5-12-2001

A Numerical Study of the Conjugate Conduction-Convection Heat Transfer Problem

Robert Samuel Webster

Recommended Citation

Webster, Robert Samuel, "A Numerical Study of the Conjugate Conduction-Convection Heat Transfer Problem" (2001). *Theses and Dissertations*. 217.
https://scholarsjunction.msstate.edu/td/217

A NUMERICAL STUDY OF THE CONJUGATE CONDUCTION–CONVECTION HEAT TRANSFER PROBLEM

By

Robert Samuel Webster

A Dissertation
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy
in Engineering
in the Department of Aerospace Engineering

Mississippi State, Mississippi

May 2001

A NUMERICAL STUDY OF THE CONJUGATE CONDUCTION–CONVECTION HEAT TRANSFER PROBLEM

By

Robert Samuel Webster

Approved:

David L. Whitfield
Professor of Aerospace Engineering
(Major Professor)

J. Mark Janus
Associate Professor of Aerospace Engineering
(Director of Dissertation)

W. Roger Briley
Professor of Mechanical Engineering
(Member of Committee)

Jianping Zhu
Professor of Mathematics
(Minor Professor)

Pasquale Cinnella
Associate Professor of Aerospace Engineering
(Member of Committee)

John C. McWhorter, III
Professor and Head of Aerospace Engineering

A. Wayne Bennett
Dean of the College of Engineering

Name: Robert Samuel Webster

Date of Degree: May 12, 2001

Institution: Mississippi State University

Major Field: Engineering (Aerospace Engineering)

Dissertation Director: Dr. J. Mark Janus

Major Professor: Dr. David L. Whitfield

Title of Study: A NUMERICAL STUDY OF THE CONJUGATE CONDUCTION–CONVECTION HEAT TRANSFER PROBLEM

Pages in Study: 107

Candidate for Degree of Doctor of Philosophy

This study investigates some of the basic aspects of conjugate, or coupled, heat transfer problems. The ultimate interest is in the improvement of an existing computational fluid dynamics (CFD) code by the inclusion of such a coupling capability. Many CFD codes in the past have treated the thermal boundary conditions of a bounding solid as the simple cases of either a surface across which there is no heat flux, or as a surface along which the temperature is a constant with respect to both space and time. These conditions are acceptable for some applications, but many real–world problems require a more realistic treatment of the thermal wall condition.

A thermal coupling may be accomplished by maintaining a continuous heat flux and temperature across the fluid–solid boundary. A heat flux is calculated on the fluid–side of the interface, and this is used as a boundary condition for a heat–conduction solver to calculate the temperature field within the solid and return an interface temperature to the fluid. This process is executed for each time–step iteration of the code, and, therefore, the temperature field of the solid and the fluid–solid interface temperature are allowed to evolve with time and space.

A new heat–conduction solver is developed and coupled with an existing flow solver. For this reason, some of the study is devoted to the testing of the accuracy of the new heat–conduction solver on simple problems for which there exist analytical solutions. Additional coverage is devoted to the possibility of thermal communication between solid grid blocks. This is due to the fact that multiple grid blocking of the solid may be required for more complex geometries. For such cases, a similar procedure as that described for the fluid–solid interface is used to accomplish the solid–solid block–to–block communication.

Relatively simple test cases of fluid–solid and solid–solid coupling are conducted; these cases are limited to two–dimensional grids. Other limitations include: the assumption of constant thermophysical properties for the solid, no consideration for thermal expansion of the solid, and no consideration for the radiation mode of heat transfer. The results indicate that the heat–conduction/flow solver shows potential.

DEDICATION

To my Mother, Robbie W. Webster, and my Father, the late James S. Webster, Jr.

ACKNOWLEDGMENTS

I would like to express my gratitude to my major professor, Dr. David Whitfield, and my dissertation director, Dr. Mark Janus for their patience and help over the years. Thanks very much for giving me the opportunity to work at a great facility, among lots of great people. I would like to also thank Drs. Roger Briley, Pasquale Cinnella, and Jianping Zhu for serving on my committee. I greatly appreciate the help and insight that you have provided, through both discussion and class lectures. I feel very fortunate to have had you as professors; your classes were interesting, and I always felt that you had a genuine concern for the student. Thanks to Dr. John McWhorter of the aerospace engineering department for having me as a student, and to Ms. Alicia Peters for her help with keeping all my forms straight. A special thanks, also, to Mrs. Nita Hartness for being a helpful secretary and a very good friend.

Others that I would like to thank for both their help and friendship include Drs. Jen–Ping Chen, Abi Arabshahi, Tim Swafford, Lafe Taylor, Boyd Gatlin, Kidambi “Sree” Sreenivas, and Ramesh Pankajakshan. I’ve had the good fortune to make many friends while being here; so many that I’m not even going to attempt a listing. You all have given me the greatest gift that I think can be given, and that is your friendship.

I give my thanks to Messrs. Victor Whitehead and Pete Huseman from Lockheed–Martin Astronautics for the financial support and for providing very interesting and challenging problems on which to work.

Finally, I thank my family for all they’ve done for me. My parents have provided me with all I’ve ever needed, and they have taught me the things I need to know. I thank my sister and her family for their help over the past several years. Special thanks to my Aunt Elaine for all of her help and support. And a special thanks also to my cousin Eric and his wife Sarah for putting up with me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

FIGURE Page

FIGURE Page

FIGURE Page

NOMENCLATURE

a	speed of sound
A,B,C	flux Jacobian matrices; matrix coefficients for heat–conduction solver
$A\left(\beta_m,\nu_n,t\right)$	transformed boundary condition (analytical solution to heat equation)
c	specific heat (for solid material)
c_p	specific heat (at constant pressure)
c_v	specific heat (at constant volume)
D	diagonal matrix
e_i	internal energy
e_t	total energy
f,g,h	flux vectors (dimensional)
F,G,H	flux vectors in curvilinear transformation
$\overline{F}(\beta_m, \nu_n)$	transformed initial condition (analytical solution to heat equation)
i,j,k	indices corresponding to transformed coordinates
I	identity matrix
$\tilde{I}$	identity tensor
J	Jacobian of coordinate transformation
$K(\beta_m, x)$	transform kernel with β–eigenvalue (analytical solution to heat equation)
$K\left(\nu_n, y\right)$	transform kernel with ν–eigenvalue (analytical solution to heat equation)
L,U	lower–diagonal matrix, upper diagonal matrix
M	Mach number

p	pressure
P_r	Prandtl number
q	cartesian conservative–variable vector
$\vec{q}$	heat flux vector
Q	transformed coordinate conservative–variable vector
R	gas constant, residual vector
R_α	ratio of thermal diffusivities across solid–solid block interface
R_e	Reynolds number
R_κ	ratio of thermal conductivities across solid–solid block interface
LHS	left–hand side of equation
RHS	right–hand side of equation
t	time
T	temperature
$\tilde{T}$	stress tensor
u,v,w	cartesian components of velocity vector
U,V,W	contravariant velocity components
$\vec{V}$	velocity vector
x,y,z	cartesian coordinates
α	thermal diffusivity
δ	difference operator
γ	ratio of specific heats
κ	thermal conductivity
Δ	change in quantity between time steps
ξ,η,ζ	curvilinear coordinates
λ	second coefficient of viscosity (assumed $-2/3\mu$ by Stokes' hypothesis)

μ	coefficient of viscosity
ρ	density (fluid and solid)
τ	non–dimensional time, components of stress tensor

Superscripts:

^	dimensional quantity
−	general vector in Newton's method
k	Newton iteration parameter
–1	inverse of matrix
n	time step parameter
+/–	indicator of upwind direction
T	transpose of matrix

Subscripts:

i,j,k	indices of cell centers
(i,j,k) ± 1/2	indices of cell faces
l	laminar property
ref	reference condition
t	differentiation with respect to time, turbulent property
x,y,z	differentiation with respect to x, y, and z
ξ,η,ζ	differentiation with respect to ξ, η, and ζ
τ	differentiation with respect to τ

Symbols:

∂	partial differentiation
$\vec{\nabla}$	divergence operator
$\cdot$	matrix multiplication operator

CHAPTER I

INTRODUCTION

The technology and application of computational science have advanced greatly over the past fifteen years, the last ten especially. Research into numerical techniques and grid generation and certainly the rapid advancement of computer speed and memory capability have aided this growth of the computational sciences. The fields of computational fluid dynamics, heat transfer and solid mechanics continue to address ever–more challenging problems. Quite often, though, the three areas of fluid dynamics, heat transfer and solid mechanics are treated independently of one another. For many applications this practice may be acceptable, but for many other problems the interaction, or coupling, of these different fields needs to be accounted for. In this work, the coupling of a heat–conduction solver with an existing flow solver will be investigated, the ultimate goal being to expand the area of application of this flow solver.

