0, 0.025)\ [m/s], & 1 \le t < 2\ [s] \\ (0.025, 0)\ [m/s], & 2 \le t < 3\ [s] \\ (0, -0.025)\ [m/s], & 3 \le t\ [s] \end{cases} \tag{175}$$

The temperature distributions resulting from the simulation for various time steps can be seen in the figure below. Notice that as the source makes its way clockwise around the path the heat being accumulated in the center of the plate.

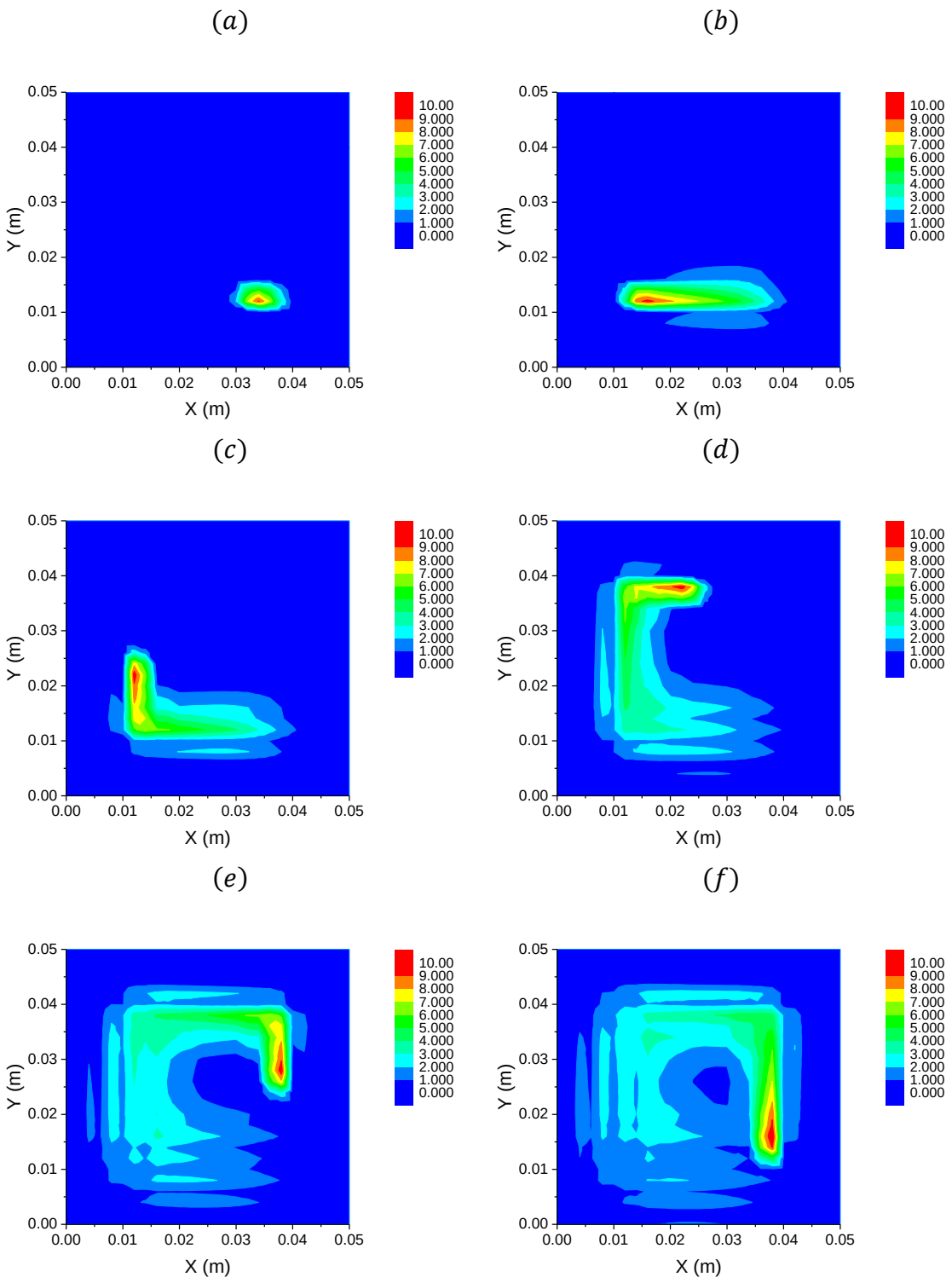


Figure 51. Temperature contour plots for Jeffreys'-type heat conduction example with time moving external source at (a) t = 0.25 seconds, (b) t = 1.0 seconds, (c) t = 1.5 seconds, (d) t = 2.5 seconds, (e) t = 3.5 seconds, and (f) t = 4 seconds

Similar to the problem shown above, we could instead model the same process but instead of a single external source transcribing the entire path, we could utilize two sources and split the path between them. For this simulation we will utilize the same geometry, material, and simulation parameters, as well as, initial conditions, boundary conditions, and time step. However since we are now using two individual sources, those parameters are as follows,

$$\begin{gathered} p_{source\,1,initial} = (0.0375, 0.0125)\ [m] \\ v_{source\,1} = \begin{cases} (-0.025, 0)\ [m/s], & t < 1\ [s] \\ (0, 0.025)\ [m/s], & 1 \le t < 2\ [s] \end{cases} \\ p_{source\,2,initial} = (0.0125, 0.0275)\ [m] \\ v_{source\,2} = \begin{cases} (0.025, 0)\ [m/s], & t < 1\ [s] \\ (0, -0.025)\ [m/s], & 1 \le t < 2\ [s] \end{cases} \end{gathered} \tag{176}$$

Results from the processes now using multiple heat sources can be seen below. Similar to as what was seen for both the Fourier-type and Cattaneo-type examples, the brighter colors at the various time steps, in comparison to the single source problem, indicate the extra heat being injected into the steel plate, however the overall maximum temperature is about the same between the two simulations.