Most flow solvers now make calculations that take into account the convective transport of mass, momentum and energy and diffusive transport of momentum and energy. The diffusive transport of energy is by heat conduction within the fluid. The typical solid–wall boundary conditions used to help solve the energy equation are either the adiabatic wall condition (i.e., zero heat conduction across a solid boundary) or the condition of a constant–temperature wall.

The existing flow solver is no exception to the above. The main code was developed over ten years ago by Arabshahi [1], and included the very useful capability to handle multi–block, structured grids in a relatively general format. That is to say, there is little or no restriction on the topology of the grids (O–, C–, or H–type may be used) or on the grid surfaces which are used for communicating with other blocks. This code carries the name "UBIFLOW". The numerical aspects of this solver are based on the work of Whitfield [2], and Whitfield, *et al.* [3]. An extensive investigation by Belk [4] into the actual communication concerns across grid block boundaries served as an important reference for [1]. All of these works were for the inviscid Euler equations as the mathematical model. Work by Simpson [5] and Gatlin [6] addressed the inclusion of diffusive effects into the governing equations, which served as a basis for further expanding the capability of [1] .

An additional code development was made by Cox [7] (named "CHEQNS") whose work allowed for the treatment of equilibrium chemical reactions in the flow. A "black–box" chemistry solver was attached to the flow solver of Ref. [1] to accomplish this. By assuming the reactions to be in local chemical equilibrium, the chemistry solver and flow solver could be kept essentially separate. Therefore, the flow solver part of Cox [7] was for the most part the same as that by Arabshahi [1]. The CHEQNS code is the one involved in this work (although no chemically reacting flows will be included).

All of the flow solvers mentioned above, having the capability to include diffusive effects, used simple thermal boundary conditions , such as those mentioned on the previous page. These codes, as well as many others over the years, have not addressed the coupling of heat conduction within solid bodies that bound the flow, because this has been of little concern. However, papers in the technical literature over the last five to ten years show an increasing interest in this coupling of the solid heat conduction with the flow solver. This is often referred to as the conjugate (i.e., coupled) heat transfer problem.

An early paper by Lau, *et al.* [8] looks at the problem of internal fins for the possible application to heat–exchanger problems. Works by Yu, *et al.* [9], and Pozzi and Lupo [10] address the coupling of external flows with conduction in simple bodies such as flat plates, wedges, and cones. A work by Trevino, *et al.* [11] analyzes the forced convection between two counter–flowing streams separated by a conducting plate. Coupled solid conduction and natural or mixed convection are studied by Joubert and Le Quere [12] and Bernier and Baliga [13]. These last two examples are, of course, low–speed flows. A study of conjugate heat transfer in high–speed flow was conducted by Shope [14]; this problem dealt with the cooling of a nozzle wall for a supersonic wind tunnel. A recent study by Janus and Newman [15] looked at the coupling of aerodynamic and thermal effects for an optimization study of turbine airfoil design. A paper by Rahaim, *et al.* [16] provides useful numerical and experimental data for coupled heat transfer in a high–speed flow. A very recent paper by Sondak and Dorney [17] investigates coupled unsteady flow and heat conduction for a turbine stage; this paper points out that the blocks within the conduction grid can be a potential source of numerical problems (a point that will be discussed in the present work). Works by Chang and Payne [18] and Shyy and Burke [19] look more directly at the numerical problems involved at interfaces where an abrupt change in diffusion coefficients occurs. The former investigates averaging of these interface coefficients, while the latter looks at the characteristics of solving a truly coupled convection–diffusion problem by iterative methods. Finally, a recent paper by Ruiz and Black [20] provides a technique for the zonal decomposition of solids and the thermal communication among these solid sub–blocks.

The papers listed above cover a wide range of problem applications and are only a sampling of the work that has been done in this area. One of the most noticeable points demonstrated by these previous works is that the interest in the coupling of conduction heat transfer with flow solvers is ever on the increase.

The objective of the present study is to enhance the present flow solver resulting from the works of Arabshahi and Cox. Aspects of Arabshahi's work will be used in an effort to provide an arbitrary arrangement of grid blocks, both fluid and solid. A number of areas will not be addressed, however: these include the modeling of temperature–dependent and direction–dependent properties within the solid, the possibility of thermal expansion of the solid, and the possibility of radiation heat transfer. These are all potentially very important aspects of the general problem, and will certainly need to be included in the future for more accurate simulations. However, it is felt that these may be neglected at this time in order to gain some basic understanding of the coupled problem.

A portion of the present work involves the code development of a heat–conduction solver. The goal is to then add this solver to the present flow solver and to allow for the two to be used either together or separately by means of defining parameters within the general input file.

The general outline of the dissertation is the following. The second chapter presents the governing equations for both the flow and heat–conduction solvers, which are included as useful reference material. The third chapter discusses the general numerical algorithm and methods. The fourth chapter discusses the possible means of block–to–block communication for both fluid–solid and solid–solid interfaces. Chapter five presents the primary results from this work. Because the heat–conduction solver is new, a portion of these results are for the validation of this code. Finally, chapter six offers a summary and some conclusions stemming from this investigation.

CHAPTER II

GOVERNING EQUATIONS

Flow Solver

A computational fluid dynamics (CFD) or computational heat transfer (CHT) code accomplishes its task by modeling the physics of the problem of interest. The heart of the physics of such problems for a CFD code involves the conservation of three fundamental entities: mass, momentum, and energy. The conservation laws for a compressible, viscous and heat conducting fluid in an unsteady state are expressed mathematically as

$$\frac{\partial \rho}{\partial t} + \vec{\nabla} \cdot \left(\rho \vec{V}\right) = 0 \tag{2.1}$$

$$\frac{\partial\left(\rho \vec{V}\right)}{\partial t} + \vec{\nabla} \cdot \left(\rho \vec{V}\vec{V}\right) = \vec{\nabla} \cdot \tilde{T} \tag{2.2}$$

$$\frac{\partial e_t}{\partial t} + \vec{\nabla} \cdot \left(e_t \vec{V}\right) = -\vec{\nabla} \cdot \vec{q} + \vec{\nabla} \cdot \left(\tilde{T} \cdot \vec{V}\right) \tag{2.3}$$

Equation (2.1) represents the conservation of mass; Equation (2.2) represents the conservation of momentum, and Equation (2.3) represents the conservation of total energy. Each of these equations is applied to a small (ideally, infinitesimal) fluid volume. The above partial differential equations are very general in nature and may be applied to any coordinate system. Also included in these expressions are the assumptions of continuous media with body forces (e.g., gravitational effects) neglected and no volumetric heat generation or heat sources. The left–hand side of each equation represents the sum of the time rate of change of each conserved quantity within a given volume and the convective flux, or flow, of that quantity across the boundaries of the volume. Equations (2.2) and (2.3) have non–zero terms on the right–hand side, which represent the diffusive flux of these quantities across the volume boundaries.

The symbols and physical parameters of the governing equations are defined in the nomenclature. Further (mathematical) definition of the stress tensor and total energy are given below:

$$\tilde{T} = \left[-p + \lambda\left(\vec{\nabla} \cdot \vec{V}\right)\right]\tilde{I} + \mu\left[\vec{\nabla}\vec{V} + \left(\vec{\nabla}\vec{V}\right)^T\right] ,$$

and $$e_t = \rho e_i + \frac{1}{2}\rho\left|\vec{V}\right|^2 .$$

As already mentioned, the governing equations as in their current format may be applied to any coordinate system. If the Cartesian system is preferred, the governing equations become Equations (2.4) through (2.8)

$$\frac{\partial\rho}{\partial t} + \frac{\partial(\rho u)}{\partial x} + \frac{\partial(\rho v)}{\partial y} + \frac{\partial(\rho w)}{\partial z} = 0 , \qquad \text{(mass)} \qquad (2.4)$$

$$\frac{\partial(\rho u)}{\partial t} + \frac{\partial}{\partial x}\left(\rho u^2 + p - \tau_{xx}\right) + \frac{\partial}{\partial y}\left(\rho uv - \tau_{xy}\right) + \frac{\partial}{\partial z}(\rho uw - \tau_{xz}) = 0 , \qquad \text{(x–momentum)} \qquad (2.5)$$

$$\frac{\partial(\rho v)}{\partial t} + \frac{\partial}{\partial x}\left(\rho uv - \tau_{xy}\right) + \frac{\partial}{\partial y}\left(\rho v^2 + p - \tau_{yy}\right) + \frac{\partial}{\partial z}\left(\rho vw - \tau_{yz}\right) = 0 , \qquad \text{(y–momentum)} \qquad (2.6)$$

$$\frac{\partial(\rho w)}{\partial t} + \frac{\partial}{\partial x}(\rho uw - \tau_{xz}) + \frac{\partial}{\partial y}\left(\rho vw - \tau_{yz}\right) + \frac{\partial}{\partial z}\left(\rho w^2 + p - \tau_{zz}\right) = 0 , \qquad \text{(z–momentum)} \qquad (2.7)$$

$$\begin{aligned}\frac{\partial e_t}{\partial t} &+ \frac{\partial}{\partial x}\left[(e_t + p)u - u\tau_{xx} - v\tau_{xy} - w\tau_{xz} + q_x\right] \\ &+ \frac{\partial}{\partial y}\left[(e_t + p)v - u\tau_{xy} - v\tau_{yy} - w\tau_{yz} + q_y\right] \\ &+ \frac{\partial}{\partial z}\left[(e_t + p)w - u\tau_{xz} - v\tau_{yz} - w\tau_{zz} + q_z\right] = 0 . \qquad \text{(energy)} \qquad (2.8)\end{aligned}$$

Using the assumption of $\lambda = -2/3\ \mu$, the shear stress components of the stress tensor may be expressed as Equations (2.9) through (2.14). And further assuming there to be no directional dependence of the thermal conductivity of the fluid, the heat flux terms are given as Equation (2.15)

$$\tau_{xx} = 2/3\mu\left(2\frac{\partial u}{\partial x} - \frac{\partial v}{\partial y} - \frac{\partial w}{\partial z}\right) , \tag{2.9}$$

$$\tau_{yy} = 2/3\mu\left(2\frac{\partial v}{\partial y} - \frac{\partial u}{\partial x} - \frac{\partial w}{\partial z}\right) , \tag{2.10}$$

$$\tau_{zz} = 2/3\mu\left(2\frac{\partial w}{\partial z} - \frac{\partial u}{\partial x} - \frac{\partial v}{\partial y}\right) , \tag{2.11}$$

$$\tau_{xy} = \mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) , \tag{2.12}$$

$$\tau_{xz} = \mu\left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right) , \tag{2.13}$$

$$\tau_{yz} = \mu\left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\right) . \tag{2.14}$$

$$q_x = -\kappa\frac{\partial T}{\partial x}, \qquad q_y = -\kappa\frac{\partial T}{\partial y}, \text{ and } \quad q_z = -\kappa\frac{\partial T}{\partial z} . \tag{2.15}$$

Equations (2.1) through (2.3), or (2.4) through (2.8) are typically referred to as the Navier–Stokes equations and are considered the governing equations for time–dependent, compressible, viscous flow. Using these equations, the unknown variables $\rho,\ \rho u,\ \rho v,\ \rho w,$ and e_t are determined. The additional thermodynamic variables, pressure and temperature, may then be found by means of the equation of state and the relation of pressure to internal energy, namely

$$\begin{aligned} p &= \rho RT && \text{(ideal gas assumption)} \\ \text{and}\quad p &= (\gamma - 1)\rho e_i && \text{(from } e_i = c_v\ T\text{).} \end{aligned} \tag{2.16}$$

Governing Equations in Vector Form

A compact form of Eqs. (2.4) – (2.8) is given by

$$\frac{\partial q}{\partial t} + \frac{\partial f}{\partial x} + \frac{\partial g}{\partial y} + \frac{\partial h}{\partial z} = 0. \tag{2.17}$$

In this sense, the unknown conservative variables (i.e., ρ, ρu, ρv, ρw, and e_t) are combined into one unknown entity, the "q" vector (not to be confused with the heat flux vector). This vector is defined as

$$q = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ e_t \end{bmatrix} . \tag{2.18}$$

The terms f, g, and h are referred to as the flux vectors, and, again, these terms represent the transport, by both convective and diffusive means, of mass, momentum, and energy across the bounding surfaces of a given control volume. These flux vectors are themselves functions of the q–vector, i.e., f = f(q), g = g(q), and h = h(q). The flux vectors are given in Equations (2.19) through (2.21) on the following page

Scaling of the Governing Equations

Until now, the governing equations have been assumed to be in terms of dimensional variables. A step that is often necessary for computations of finite precision is that of scaling these variables to some non–dimensional form. This process allows the variables to be expressed in roughly the same order of magnitude, where large differences in magnitude may be present when in dimensional format.

The scaled format is also a very general representation in which the entire problem is defined in terms of a few non–dimensional parameters. For example, the convective terms are linked to the Mach Number, the diffusive terms to the Reynolds Number (the viscous shear terms) and the Prandtl Number (the heat conduction terms). Spatial terms

$$f = \begin{bmatrix} \rho u \\ \rho u^2 + p - \tau_{xx} \\ \rho uv - \tau_{xy} \\ \rho uw - \tau_{xz} \\ (e_t + p)u - u\tau_{xx} - v\tau_{xy} - w\tau_{xz} + q_x \end{bmatrix}, \tag{2.19}$$

$$g = \begin{bmatrix} \rho v \\ \rho uv - \tau_{xy} \\ \rho v^2 + p - \tau_{yy} \\ \rho vw - \tau_{yz} \\ (e_t + p)v - u\tau_{yx} - v\tau_{yy} - w\tau_{yz} + q_y \end{bmatrix}, \tag{2.20}$$

$$h = \begin{bmatrix} \rho w \\ \rho uw - \tau_{zx} \\ \rho vw - \tau_{zy} \\ \rho w^2 + p - \tau_{zz} \\ (e_t + p)w - u\tau_{zx} - v\tau_{zy} - w\tau_{zz} + q_z \end{bmatrix}. \tag{2.21}$$

are associated with a so–called characteristic length, and the thermodynamic aspect to the ratio of specific heats, γ. Most of the flow solvers in use at MSU have been scaled in this manner (e.g., see Janus [21], Arabshahi [1], and Chen [22]). Mach Number, Reynolds Number, Prandtl Number, and γ are all input, and the length is accounted for in the size of the grid (i.e., the grid is in terms of the characteristic length). This is satisfactory so long as the gas in question is of a constant and uniform makeup. The solver used in this work has been modified by Cox [7] so as to allow for flows in which chemical (equilibrium) reactions are involved. In this type of flow, γ will change with the chemical makeup; the flow variables can no longer be scaled by constant, free–stream values of the Mach, Reynolds, and Prandtl Numbers. Cox's method of scaling is similar to the "conventional" method (see Arabshahi [1]), with a few exceptions, and is shown on the following page.

$$x = \frac{\hat{x}}{L_{ref}}, \quad y = \frac{\hat{y}}{L_{ref}}, \quad z = \frac{\hat{z}}{L_{ref}},$$

$$u = \frac{\hat{u}}{a_{ref}}, \quad v = \frac{\hat{v}}{a_{ref}}, \quad w = \frac{\hat{w}}{a_{ref}}, \quad t = \frac{a_{ref}\hat{t}}{L_{ref}},$$

$$\rho = \frac{\hat{\rho}}{\rho_{ref}}, \quad p = \frac{\hat{p}}{\left(\rho a^2\right)_{ref}},$$

$$e_t = \frac{\hat{e}_t}{a^2{}_{ref}}, \quad T = \frac{\hat{T}}{T_{ref}},$$

$$\mu = \frac{\hat{\mu}}{\mu_{ref}}, \quad \kappa = \frac{\hat{\kappa}}{\kappa_{ref}},$$

where

$$\mu_{ref} = (\rho a L)_{ref}\ , \quad \kappa_{ref} = (\mu R)_{ref}\ , \quad a^2{}_{ref} = (RT)_{ref}$$

with

$R_{ref} \equiv$ Universal Gas Constant

In the following relations, the “ ^ ” and the subscript “ref” represent dimensional quantities. Instead of inputting the Reynolds and Prandtl Numbers directly, as in the previous codes, dimensional values of pressure, temperature, velocity, density, and energy and a designation of which pair of these variables to use is read for initialization and scaling purposes. From the designated pair of thermodynamic variables, the others are then calculated and initial and reference values determined. Either Mach Number or velocity is still input and used to determine free–stream speed. For example, suppose free–stream values of pressure and temperature are designated for initialization and scaling. From this, free–stream speed of sound, total energy, velocity (from the input Mach Number), and density are calculated. The length scale is still accounted for in the grid, but instead of a so–called characteristic length, a length scale of one meter is used (i.e., the grid is in terms of meters). All of the information needed by the above relations is now available. By scaling in this manner, the non–dimensional equations take the exact same form as the the dimensional ones (i.e., Equation (2.17) with the vectors defined by Equations (2.18) through (2.21)).