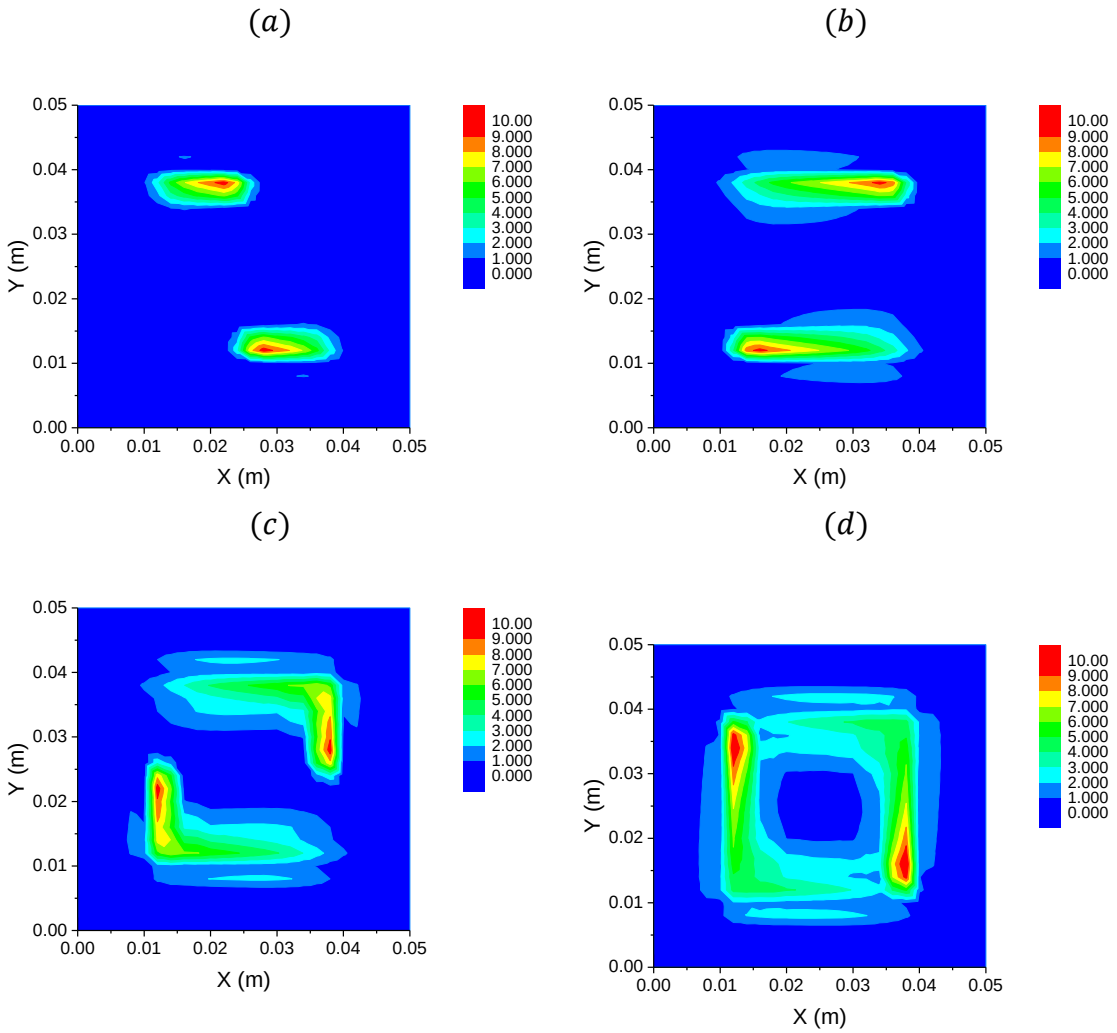


Figure 52. Numerical results for Jeffreys'-type example with multiple sources at (a) t = 0.5 seconds, (b) t = 1.0 seconds, (c) t = 1.5 seconds, and (d) t = 2.0 seconds

Conclusions

In this chapter the numerical implementation of the C- and F- model with respect to Fourier-type parabolic, Cattaneo-type hyperbolic, and Jeffreys'-type lagged parabolic heat conduction with the addition of a moving external source was presented. Results for Fourier-type heat conduction with a single moving source term were validated against previously published results for the same problem, utilizing a different modeling technique. It was found that the numerical results based upon the method proposed in this

thesis matched the previously published numerical results. It is also claimed that in addition to the validation provided in Chapters 5 and 6, that the addition of a moving source term need only be validated for a single heat conduction process because if the implementation was correct for that process then it would be assumed to be implemented correctly for the other processes.

Also, additional examples were presented for all three types of heat conduction, to further illustrate the versatility of the model. However, the same types of numerical oscillations seen in Chapters 5 and 6 for Cattaneo-type heat conduction were present again in this chapter, even with the use of maximum numerical dissipation provided within the GS4 *i*Integration framework. Once again corroborating the assertion that this framework might not be the best suited for numerical simulations exhibiting wave like behavior.

Chapter 8: Conclusion

There are three ways to solve any problem in engineering; analytically, experimentally, or numerically. Analytical solutions tend to be limited to simple geometries and mathematical formulations, and experimental solutions are labor intensive and can become expensive based upon scale; either small or large. Numerical simulations allow for relatively fast solutions to extremely complex problems in both the small scale and large scale, and allowing for multiple iterations of inputs. They are the preferred method in many practical engineering applications.

Presented within this thesis is a numerical method for heat conduction processes that is more versatile than previously proposed methods. This was accomplished by implementing the C- and F- model for heat conduction processes; which has been shown to be able to recover the currently prevalent models, in a meshless based particle method. It allows for uniform domain discretization, with the GS4 *i*Integration time stepping

scheme; which, to this author's knowledge, is the most general and wide-reaching time-stepping algorithm proposed in the literature to date.

Results were presented showing that this framework was able to recover steady state and transient results for three types of conduction models, all of which are encompassed within the C- and F- model; Fourier-type parabolic heat conduction, Cattaneo-type hyperbolic heat conduction, and Jeffreys'-type lagged parabolic heat conduction. This was done at three different levels; firstly with just simple heat conduction without source terms, secondly, heat conduction with stationary heat sources terms that were time-dependent, and lastly, with moving heat sources terms, illustrating the versatility of the framework.

However, it was shown that for Cattaneo-type heat conduction, and, to a lesser extent, Jeffreys'-type heat conduction, problems the formulation within the proposed framework led to numerical oscillations in the results. Some of these oscillations could be removed utilizing controllable numerical dissipation built into the GS4 *i*Integration time stepping scheme.

In conclusion, to my knowledge I was the first person to implement non-Fourier heat conduction in a particle based system in a moving heat source system. Furthermore the framework I developed is one of the most, if not the most, universal models for simulating heat conduction problems because of its general heat conduction model,

implemented in a particle framework, utilizing a time stepping algorithm that can be easily applied to Multiphysics problems.

Chapter 9: Bibliography

[1] D. Rosenthal, "Mathematical Theory of Heat Distribution in Welding and cutgting," *Welding Journal,* vol. 20, no. 5, pp. 220-234, 1941.

[2] D. Rosenthal, "The Theory of Moving Source of Heat and its Applciation to Metal Treatments," *Trans. of the American Society of Mechanical Engineers,* vol. 68, pp. 849-865, 1946.

[3] J. Goldak, A. Chakravarti and M. Bibby, "A New Finite Element Model for Welding Heat Sources," *Metallurgical Transactions,* vol. 15B, pp. 299-305, 1994.

[4] J. Ascough, "A Single Step Finite Element Analysis of the Temperature Distribution around a Moving Laser Heat Source," *Optics and Laser in Engineering,* vol. 6, pp. 137-143, 1985.

[5] J. J. Monaghan, "An Introduction to SPH," *Computational Physics Communications,* vol. 48, pp. 89-96, 1988.

[6] K. Koshizuka and Y. Oka, "Moving-Particle Semi-Implicit Method for Fragmentation of Incompressible Fluid," *Nuclear Science and Engineering,* vol. 123, pp. 421-434, 16 February 1996.

[7] P. A. Cundall and O. D. L. Strack, "A discrete numerical model for granular assemblies," *Geotechnique,* vol. 29, no. 1, pp. 47-65, 1979.

[8] X.-T. Pham, "Two-Dimensional Rosenthal Moving Heat Source Analysis using the Meshless Element Freek Galerkin Method," *Numerical Heat Transfer,* vol. 63, pp. 807-823, 2013.

[9] E. O. Resendiz-Flores and F. R. Saucedo-Zendejo, "Two-Dimensional numerical simulation of heat transfer with moving heat source in welding using the Finite Pointset Method," *International Journal of Heat and Mass Transfer,* vol. 90, pp. 239-245, 2015.

[10] J. Crank and P. Nicolson, "A practical method for numerical evaluation of solutiosn of partial differential equations of the heat conduction type," *Proc. Camb. Phil. Soc.,* vol. 43, no. 1, pp. 50-67, 1947.

[11] T. J. R. Hughes, The Finite Element Method: Linear Static and Dynamic Finite Element Analysis, Englewood Cliffs, New Jersey: Prentice-Hall, 1987.

[12] K. K. Tamma and C. V. Anderson, "Novel Heat Conduction Model for Bridging Different Space and Time Scales," *Physical Review Letters,* 2006.

[13] H. Y. Yoon, S. Koshizuka and Y. Oka, "A particle-gridless hybrid method for incompressible flows," *International Journal for Numerical Methods in Fluids,* vol. 30, pp. 407-424, 1999.

[14] M. S. Robortell, N. Kazuo and L.-Y. Cheng, "Dynamic analizes of elastic structure by using moving particle semi-implicit method (MPS)," in *International Congress of Mechanical Engineering,* Gramado, Brazil, 2009.