Curvilinear Transformation

The Cartesian reference frame is useful for expressing the vector and tensor quantities of the governing equations in their respective component forms. However, this reference frame is not very practical for "real–world" problems. A grid that conforms to the shape of the geometries involved is much more useful and greatly simplifies boundary condition treatment. In using a body–conforming grid, the physical space of the problem may be mapped to a simpler computational space (see Thompson, et al [23]).

The three–dimensional, time–dependent, curvilinear coordinates are defined in a general form as

$$\xi = \xi(x,y,z,t), \quad \eta = \eta(x,y,z,t), \quad \zeta = \zeta(x,y,z,t), \quad \tau = t \ . \tag{2.22}$$

By use of the chain rule, the following relations give the Cartesian derivatives in terms of their curvilinear counterparts:

$$\begin{aligned}
\frac{\partial}{\partial t} &= \frac{\partial}{\partial \tau} + \xi_t \frac{\partial}{\partial \xi} + \eta_t \frac{\partial}{\partial \eta} + \zeta_t \frac{\partial}{\partial \zeta} \ , \\
\frac{\partial}{\partial x} &= \xi_x \frac{\partial}{\partial \xi} + \eta_x \frac{\partial}{\partial \eta} + \zeta_x \frac{\partial}{\partial \zeta} \ , \\
\frac{\partial}{\partial y} &= \xi_y \frac{\partial}{\partial \xi} + \eta_y \frac{\partial}{\partial \eta} + \zeta_y \frac{\partial}{\partial \zeta} \ , \\
\frac{\partial}{\partial z} &= \xi_z \frac{\partial}{\partial \xi} + \eta_z \frac{\partial}{\partial \eta} + \zeta_z \frac{\partial}{\partial \zeta} \ .
\end{aligned} \tag{2.23}$$

The partial derivative terms in Equation (2.23) are given by the expressions at the top of following page.

$$\xi_x = J^{-1}\left(y_\eta z_\zeta - z_\eta y_\zeta\right) \qquad \eta_x = J^{-1}\left(z_\xi y_\zeta - y_\xi z_\zeta\right) \qquad \zeta_x = J^{-1}\left(y_\xi z_\eta - z_\xi y_\eta\right)$$

$$\xi_y = J^{-1}\left(z_\eta x_\zeta - x_\eta z_\zeta\right) \qquad \eta_y = J^{-1}\left(x_\xi z_\zeta - z_\xi x_\zeta\right) \qquad \zeta_y = J^{-1}\left(z_\xi x_\eta - x_\xi z_\eta\right)$$

$$\xi_z = J^{-1}\left(x_\eta y_\zeta - y_\eta x_\zeta\right) \qquad \eta_z = J^{-1}\left(y_\xi x_\zeta - x_\xi y_\zeta\right) \qquad \zeta_z = J^{-1}\left(x_\xi y_\eta - y_\xi z_\eta\right)$$

and (2.24)

$$\begin{aligned} \xi_t &= -x_\tau \xi_x - y_\tau \xi_y - z_\tau \xi_z, \\ \eta_t &= -x_\tau \eta_x - y_\tau \eta_y - z_\tau \eta_z, \\ \zeta_t &= -x_\tau \zeta_x - y_\tau \zeta_y - z_\tau \zeta_z. \end{aligned}$$

The Jacobian, J, of the inverse coordinate transformation is defined mathematically as:

$$J \equiv \det\left|\frac{\partial(x,y,z)}{\partial(\xi,\eta,\zeta)}\right|,$$

or (2.25)

$$J = x_\xi\left(y_\eta z_\zeta - z_\eta y_\zeta\right) - y_\xi\left(x_\eta z_\zeta - z_\eta x_\zeta\right) + z_\xi\left(x_\eta y_\zeta - y_\eta x_\zeta\right) .$$

Physically speaking, the Jacobian represents the volume of a grid cell. The grid cell faces, or surfaces, are given by Jk_x, Jk_y, and Jk_z where k = ξ, η, or ζ. Referring to the so–called time metric terms (ξ_t, η_t, and ζ_t), these are the measures of the amount that the grid changes with respect to time. For a stationary grid, therefore, these terms are zero.

In terms of these curvilinear coordinates, the governing equations (in their vector form) become:

$$\frac{\partial Q}{\partial \tau} + \frac{\partial F}{\partial \xi} + \frac{\partial G}{\partial \eta} + \frac{\partial H}{\partial \zeta} = 0. \qquad (2.26)$$

The vector of conserved variables, Q, in Equation (2.26) is defined below; this expression is simply the product of the cell volume, J, and Equation (2.18):

$$Q = J\begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ e_t \end{bmatrix} . \tag{2.27}$$

The flux vectors, F, G, and H are defined as:

$$F = J\begin{bmatrix} \rho U \\ \rho u U + \xi_x p - T_{\xi 1} \\ \rho v U + \xi_y p - T_{\xi 2} \\ \rho w U + \xi_z p - T_{\xi 3} \\ e_t U + p\left(\xi_x u + \xi_y v + \xi_z w\right) - u T_{\xi 1} - v T_{\xi 2} - w T_{\xi 3} + Q_\xi \end{bmatrix} , \tag{2.28}$$

$$G = J\begin{bmatrix} \rho V \\ \rho u V + \eta_x p - T_{\eta 1} \\ \rho v V + \eta_y p - T_{\eta 2} \\ \rho w V + \eta_z p - T_{\eta 3} \\ e_t V + p\left(\eta_x u + \eta_y v + \eta_z w\right) - u T_{\eta 1} - v T_{\eta 2} - w T_{\eta 3} + Q_\eta \end{bmatrix} , \tag{2.29}$$

$$H = J\begin{bmatrix} \rho W \\ \rho u W + \zeta_x p - T_{\zeta 1} \\ \rho v W + \zeta_y p - T_{\zeta 2} \\ \rho w W + \zeta_z p - T_{\zeta 3} \\ e_t W + p\left(\zeta_x u + \zeta_y v + \zeta_z w\right) - u T_{\zeta 1} - v T_{\zeta 2} - w T_{\zeta 3} + Q_\zeta \end{bmatrix} . \tag{2.30}$$

Three new terms appearing in these transformed flux vectors are the contravarient velocities, U, V, and W. These velocity components are in directions normal to their respective cell surfaces (i.e., ξ, η, and ζ) and are given as:

$$\begin{aligned} U &= \xi_t + \xi_x u + \xi_y v + \xi_z w \ , \\ V &= \eta_t + \eta_x u + \eta_y v + \eta_z w \ , \\ W &= \zeta_t + \zeta_x u + \zeta_y v + \zeta_z w \ . \end{aligned} \tag{2.31}$$

The transformed viscous shear and heat flux terms are:

$$\begin{aligned} T_{k1} &= k_x\tau_{xx} + k_y\tau_{xy} + k_z\tau_{xz} \ , \\ T_{k2} &= k_x\tau_{yx} + k_y\tau_{yy} + k_z\tau_{yz} \ , \\ T_{k3} &= k_x\tau_{zx} + k_y\tau_{zy} + k_z\tau_{zz} \ , \\ Q_k &= k_x q_x + k_y q_y + k_z q_z \qquad \text{where } k = \xi, \eta, \text{ or } \zeta \ . \end{aligned} \tag{2.32}$$

The actual Cartesian shear stress and heat flux components in the curvilinear coordinate system are given on the following page.

The diffusive terms given by Equations (2.33) and (2.34) will be referred to as the “full 3–D” because they contain the fully expanded velocity and temperature derivatives, fully expanded, that is, in terms of the curvilinear coordinates. When these terms are placed in their respective places in Equation (2.32), a number of cross–derivative terms will result. This is the more correct form of the equations, but it is also more expensive. On some occasions, an approximation can be made to these diffusive terms such that the cross derivatives are neglected. This approximation is known as the “thin–layer” approximation (see Chen [22], or Cox [7] for further details) and may be viewed as similar to Prandtl’s famous boundary layer approximation. However, the problems of interest do not

$$\tau_{xx} = \frac{2}{3}\mu\left[2\left(\xi_x\frac{\partial u}{\partial \xi} + \eta_x\frac{\partial u}{\partial \eta} + \zeta_x\frac{\partial u}{\partial \zeta}\right) - \left(\xi_y\frac{\partial v}{\partial \xi} + \eta_y\frac{\partial v}{\partial \eta} + \zeta_y\frac{\partial v}{\partial \zeta}\right)\right]$$

$$- \frac{2}{3}\mu\left(\xi_z\frac{\partial w}{\partial \xi} + \eta_z\frac{\partial w}{\partial \eta} + \zeta_z\frac{\partial w}{\partial \zeta}\right) ,$$

$$\tau_{yy} = \frac{2}{3}\mu\left[2\left(\xi_y\frac{\partial v}{\partial \xi} + \eta_y\frac{\partial v}{\partial \eta} + \zeta_y\frac{\partial v}{\partial \zeta}\right) - \left(\xi_x\frac{\partial u}{\partial \xi} + \eta_x\frac{\partial u}{\partial \eta} + \zeta_x\frac{\partial u}{\partial \zeta}\right)\right]$$

$$- \frac{2}{3}\mu\left(\xi_z\frac{\partial w}{\partial \xi} + \eta_z\frac{\partial w}{\partial \eta} + \zeta_z\frac{\partial w}{\partial \zeta}\right) ,$$

$$\tau_{zz} = \frac{2}{3}\mu\left[2\left(\xi_z\frac{\partial w}{\partial \xi} + \eta_z\frac{\partial w}{\partial \eta} + \zeta_z\frac{\partial w}{\partial \zeta}\right) - \left(\xi_x\frac{\partial u}{\partial \xi} + \eta_x\frac{\partial u}{\partial \eta} + \zeta_x\frac{\partial u}{\partial \zeta}\right)\right]$$

$$- \frac{2}{3}\mu\left(\xi_y\frac{\partial v}{\partial \xi} + \eta_y\frac{\partial v}{\partial \eta} + \zeta_y\frac{\partial v}{\partial \zeta}\right) ,$$

$$\tau_{xy} = \mu\left[\left(\xi_y\frac{\partial u}{\partial \xi} + \eta_y\frac{\partial u}{\partial \eta} + \zeta_y\frac{\partial u}{\partial \zeta}\right) + \left(\xi_x\frac{\partial v}{\partial \xi} + \eta_x\frac{\partial v}{\partial \eta} + \zeta_x\frac{\partial v}{\partial \zeta}\right)\right] ,$$

$$\tau_{xz} = \mu\left[\left(\xi_z\frac{\partial u}{\partial \xi} + \eta_z\frac{\partial u}{\partial \eta} + \zeta_z\frac{\partial u}{\partial \zeta}\right) + \left(\xi_x\frac{\partial w}{\partial \xi} + \eta_x\frac{\partial w}{\partial \eta} + \zeta_x\frac{\partial w}{\partial \zeta}\right)\right] ,$$

$$\tau_{yz} = \mu\left[\left(\xi_z\frac{\partial v}{\partial \xi} + \eta_z\frac{\partial v}{\partial \eta} + \zeta_z\frac{\partial v}{\partial \zeta}\right) + \left(\xi_y\frac{\partial w}{\partial \xi} + \eta_y\frac{\partial w}{\partial \eta} + \zeta_y\frac{\partial w}{\partial \zeta}\right)\right] . \tag{2.33}$$

$$q_x = -\kappa\left(\xi_x\frac{\partial T}{\partial \xi} + \eta_x\frac{\partial T}{\partial \eta} + \zeta_x\frac{\partial T}{\partial \zeta}\right) ,$$

$$q_y = -\kappa\left(\xi_y\frac{\partial T}{\partial \xi} + \eta_y\frac{\partial T}{\partial \eta} + \zeta_y\frac{\partial T}{\partial \zeta}\right) ,$$

$$q_z = -\kappa\left(\xi_z\frac{\partial T}{\partial \xi} + \eta_z\frac{\partial T}{\partial \eta} + \zeta_z\frac{\partial T}{\partial \zeta}\right) . \tag{2.34}$$

always lend themselves to this approximation, and, thus, the full 3–D form of the equations is necessary. The present code has been modified so that an input parameter (logical) is designated "true" to use the full 3–D version, or "false" to use the thin–layer version.

Finally, the coefficients of viscosity and thermal conductivity are given as the sum of their laminar and turbulent components; i.e.,

$$\mu = \mu_l + \mu_t \qquad \text{and} \qquad \kappa = \kappa_l + \kappa_t \ .$$

The laminar components are primarily functions of temperature and are determined through the use of Sutherland's Law (see White [24] or Warsi [25] for descriptions and for the relevant coefficients for some gases). The turbulent eddy viscosity, μ_t, and eddy conductivity, κ_t, are determined by use of the algebraic turbulence model developed by Baldwin and Lomax [26], which is based on Prandtl's mixing–length theory. Though limited in its applicability, as most turbulence models are, the Baldwin–Lomax model has proved to be reliable for many wall–bounded turbulent boundary layer flows.

Heat Conduction Equation

For heat conduction in a solid medium, the governing equation may be obtained by simply setting $\vec{V} = 0$ in Equation (2.3). This may be done due to the fact that the energy transfer is purely diffusive in nature. The resulting conservation of energy equation is

$$\frac{\partial e_i}{\partial t} + \vec{\nabla} \cdot \vec{q} = 0 \tag{2.35}$$

In this case, the energy variable is basically the same as the internal energy component from the flow problem. This energy term may be expressed (per volume) in terms of the specific heat of the solid and the temperature as

$$e_i = \rho c T \ .$$

If the Cartesian coordinate system is chosen, then the coordinate–invariant form of Equation (2.35) becomes

$$\frac{\partial e_i}{\partial t} + \frac{\partial q_x}{\partial x} + \frac{\partial q_y}{\partial y} + \frac{\partial q_z}{\partial z} = 0 \ . \tag{2.36}$$

In terms of conductivity, specific heat, and temperature, Equation (2.35) is

$$\frac{\partial}{\partial t}(\rho c T) - \frac{\partial}{\partial x}\left(\kappa \frac{\partial T}{\partial x}\right) - \frac{\partial}{\partial y}\left(\kappa \frac{\partial T}{\partial y}\right) - \frac{\partial}{\partial z}\left(\kappa \frac{\partial T}{\partial z}\right) = 0 \ , \tag{2.37}$$

which in the case of constant properties, ρ, c, and κ is the familiar heat equation

$$\frac{\partial T}{\partial t} = \alpha\left[\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2}\right] , \tag{2.38}$$

where $\alpha \equiv$ thermal diffusivity = $\kappa/\rho c$.