[15] M. Shimada, S. Masuri and K. K. Tamma, "A novel design of an isochronous integration [iIntegration] framework for first/second order multidisciplinary transient systems," *Internationl Journal for Numerical Methods in Engineering,* vol. 102, pp. 867-891, 15 July 2014.

[16] M. Shimada, S. Masuri and K. K. Tamma, "Isochronous explicit time integration framework: Illustration to thermal stress problems involving both first- and second-order transient systems," *Journal of Thermal Stresses,* vol. 37, pp. 1066-1079, 2014.

[17] D. D. Joseph and L. Preziosi, "Heat Waves," *Reviews of Modern Physics,* vol. 61, pp. 41-73, January 1989.

[18] K. K. Tamma and X. Zhou, "Macroscale and Microscale thermal transport and thermomechanical interactions: some noteworthy perspectives," *Jounral of Thermal Stresses,* vol. 21, no. 3-4, pp. 405-449, 1998.

[19] T. Xu and Y.-C. Jin, "Improvements for accuracy and stability in a weakly-compressible particle method," *Computers and Fluids,* vol. 137, pp. 1-14, 2016.

[20] T. L. Bergman, A. S. Lavine, F. P. Incropera and D. P. Dewitt, Fundamentals of Heat and Mass Transfer, 7th ed., John Wiley & Sons, 2011.

[21] G. F. Carey and M. Tsai, "Hyperbolic Heat Transfer with Reflection," *Numerical Heat Transfer,* vol. 5, pp. 309-327, 1982.

[22] B. Shen and P. Zhang, "Notable physical anomalies manifested in non-Fourier heat conduction under the dual-phase-lag model," *International Journal of Heat and Mass Transfer,* vol. 51, pp. 1713-1727, 2008.

[23] P.-O. Persson and G. Strang, "A Simple Mesh Geneartor in MATLAB," *SIAM Review,* vol. 46, no. 2, pp. 329-345, 2004.

[24] L. Yangyang, S. Zhongguo, X. Guang and L. Ling, "Numerical models for head conduction and natural convection with symmetry boundary condition based on particle method," *International Journal of Heat and Mass Transfer,* vol. 88, pp. 433-444, 2015.

[25] S. Oterkus, E. Madenci and A. Agwai, "Peridynamic thermal diffusion," *Jounral of Computational Physics,* vol. 265, pp. 71-96, 2014.

[26] T. Xue, K. K. Tamma and X. Zhang, "A consistent Moving Particle System Simulation method: Applications to parabolic/hyperbolic heat conduction type problems," *International Journal of Heat and Mass Transfer,* vol. 101, pp. 365-372, 2016.

[27] D. E. Glass, K. K. Tamma and S. B. Railkar, "Hyperbolic Heat Conduction with Convection Boudnary Conditions and Pulse Heating Effects," *Journal of Thermophysics,* vol. 5, no. 1, 1991.

[28] E. O. Resendiz-Flores and F. R. Saucedo-Zendejo, "Two-dimensional numerical simulation of heat transfer with moving heat source in welding using the Finite Pointset Method," *Internatioanl Journal of Heat and Mass Transfer,* vol. 90, pp. 239-245, 2015.

[29] A. Shakibaeinia and Y.-C. Jin, "A weakly compressible MPS method for modeling of open-boundary free-surface flows," *INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS,* vol. 63, pp. 1208-1232, 7 August 2009.

[30] M. S. Song, S. Koshizuka and Y. Oka, "Dynamic analysis of elastic solids by MPS method," *International conference on global environment and advanced nuclear power plants,* p. 1675, 2003.

[31] R. A. Gingold and J. J. Monaghan, "Smoothed particle hydrodynamics - Theory and application to non-spherical stars," *Monthly Notices of the Royal Astronomical Society,* vol. 181, pp. 375-389, November 1977.

[32] B. Ataie-Ashtiani and L. Farhadi, "A stable moving-particle semi-implicit method for free surface flows," *Fluid Dynamics Research,* vol. 38, pp. 241-256, 2006.

[33] C. R. Cattaneo, "Sur une forme de l'équation de la chaleur éliminant le paradoxe d'une propagation instantanée," *Comptes Rendus,* vol. 247, 1958.

[34] V. M. Wheeler, S. Masuri, K. K. Tamma, X. Zhou and M. Sellier, "On the applicability of an Isochronous Integration Framework for Parabolic/Hyperbolic Heat Conduction Type Problems," *Numerical Heat Transfer,* vol. 62, pp. 372-392, 2012.

[35] Y. Chikazawa, S. Koshizuka and Y. Oka, "A particle method for elastic and visco-plastic structures and fluid-structure interaction," *Computational Mechanics,* vol. 27, pp. 97-106, 2001.

[36] E. O. Resendiz-Flores and F. R. Saucedo-Zendejo, "Two-dimensional numerical simulation of heat transfer with moving heat source in welding using the Finite Pointset Method," *Internatioanl Journal of Heat and Mass Transfer,* vol. 90, pp. 239-245, 2015.

[37] X. Zhou, K. K. Tamma and C. V. Anderson, "On a New C- and F- Processes heat conduction monstitutive Model and the associated genearlized theory of dynamic thermoelasticity," *Journal of Thermal Stresses,* vol. 24, no. 6, pp. 531-564, 2001.