Scaling and Curvilinear Transformation of the Heat Equation

As already discussed in the case of the flow solver, the heat conduction equation may require scaling to a non–dimensional form. The scaling relations from before can be used unchanged for ρ, T, x, y, and z. Reference values for conductivity, κ, and specific heat, c, are defined in the input file. The non–dimensional time is slightly different from before and is given as

$$t = \frac{\alpha_{ref}\ \hat{t}}{L^2{}_{ref}} \ ,$$

or (2.39)

$$t = \frac{\kappa_{ref}\ \hat{t}}{(\rho c L^2)_{ref}} \ .$$

The transformation to the curvilinear coordinate system is similar to what was done for the flow solver. When transformed to the curvilinear system, Equation (2.36) becomes

$$\frac{\partial}{\partial \tau}(J\rho c T) + \frac{\partial Q_\xi}{\partial \xi} + \frac{\partial Q_\eta}{\partial \eta} + \frac{\partial Q_\zeta}{\partial \zeta} = 0 \ . \tag{2.40}$$

where

$$\begin{aligned}
Q_\xi &= -J\kappa\left[\xi_x\left(\xi_x\frac{\partial T}{\partial \xi} + \eta_x\frac{\partial T}{\partial \eta} + \zeta_x\frac{\partial T}{\partial \zeta}\right) + \xi_y\left(\xi_y\frac{\partial T}{\partial \xi} + \eta_y\frac{\partial T}{\partial \eta} + \zeta_y\frac{\partial T}{\partial \zeta}\right)\right] \\
&\qquad - J\kappa\xi_z\left(\xi_z\frac{\partial T}{\partial \xi} + \eta_z\frac{\partial T}{\partial \eta} + \zeta_z\frac{\partial T}{\partial \zeta}\right) \\
Q_\eta &= -J\kappa\left[\eta_x\left(\xi_x\frac{\partial T}{\partial \xi} + \eta_x\frac{\partial T}{\partial \eta} + \zeta_x\frac{\partial T}{\partial \zeta}\right) + \eta_y\left(\xi_y\frac{\partial T}{\partial \xi} + \eta_y\frac{\partial T}{\partial \eta} + \zeta_y\frac{\partial T}{\partial \zeta}\right)\right] \\
&\qquad - J\kappa\eta_z\left(\xi_z\frac{\partial T}{\partial \xi} + \eta_z\frac{\partial T}{\partial \eta} + \zeta_z\frac{\partial T}{\partial \zeta}\right) \\
Q_\zeta &= -J\kappa\left[\zeta_x\left(\xi_x\frac{\partial T}{\partial \xi} + \eta_x\frac{\partial T}{\partial \eta} + \zeta_x\frac{\partial T}{\partial \zeta}\right) + \zeta_y\left(\xi_y\frac{\partial T}{\partial \xi} + \eta_y\frac{\partial T}{\partial \eta} + \zeta_y\frac{\partial T}{\partial \zeta}\right)\right] \\
&\qquad - J\kappa\zeta_z\left(\xi_z\frac{\partial T}{\partial \xi} + \eta_z\frac{\partial T}{\partial \eta} + \zeta_z\frac{\partial T}{\partial \zeta}\right)
\end{aligned} \tag{2.41}$$

All derivative terms are retained for heat conduction within a solid (i.e., no thin–layer approximation is made).

Heat conduction requires only the determination of temperature, and so this type of problem is smaller, with respect to computational demands, relative to that of fluid flow. Since only temperature is required to define the heat conduction field, a scalar variable rather than a five–variable vector is all that is solved for at any given grid cell location. Significantly less memory requirements and computational operations are a result.

CHAPTER III

NUMERICAL FORMULATION

The governing equations of the previous chapter are the mathematical expressions of the physics involved in the problem of interest. Solutions of these equations can be viewed as exact solutions to the problem, keeping in mind the assumptions made in the formulation of the equations. However, actual exact solutions to these equations are limited to problems in which further simplifying assumptions must be made. So, for most practical problems, it is an impossibility to solve these equations in an "exact" manner. Therefore, numerical approximations are the means by which solutions can be obtained for practical, real–world problems.

The finite–volume approach is used in this study as the basic numerical approach. This methodology matches well with the control–volume approach of obtaining the governing equations. The domain of interest is discretized into grid cells; then for each grid cell the value of the "Q–vector" is desired. The cell center is assumed to be the point at which the Q–vector is calculated, though the value of Q is assumed to be constant over the whole cell. The fluxes F, G, and H are assumed to be acting on the respective cell surfaces, or faces, and are assumed constant over the given face. The governing vector equation, when spatially discretized, becomes

$$\frac{\partial Q}{\partial \tau}\Delta\xi\Delta\eta\Delta\zeta + \delta_i(F)\Delta\eta\Delta\zeta + \delta_j(G)\Delta\xi\Delta\zeta + \delta_k(H)\Delta\xi\Delta\eta = 0 \ . \qquad (3.1)$$

The terms $\Delta\xi$, $\Delta\eta$, and $\Delta\zeta$ may be taken to be unit values with an appropriate choice of curvilinear coordinates; Q, F, G, and H are those vectors defined in the previous chapter. Also the operator $\delta_l = (*)_{l+1/2} - (*)_{l-1/2}$. Since the Q–vector is assumed to be at the cell center, the grid index is assumed to be located there also; therefore, the index of l ± 1/2 is associated with cell faces.

Implicit Algorithm and Numerical Flux

Numerical algorithms using the time–marching technique may be employed for both steady ($\partial/\partial t = 0$) or unsteady ($\partial/\partial t \neq 0$) flow problems. In the former case, the time variable serves simply as an iteration parameter; in the latter case, it represents the actual physical time (or time–step in the discretized sense). Such algorithms fall typically into one of two categories: explicit or implicit. By choosing the implicit approach, an algorithm with a higher stability bound is generally achieved, though at a higher computational price per time step iteration.

Using a 1st–order, backward difference to discretize the time derivative and recognizing the values of $\Delta\xi$, $\Delta\eta$, and $\Delta\zeta$ as defined above, Equation (3.1) becomes (in an implicit sense)

$$\frac{\Delta Q^n}{\Delta \tau} + \delta_i F^{n+1} + \delta_j G^{n+1} + \delta_k H^{n+1} = 0 \tag{3.2}$$

where $\Delta Q^n = Q^{n+1} - Q^n$, superscript n => time step value. The flux terms are, in general, nonlinear in nature and, thus, require a linear approximation to obtain a solution. The common linearization with respect to time is (using F as an example)

$$F^{n+1} = F^n + \left(\frac{\partial F}{\partial Q}\right)^n \Delta Q^n + O(\Delta\tau)^2 \tag{3.3}$$

Assuming for reasons of simplicity that the problem is one–dimensional in space, Equation (3.2) is re–written as

$$\frac{\Delta Q^n}{\Delta\tau} + \left(F^{n+1}\right)_{i+1/2} - \left(F^{n+1}\right)_{i-1/2} = 0 \tag{3.4}$$

In general, the numerical flux acting on a cell face is a function of the Q–vector in the neighboring cells (e.g., $F_{i+1/2} = F(Q_i,Q_{i+1})$ and $F_{i-1/2} = F(Q_i,Q_{i-1})$). Taking this fact into account and applying the linearization technique described in Equation (3.3) to Equation (3.4), the basic form of the implicit numerical algorithm is given as:

$$\left[\frac{I}{\Delta\tau} + \frac{\partial F^n_{i+1/2}}{\partial Q^n_i} - \frac{\partial F^n_{i-1/2}}{\partial Q^n_i}\right]\Delta Q^n_i + \frac{\partial F^n_{i+1/2}}{\partial Q^n_{i+1}}\Delta Q^n_{i+1} - \frac{\partial F^n_{i-1/2}}{\partial Q^n_{i-1}}\Delta Q^n_{i-1} = F^n_{i-1/2} - F^n_{i+1/2} = -\,\delta_i F^n \tag{3.5}$$

For problems in two or three spatial dimensions, the above equation is simply expanded using the same technique (a summary of this may be found in Vanden and Whitfield [28]).

The convective part of the flux vector is based on the classic wave equation of physics. Wave propagation has a natural directional dependence associated with it. Therefore, since the convective fluxes are modeled as wave propagation, these, too, can be seen as having directional dependence. If one divides, or splits, the convective flux vector in accordance with its direction of wave propagation, a physically accurate representation of the flux at a cell face is determined; this general technique is commonly referred to as "upwinding". The splitting of the flux vector, F, is written as

$$F = F^+ + F^-$$

The same notation is used to represent the splitting of the G and H vectors. The F^+ sub–vector consists of flux terms with a wave propagation in the positive coordinate direction, and the F^- sub–vector has terms with negative directional propagation.

Referring to Equation (3.5), the terms of "∂F/∂Q" are *n* x *n* matrices (n = 3,4, or 5 for 1–, 2–, or 3–D flow problems) and are known as flux Jacobians. A common naming convention for these terms is $A_{i+1/2} \equiv (\partial F/\partial Q)_{i+1/2}$ and $A_{i-1/2} \equiv (\partial F/\partial Q)_{i-1/2}$. The wave speeds and directions mentioned in the previous paragraph are determined from the eigenvalues of the flux Jacobian, A. The "split" flux Jacobians are then defined as $A^+ \equiv \partial F^+/\partial Q$ and $A^- \equiv \partial F^-/\partial Q$. Further details of the actual splitting of the flux vectors and the split flux Jacobians have been presented a number of times (see Arabshahi [1], Janus [21], Chen [22] for the ideal–gas flows and Cox [7] and Cinnella [29] for chemically reacting flows) and will not be repeated here. A detailed derivation of the so–called analytical flux Jacobians may be found in Appendix B of Belk [4].

For the present code, Cox made use of a splitting of the diffusive components of the flux vectors, based on the work of Cinnella [29], in order to form viscous flux Jacobians. This particular technique allows for the viscous Jacobians to be combined with the inviscid (convective) Jacobians such that the coefficient matrix (i.e., the left–hand side) maintains the so–called "LU" structure. Details of these discretized viscous Jacobians can be found in Appendix E of Cox [7]. In the past, the viscous fluxes were treated explicitly. These viscous Jacobians allow for the implicit treatment of the viscous fluxes, if so desired.

Returning to Equation (3.5) and re–writing it in terms of A^+ and A^-,

$$\left(\frac{I}{\Delta\tau} + A^{+}_{i+1/2} - A^{-}_{i-1/2}\right)^{n} \Delta Q^{n}_{i} + A^{-n}_{i+1/2}\Delta Q^{n}_{i+1} - A^{+n}_{i-1/2}\Delta Q^{n}_{i-1} = -R^{n} \quad (3.6)$$

where $R^n = \delta_i F^n$. For the purpose of brevity, this may be re–written as

$$\left(\frac{I}{\Delta\tau} + \delta_i A^{+} \cdot + \delta_i A^{-} \cdot\right)^{n} \Delta Q^{n}_{i} = -R^{n} \quad (3.7)$$

where " . " implies the matrix–vector multiplication of the appropriate terms. Again, Equations (3.5) through (3.7) have assumed a problem in just one spatial dimension. As mentioned earlier, to expand the implicit algorithm to two– or three–dimensional problems, the same concepts and notation are used as that for Equations (3.5) through (3.7). The split flux sub–vectors are G^+ , G^- and H^+ , H^- for the j and k grid directions, respectively. The accompanying flux Jacobians are $B^+ \equiv \partial G^+/\partial Q$ and $B^- \equiv \partial G^-/\partial Q$, $C^+ \equiv \partial H^+/\partial Q$ and $C^- \equiv \partial H^-/\partial Q$. In the compact form of Equation (3.7), a three–dimensional problem may be written as

$$\left(\frac{I}{\Delta\tau} + \delta_i A^+ \cdot + \delta_i A^- \cdot + \delta_j B^+ \cdot + \delta_j B^- \cdot + \delta_k C^+ \cdot + \delta_k C^- \cdot \right)^n \Delta Q^n_{ijk} = -R^n$$

$$\text{with} \quad R^n = \delta_i F^n + \delta_j G^n + \delta_k H^n \tag{3.8}$$

The right–hand side of Equation (3.8), often referred to as the “steady–state residual”, represents the difference in flux between the respective faces of a given grid cell. Therefore, the flux must be constructed at each face in order to calculate this difference. The methodology for constructing the numerical flux at the cell faces will be briefly discussed below. Details have been covered numerous times, so only a summary will be given. The diffusive fluxes are comprised of the product of a diffusive coefficient (viscosity or thermal conductivity, for example) and the derivative of velocity or temperature at the cell face. A second–order, central difference approximation to this derivative may be constructed by simply taking the difference of the cell–centered values of velocity or temperature in the cells adjacent to the cell face. This is a relatively easy task; the cell–centered values are known and a simple subtraction is all that is required.

Constructing the convective flux at the cell face is not quite as simple, due to possible discontinuties. Again, the value is known at the cell center, so the challenge comes in calculating a reasonably accurate value at the cell face. And in keeping with the upwind philosophy of spatial differencing, the direction of wave propagation should be accounted for in the flux calculation. A method originated by Godunov in 1959 [30] was a concept in which a discontinuity is assumed to exist at each cell face. This initial discontinuity in dependent variables gives rise to a 1–D Riemann (shock tube) problem; the basic structure of such a problem consists of an expansion fan, a shock wave, and a contact discontinuity. The propagation of these wave structures, in addition to the known cell–centered values, is then used to obtain a flux at the cell face. This method can give exact solutions to the problem in question. However, when used for nonlinear problems (as in the present case), the method is quite expensive. A good basic description of the principles of this method may be found in Hirsch [31].

A number of methods based on the basic concept proposed by Godunov have since been developed; unlike Godunov's scheme, some of these newer methods do not attempt to solve the exact problem, and are referred to as approximate Riemann solvers. Perhaps the most popular is the method developed by Roe [32]. Often called Roe's flux differencing scheme, it approximates the exact problem by means of a special linearization. Based on this linearization technique, average values for the dependent variables may be calculated in terms of the cell–centered values on either side of the cell face. These Roe–averaged variables may then be used to construct the flux at the given cell face. This method works very well for many cases, but it is not perfect. It has only a first–order spatial accuracy, and it may allow the calculation of non–physical solutions (also known as expansion shocks). Since the origin of the Roe scheme, a number of methods have been developed to improve on its faults, especially the one of non–physical solutions. The methods of Quirk [33] and Harten, Lax, and van Leer (HLLE) [34] are two examples. Hirsch [31] again gives a good basic explanation of Roe's scheme, and the essential details may also be found in Simpson [5] or Arabshahi [1] for ideal gas flows and Cox [7] for equilibrium reacting flows.

As already mentioned, Roe's method gives numerical flux values of first–order spatial accuracy. For higher–order approximations, corrections are added to this first–order flux. The total–variation–diminishing scheme of Osher and Chakravarthy [35] was incorporated into the present code by Arabshahi for higher spatial accuracy. Also, the use of so–called flux limiters is needed for the higher–order scheme near shocks. The flux limiter is required to act against an over– or under shoot of the higher–order solution at the shock. The limiter basically reduces the solution to first order at the shock by “chopping” off an over– or under shoot. By doing this, non–physical oscillations are prevented from forming in the region near the shock and then corrupting the solution. A brief description of the higher–order method and limiters may be found in [1].

Modified Two–Pass Scheme and Symmetric Gauss–Seidel Iterations

In order to solve the linearized system of equations represented by Equation (3.8), the modified two–pass scheme of Whitfield [36] is employed and will be briefly discussed in the following. First, define three matrix operators:

$$D^n_{i,j,k} = \frac{I}{\Delta\tau} + \left(A^+_{i+1/2,j,k} - A^-_{i-1/2,j,k}\right)^n + \left(B^+_{i,j+1/2,k} - B^-_{i,j-1/2,k}\right)^n + \left(C^+_{i,j,k+1/2} - C^-_{i,j,k-1/2}\right)^n \ , \tag{3.9}$$

$$L^n_{i,j,k} \cdot \Delta Q^n_{i,j,k} = \left(A^+_{i-1/2,j,k} \Delta Q_{i-1,j,k} + B^+_{i,j-1/2,k} \Delta Q_{i,j-1,k} + C^+_{i,j,k-1/2} \Delta Q_{i,j,k-1}\right)^n \ , \tag{3.10}$$

$$U^n_{i,j,k} \cdot \Delta Q^n_{i,j,k} = \left(A^-_{i+1/2,j,k} \Delta Q_{i+1,j,k} + B^-_{i,j+1/2,k} \Delta Q_{i,j+1,k} + C^-_{i,j,k+1/2} \Delta Q_{i,j,k+1}\right)^n \ . \tag{3.11}$$

Equation (3.8) may now be written as

$$\left(D_{i,j,k} \cdot - L_{i,j,k} \cdot + U_{i,j,k} \cdot\right)^n \Delta Q^n_{i,j,k} = - R^n_{i,j,k} \ . \tag{3.12}$$

One may then perform an approximate factorization on the above equation similar to that of Briley and McDonald [27] or Beam and Warming [37]:

$$\left[D^n_{i,j,k} - L^n_{i,j,k}\cdot\right] D^{-1}\left[D^n_{i,j,k} + U^n_{i,j,k}\cdot\right]\Delta Q^n_{i,j,k} = -R^n_{i,j,k}\ . \qquad (3.13)$$