Appendix A: MATLAB Code

Chapter 5 Code

```
%% Admin.
clc; clear all; close all;

%% Time Setup
tStart = 0;
Tol=1e-4;
dt = 0.00025
totTime = 50*1e-4;
nSteps = round(totTime/dt)-1;
tCurr = tStart;

%% Material Properties
rho = 1; cp = 1; k = 10;
tau = 1;

%% Meshing
        dx = 0.0025;
        xLen = 1.0;
        nDivX = round(xLen/dx);

        yLen = 1*dx;
        nDivY = round(yLen/dx);

        Rd = 1.05*dx;

        [p0,t] = mesh_regul_dave(xLen,yLen,nDivX,nDivY); %t is
unncessary
        thick = 0.001; %z thickness [m]
        numP = length(p0(:,1));

        vol = (xLen*yLen*thick)/numP; %particle volume
```

```
%% Simulation Information
nDim = 2;

Ft = 1; %Fourier Model
% Ft = 0; %Cattaneo Model
% Ft = 0.5; %somewhere in between

Wij0 = weightfunction_2d(p0,Rd);
[Ks, Kst] = Shapefun(p0,Wij0,vol);
[K1, K2, K3, K4, K5] = Kmtr(Kst,p0,Wij0,vol);

for i=1:numP
    n(i)=sum(Wij0(i,:));
end

n0=max(n);
[Kd1 Kd2] = Div(Kst,p0,Wij0,vol);

%% GS4 - 2, altered to be GS41
%for U0 set rhoSMin to rhoS for GS4-1
rhos=0;rhomin=0;rhomax=1;
sw=0; %1==Vo  0= Uo
[w1v1,w2v2,w3v3,w1v4,w2v5,w1v6,L1,L2,L3,L4,L5]=GS4_parameters(rhos,rhom
in,rhomax,sw);

%% Matrix Initilization
Ti = 0; %Initial temperature distribution
T = Ti*ones(numP,1); Tdot = zeros(numP,1);

qFx = k*Ft*K1*T; qFy = k*Ft*K2*T;
qFxDot = zeros(numP,1); qFyDot = zeros(numP,1);

qCx = k*(1-Ft)*K1*T; qCy = k*(1-Ft)*K2*T;
qCxDot = zeros(numP,1); qCyDot = zeros(numP,1);

xDot = [Tdot;qFxDot;qFyDot;qCxDot;qCyDot];
x = [T;qFx;qFy;qCx;qCy];

zeroNbyN = zeros(numP);
I = eye(numP);

%for stationary sources
massMatrix = [(rho*cp)*I,zeroNbyN,zeroNbyN,zeroNbyN,zeroNbyN;
    zeroNbyN, zeroNbyN, zeroNbyN, zeroNbyN, zeroNbyN;
    zeroNbyN, zeroNbyN, zeroNbyN, zeroNbyN, zeroNbyN;
    zeroNbyN, zeroNbyN, zeroNbyN, tau*I, zeroNbyN;
    zeroNbyN, zeroNbyN, zeroNbyN, zeroNbyN, tau*I];

stiffMatrix = [zeroNbyN, Kd1, Kd2, Kd1, Kd2;
    k*Ft*K1, I, zeroNbyN zeroNbyN, zeroNbyN;
    k*Ft*K2, zeroNbyN, I, zeroNbyN, zeroNbyN;
    k*(1-Ft)*K1, zeroNbyN, zeroNbyN, I, zeroNbyN;
```

```
    k*(1-Ft)*K2, zeroNbyN, zeroNbyN, zeroNbyN, I];

fn = zeros(length(stiffMatrix),1);

%% Boundary Conditions
TBC = 100;
index = 1;
for i = 1:numP
    if(p0(i,1) == 0)
        nodeBC(index,1) = i;
        index = index+1;
    end
end

EBC = TBC*ones(length(nodeBC),1);
fn = zeros(length(stiffMatrix),1);
fnP1 = fn;

for i = 1:length(nodeBC)
    x(nodeBC(i),1) = EBC(i);
end

%% Applying initial q values:
qFx = k*Ft*K1*T; qFy = k*Ft*K2*T;
qCx = k*(1-Ft)*K1*T; qCy = k*(1-Ft)*K2*T;

xDot = [Tdot;qFxDot;qFyDot;qCxDot;qCyDot];
x(numP+1:end) = [qFx;qFy;qCx;qCy];

%% Simulation Loop
tempSave{1,1} = x(1:numP); %For initial
qFxSave{1,1} = x(numP + 1: 2*numP);
qFySave{1,1} = x(2*numP + 1:3*numP);
xSave{1,1} = x;
xDotSave{1,1} = xDot;
tMax(1,1) = max(tempSave{1,1});
tMin(1,1) = min(tempSave{1,1});
tAvg(1,1) = mean(tempSave{1,1});

for simStep = 1:nSteps

    %-----Updating time step-----%
    tCurr = tCurr+dt

    jac = massMatrix*w1v6 + w2v5*stiffMatrix*dt;
    RR1 = -massMatrix*xDot-stiffMatrix*(x +
w1v4*dt*xDot)+fn+w1v1*(fnP1-fn);

    xDotP = xDot;
    xP = x + w1v1*xDot*dt;
    fP = fn +w1v1*(fnP1-fn);
```

```
    %-----Iteration-----%
    m=1;
    R=1;
    while R > Tol

        R1 = massMatrix*xDotP+stiffMatrix*xP-fP;
        RR = -R1;

        %-----Alg Residual-----%
        for i = 1:length(nodeBC)
            s = nodeBC(i);
            RR = RR - EBC(i)*jac(:,s);

            RR(s,1) = 0;
            jac(s,:) = 0; jac(:,s) = 0;
            jac(s,s) = 1;
        end

        delXDot = jac\RR;
        xDotP = xDotP + w1v6*delXDot;
        xP = xP + w2v5*delXDot*dt;


        R=norm(RR,2)
        if R<=Tol
            break;
        end
        iteration = m;
        m = m +1;
    end

    delXDot1 = (xDotP-xDot)/w1v6;

    x = x + L4*xDot*dt + L5*delXDot1*dt;
    xDot = xDot + delXDot1;

    tempSave{simStep+1,1} = x(1:numP);
    qFxSave{simStep+1,1} = x(numP + 1: 2*numP);
    qFySave{simStep+1,1} = x(2*numP + 1:3*numP);
    xSave{simStep+1,1} = x;
    xDotSave{simStep+1,1} = xDot;

end
```

Chapter 6 Code

```
%% Admin.
clc; clear all; close all;

%% Time Setup
tStart = 0;
Tol=1e-4;
dt = 0.0001/2
totTime = 0.005;
nSteps = round(totTime/dt)-1;
tCurr = tStart;

%% Material Properties
rho = 1; cp = 1; k = 1;
tau = 1;

%% Meshing
dx = 0.001;
xLen = 250*dx; yLen = 1*dx;
nDivX = xLen/dx; nDivY = yLen/dx;

Rd = 1.01*min((xLen/nDivX),yLen/nDivY);
[p0,t] = mesh_regul_dave(xLen,yLen,nDivX,nDivY); %t is unncessary
thick = 0.01; %z thickness [m]
numP = length(p0(:,1));

vol = (xLen*yLen*thick)/numP; %particle volume

%% Simulation Information
nDim = 2;

Ft = 1; %Fourier Model
% Ft = 0; %Cattaneo Model
% Ft = 0.5; %somewhere in between

Wij0 = weightfunction_2d(p0,Rd);
[Ks, Kst] = Shapefun(p0,Wij0,vol);
[K1, K2, K3, K4, K5] = Kmtr(Kst,p0,Wij0,vol);

for i=1:numP
    n(i)=sum(Wij0(i,:));
end

n0=max(n);
[Kd1 Kd2] = Div(Kst,p0,Wij0,vol);

%% GS4 - 2, altered to be GS41
%for U0 set rhoSMin to rhoS for GS4-1
rhos=0;rhomin=0;rhomax=1;
sw=0; %1==Vo  0= Uo
```

```
[w1v1,w2v2,w3v3,w1v4,w2v5,w1v6,L1,L2,L3,L4,L5]=GS4_parameters(rhos,rhom
in,rhomax,sw);

%% Matrix Initilization
Ti = 0; %Initial temperature distribution
T = Ti*ones(numP,1); Tdot = zeros(numP,1);

qFx = k*Ft*K1*T; qFy = k*Ft*K2*T;
qFxDot = zeros(numP,1); qFyDot = zeros(numP,1);

qCx = k*(1-Ft)*K1*T; qCy = k*(1-Ft)*K2*T;
qCxDot = zeros(numP,1); qCyDot = zeros(numP,1);

xDot = [Tdot;qFxDot;qFyDot;qCxDot;qCyDot];
x = [T;qFx;qFy;qCx;qCy];

zeroNbyN = zeros(numP);
I = eye(numP);

%for stationary sources
massMatrix = [(rho*cp)*I,zeroNbyN,zeroNbyN,zeroNbyN,zeroNbyN;
    zeroNbyN, zeroNbyN, zeroNbyN, zeroNbyN, zeroNbyN;
    zeroNbyN, zeroNbyN, zeroNbyN, zeroNbyN, zeroNbyN;
    zeroNbyN, zeroNbyN, zeroNbyN, tau*I, zeroNbyN;
    zeroNbyN, zeroNbyN, zeroNbyN, zeroNbyN, tau*I];

stiffMatrix = [zeroNbyN, Kd1, Kd2, Kd1, Kd2;
    k*Ft*K1, I, zeroNbyN zeroNbyN, zeroNbyN;
    k*Ft*K2, zeroNbyN, I, zeroNbyN, zeroNbyN;
    k*(1-Ft)*K1, zeroNbyN, zeroNbyN, I, zeroNbyN;
    k*(1-Ft)*K2, zeroNbyN, zeroNbyN, zeroNbyN, I];

fn = zeros(length(stiffMatrix),1);
%% Boundary Conditions
        Tx0 = 0;
        qBC = 0;
        index = 1;

        coeff = 100;

        Q = coeff*1*rho*cp/(dx/2*thick);
        fn = zeros(length(stiffMatrix),1);

        for i = 1:numP

            if(  p0(i,1) == xLen)
                x(numP + i) = 0;
                nodeBC(index,1) = numP + i;
                EBC(index) = 0;
                index = index+1;
            end
```

```
            if(p0(i,1) == 0.0)
                fn(i,1) = Q;
            end
        end

        fnP1 = fn;

%% Applying initial q values:
qFx = k*Ft*K1*T; qFy = k*Ft*K2*T;
qCx = k*(1-Ft)*K1*T; qCy = k*(1-Ft)*K2*T;

xDot = [Tdot;qFxDot;qFyDot;qCxDot;qCyDot];
x(numP+1:end) = [qFx;qFy;qCx;qCy];

%% Simulation Loop
tempSave{1,1} = x(1:numP); %For initial
qFxSave{1,1} = x(numP + 1: 2*numP);
qFySave{1,1} = x(2*numP + 1:3*numP);
xSave{1,1} = x;
xDotSave{1,1} = xDot;
tMax(1,1) = max(tempSave{1,1});
tMin(1,1) = min(tempSave{1,1});
tAvg(1,1) = mean(tempSave{1,1});

for simStep = 1:nSteps

    if( simStep > 1)
        fn = 0*fn;
        fnP1 = 0*fn;
    end

    %-----Updating time step-----%
    tCurr = tCurr+dt

    jac = massMatrix*w1v6 + w2v5*stiffMatrix*dt;
    RR1 = -massMatrix*xDot-stiffMatrix*(x +
w1v4*dt*xDot)+fn+w1v1*(fnP1-fn);

    xDotP = xDot;
    xP = x + w1v1*xDot*dt;
    fP = fn +w1v1*(fnP1-fn);

    %-----Iteration-----%
    m=1;
    R=1;
    while R > Tol

        R1 = massMatrix*xDotP+stiffMatrix*xP-fP;
        RR = -R1;

        %-----Alg Residual-----%
        for i = 1:length(nodeBC)
```

```
            s = nodeBC(i);
            RR = RR - EBC(i)*jac(:,s);

            RR(s,1) = 0;
            jac(s,:) = 0; jac(:,s) = 0;
            jac(s,s) = 1;
        end

        delXDot = jac\RR;
        xDotP = xDotP + w1v6*delXDot;
        xP = xP + w2v5*delXDot*dt;


        R=norm(RR,2)
        if R<=Tol
            break;
        end
        iteration = m;
        m = m +1;
    end

    delXDot1 = (xDotP-xDot)/w1v6;

    x = x + L4*xDot*dt + L5*delXDot1*dt;
    xDot = xDot + delXDot1;

    tempSave{simStep+1,1} = x(1:numP);
    qFxSave{simStep+1,1} = x(numP + 1: 2*numP);
    qFySave{simStep+1,1} = x(2*numP + 1:3*numP);
    xSave{simStep+1,1} = x;
    xDotSave{simStep+1,1} = xDot;
end
```

Chapter 7 Code

```
%% Admin.
clc; clear all; close all;

%% Time Setup
tStart = 0;
Tol=1e-2;
% Tol=1e-3;
dt = 0.002;
totTime = 0.1;
nSteps = round(totTime/dt)-1;
tCurr = tStart;

%% Material Properties
tau = 10; %rate of propagation
rho = 100; %Density [kg/m^3];
cp = 100; %specific heat [J/(kg K)]
k = 1; %thermal conductivity [W/(m*k)]

waveSpeed = sqrt(k/(rho*cp*tau));

%% Meshing
xLen = 0.05;
nDivX = 50;
dx = xLen/nDivX;

yLen = 0.025;
nDivY = yLen/dx;

Rd = 1.015*min((xLen/nDivX),yLen/nDivY);

RdS  = 1.065*min((xLen/nDivX),yLen/nDivY);

[p0,t] = mesh_regul_dave(xLen,yLen,nDivX,nDivY); %t is unncessary
thick = 0.001; %z thickness [m]
numP = length(p0(:,1));

vol = (xLen*yLen*thick)/numP; %particle volume

for as = 1:length(p0(:,1))
    txtNumber(as,1)=as;
end


%% Simulation Information
nDim = 2;

Ft = 0;
```

```
Wij0 = weightfunction_2d(p0,Rd);
Wsmooth = weightfunction_2d(p0,RdS);
[Ks, Kst] = Shapefun(p0,Wij0,vol);

[K1, K2, K3, K4, K5] = Kmtr(Kst,p0,Wij0,vol);

for i=1:numP
    n(i)=sum(Wij0(i,:));
end

n0=max(n);
[Kd1 Kd2] = Div(Kst,p0,Wij0,vol);

%% GS4 - 2, altered to be GS41

%for U0 set rhoSMin to rhoS for GS4-1
rhos=0;rhomin=0;rhomax=1;
eta1 =0;
eta2= eta1;
eta3 = 1;
sw=0; %1==Vo  0= Uo
[w1v1,w2v2,w3v3,w1v4,w2v5,w1v6,L1,L2,L3,L4,L5]=GS4_parameters_EXP(rhos,
rhomin,rhomax,sw,eta1,eta2,eta3);

%% Moving Source (Part 1)
velHeat = 15*[-0.01,0];
pHeat1 = [xLen*0.5, yLen/2];
pHeat2 = pHeat1  + velHeat*dt;

%% Matrix Initilization
Ti = 0; %Initial temperature distribution
T = Ti*ones(numP,1); Tdot = zeros(numP,1);

qFx = k*(Ft)*K1*T; qFy = k*(Ft)*K2*T;
qFxDot = zeros(numP,1); qFyDot = zeros(numP,1);
qCx = k*(1-Ft)*K1*T; qCy = k*(1-Ft)*K2*T;
qCxDot = zeros(numP,1); qCyDot = zeros(numP,1);

xDot = [Tdot; qFxDot; qFyDot; qCxDot; qCyDot];
x = [T; qFx; qFy; qCx; qCy];

zeroNbyN = zeros(numP);
I = eye(numP);

%for stationary sources
massMatrix = [(rho*cp)*I,zeroNbyN,zeroNbyN,zeroNbyN,zeroNbyN;
    zeroNbyN, zeroNbyN, zeroNbyN, zeroNbyN, zeroNbyN;
    zeroNbyN, zeroNbyN, zeroNbyN, zeroNbyN, zeroNbyN;
    zeroNbyN, zeroNbyN, zeroNbyN, tau*I, zeroNbyN;
    zeroNbyN, zeroNbyN, zeroNbyN, zeroNbyN, tau*I];

stiffMatrix = [1*velHeat(1,1)*K1+1*velHeat(1,2)*K2, Kd1, Kd2, Kd1, Kd2;
```

```
    k*Ft*K1, I, zeroNbyN zeroNbyN, zeroNbyN;
    k*Ft*K2, zeroNbyN, I, zeroNbyN, zeroNbyN;
    k*(1-Ft)*K1, zeroNbyN, zeroNbyN,
tau*(1*velHeat(1,1)*K1+1*velHeat(1,2)*K2), zeroNbyN;
    k*(1-Ft)*K2, zeroNbyN, zeroNbyN, zeroNbyN,
tau*(1*velHeat(1,1)*K1+1*velHeat(1,2)*K2)];

fn = zeros(length(stiffMatrix),1);
fBC = fn;
%% Boundary Conditions
qx0 = 0;
index = 1;
fn = zeros(length(stiffMatrix),1);

Tx0 = 0;

for i = 1:numP

    if(  p0(i,1) == 0)
        x(i,1) = Tx0;
        nodeBC(index,1) = i;
        EBC(index,1) = Tx0;
        index = index+1;
    end

    if (p0(i,2) == 0 || p0(i,2) == yLen)
        x(i,1) = Tx0;
        nodeBC(index,1) = i;
        EBC(index,1) = Tx0;
        index = index+1;
    end


    if (p0(i,1) == xLen)
        x(i,1) = Tx0;
        nodeBC(index,1) = i;
        EBC(index,1) = Tx0;
        index = index+1;
    end

end


for i = 1:numP

    if(p0(i,2) == 0 || p0(i,1) == xLen || p0(i,2) == yLen)
        fBC(i,1) = qx0/dx;
    end
end

fBC = 0*fBC;
fn = fn + fBC;
```

```
fnP1 = fn;
fnSave = fn;


%% Applying Source Term
rHeat = 0.003;
QHeat = (1e6);
Wn1 = sourceWeight(pHeat1,p0,rHeat);
WnP1 = sourceWeight(pHeat2,p0,rHeat);
fn(1:numP) = QHeat*Wn1;
fnP1(1:numP) = QHeat*WnP1;

fn = fn+fBC;
fnN1 = fn;
fnP1 = fnP1 + fBC;

%% Simulation Loop
for simStep = 1:nSteps

    fn = fn + (1-Ft)*tau*(fnP1-fn)/dt;

    %-----Updating time step-----%
    tCurr = tCurr+dt
    jac = massMatrix*w1v6 + w2v5*stiffMatrix*dt;
    RR1 = -massMatrix*xDot-stiffMatrix*(x +
w1v4*dt*xDot)+fn+w1v1*(fnP1-fn);

    xDotP = xDot;
    xP = x + w1v4*xDot*dt;
    fP = fn +w1v1*(fnP1-fn);

    %-----Iteration-----%
    m=1;
    R=1;
    while R > Tol

        R1 = massMatrix*xDotP+stiffMatrix*xP-fP;
        RR = -R1;

        %-----Alg Residual-----%
        for i = 1:length(nodeBC)
            s = nodeBC(i);
            RR = RR - EBC(i)*jac(:,s);

            RR(s,1) = 0;
            jac(s,:) = 0; jac(:,s) = 0;
            jac(s,s) = 1;
        end

        delXDot = jac\RR;
        xDotP = xDotP + w1v6*delXDot;
```

```
        xP = xP + w2v5*delXDot*dt;


        R=norm(RR,2)

        if(sum(isnan(RR)) > 0)
            return
        end

        if R<=Tol
            break;
        end
        iteration = m;
        m = m +1;
    end

    delXDot1 = (xDotP-xDot)/w1v6;

    x = x + L4*xDot*dt + L5*delXDot1*dt;
    xDot = xDot + delXDot1;

    pHeat1 = pHeat2;
    pHeat2 = pHeat1 + velHeat*dt;

    Wn1 = sourceWeight(pHeat1,p0,rHeat);
    WnP1 = sourceWeight(pHeat2,p0,rHeat);

    fnN1 = fn;

    fn(1:numP) = QHeat*Wn1;
    fnP1(1:numP) = QHeat*WnP1;

    fn = fn+fBC;
    fnP1 = fnP1 + fBC;

end
```