Equation (3.13) may be viewed as the same form of the LU factorization from basic linear algebra. That is to say

$$[LU]x = b\ . \qquad (3.14)$$

This system of equations is solved in two steps (passes):

$$\begin{aligned} Ly &= b && \text{forward substitution,} \\ Ux &= y && \text{back substitution.} \end{aligned}$$

In the same sort of way, Equation (3.13) is solved in two passes:

$$\begin{aligned} \left[D_{i,j,k} - L_{i,j,k}\cdot\right]^n\left[Q^*_{i,j,k}\right] &= -R^n_{i,j,k} && \text{forward pass} \\ \left[D_{i,j,k} + U_{i,j,k}\cdot\right]^n\left[\Delta Q^n_{i,j,k}\right] &= D^n_{i,j,k}Q^*_{i,j,k} && \text{backward pass} \end{aligned} \qquad (3.15)$$

where $$D^n_{i,j,k}Q^*_{i,j,k} = -R^n_{i,j,k} + \left[L^n_{i,j,k}\cdot\right]\left[Q^*_{i,j,k}\right]\ .$$

Equation (3.15) is the modified two–pass algorithm. This two–pass sequence of operations will give an approximate solution to the system of equations; the solution is approximate because, again, Equation (3.13) is an approximate factorization of the original system of equations. However, an iteration process that may be viewed as a symmetric Gauss–Seidel sequence can easily be made from Equation (3.15). The iterative sequence if advanced to convergence (i.e., ΔQ = 0) would remove the factorization error mentioned above.

A single iteration step of the symmetric Gauss–Seidel (SGS) method for solving systems of linear equations is practically the same as that given by Equation (3.15). A forward pass with a block lower–triangular coefficient matrix is followed by a backward pass with a block upper–triangular coefficient matrix. This process is repeated for either some specified number of iterations or until some convergence criterion is met. Therefore, Equation (3.15) is converted to a SGS sequence by placing the modified two–pass scheme within a loop and executing this loop for a certain number of times. Equation (3.15) as a SGS iterative sequence is then written as:

$$\left[D_{i,j,k} - L_{i,j,k}\right]^{n}\left[\mathit{\Delta} Q^{n}_{i,j,k}\right]^{2p+1} = -R^{n}_{i,j,k} - \left[U^{n}_{i,j,k}\cdot\right]\left[\mathit{\Delta} Q^{n}_{i,j,k}\right]^{2p} ,$$

$$\tag{3.16}$$

$$\left[D_{i,j,k} + U_{i,j,k}\right]^{n}\left[\mathit{\Delta} Q^{n}_{i,j,k}\right]^{2p+2} = -R^{n}_{i,j,k} + \left[L^{n}_{i,j,k}\cdot\right]\left[\mathit{\Delta} Q^{n}_{i,j,k}\right]^{2p+1} ,$$

where $\left[\mathit{\Delta} Q^{n}_{i,j,k}\right]^{0} = [0]$

and $p = 0, 1, 2, \ldots, SGS\ \ LIMIT$.

If the SGS limit is set to zero, then Equation (3.16) is exactly the same as the modified two–pass of Equation (3.15).

Newton's Method for Nonlinear Equations

Newton's method is an iterative technique for solving nonlinear equations. An important advantage of this method is its rapid convergence property. Assuming that an initial value of the iterative sequence is sufficiently close to the solution, the sequence will rapidly meet a given convergence criterion for many problems. A good theoretical discussion of Newton's method (for systems of equations) may be found in chapter five of Dennis and Schnabel [38] (chapter two covers this for a single variable).

For the case of finding the root of a scalar function of a single variable (f(x) = 0), Newton's method is simply

$$\Delta x^k = -\left[\frac{f(x^k)}{f'(x^k)}\right] , \tag{3.17}$$

where $\Delta x^k = x^{k+1} - x^k$; k ≡ Newton iteration parameter.

When desiring the solution or root of a vector function of a vector variable, $[\overline{F}(\overline{x})] = [\overline{0}]$, Newton's method results in an expression similar to that of Equation (3.17):

$$\Delta\overline{x}^k = -\left[\overline{F}'\left(\overline{x}^k\right)\right]^{-1}\left[\overline{F}\left(\overline{x}^k\right)\right] . \tag{3.18}$$

where $\left[\overline{F}'(\overline{x})\right]$ is the Jacobian matrix, $[\partial\overline{F}/\partial\overline{x}]$. Calculating the actual inverse of the Jacobian could be quite difficult, so a re–arranging of Equation (3.18) is needed to avoid this difficulty. Therefore, Equation (3.18) is re–written as

$$\overline{F}'\left(\overline{x}^k\right)\Delta\overline{x}^k = -\overline{F}\left(\overline{x}^k\right) . \tag{3.19}$$

which has a form similar to that of the implicit algorithm described above. Comparing Equation (3.19) to the one–dimensional example of Equation (3.7):

$$\Delta\overline{x}^k \leftrightarrow \Delta Q_i^{n+1,k} = Q_i^{n+1,k+1} - Q_i^{n+1,k} ,$$

$$\overline{F}'\left(\overline{x}^k\right) \leftrightarrow \left[\frac{I}{\Delta\tau} + \delta_i A^+ \cdot + \delta_i A^- \cdot\right]^{n+1,k} , \tag{3.20}$$

$$-\overline{F}\left(\overline{x}^k\right) \leftrightarrow -R_i^{n+1,k} = -\left[\frac{Q_i^{n+1,k} - Q_i^n}{\Delta\tau} + \delta_i F^{n+1,k}\right] ,$$

where $\left[Q_i\right]^{n+1,1} = \left[Q_i\right]^n$

and $k = 1, 2, \ldots, NEWTON\ \ LIMIT$.

There is an important difference between Equation (3.20) and Equation (3.7); the discretized time derivative is now included on the right–hand side. The general time–dependent equation, Equation (3.2), is a nonlinear vector equation equal to the zero vector, just as $[\bar{F}(\bar{x})] = [\bar{0}]$. Therefore, the time–derivative approximation must be included in addition to the flux differences, $\delta_i F$. For two– or three–dimensional problems, one can simply include the additional terms as shown in Equation (3.8).

For any given time step, a Newton sub–iteration procedure is performed to drive the term $\Delta Q^{n+1,k}$ toward zero. As $\Delta Q^{n+1,k} \rightarrow 0$, $R^{n+1,k} \rightarrow \quad 0$ which, as shown by Equation (3.20), means that the discretized governing equation is satisfied. The beauty of this entire formulation, including also the embedded symmetric Gauss–Seidel procedure, is that a couple of input parameters can control whether a steady or unsteady problem is to be solved, or whether the modified two–pass or symmetric Gauss–Seidel procedure is used to solve the actual linear system. For example, suppose a solution to a steady–state problem is desired. One may specify the "NEWTON LIMIT" to be 1, and Equation (3.20) then reverts back to Equation (3.7). In that case, the time step serves essentially the same purpose as the iteration parameter, k, from above. For time–dependent problems, simply set NEWTON LIMIT > 1, so that sub–iterations may be performed for each time–step iteration. The actual value required for the Newton sub–iteration limit is, in general, dependent on the actual problem itself. However, often times a value of 3 to 5 will suffice. Details of this overall formulation may be found in Whitfield [36] and a good, brief presentation in Vanden and Whitfield [28].

Heat Conduction Algorithm

An implicit algorithm for the heat–conduction solver may be constructed in practically the same manner as that for the flow solver. Again, a control–volume (finite–volume) approach is used, such that the time rate of change of the quantity of interest (thermal energy, in this case) is balanced by the flux of thermal energy across the surface boundaries of the volume. The basic governing equation, Equation (2.40) , is repeated here for convenience:

$$\frac{\partial}{\partial\tau}(J\rho cT) + \frac{\partial Q_\xi}{\partial\xi} + \frac{\partial Q_\eta}{\partial\eta} + \frac{\partial Q_\zeta}{\partial\zeta} = 0 \ , \qquad (3.21)$$

or in discretized form:

$$\frac{\Delta(J\rho cT)}{\Delta\tau} + \delta_i Q_\xi + \delta_j Q_\eta + \delta_k Q_\zeta = 0 \ . \qquad (3.22)$$

The effect of thermal expansion is not being taken into account for this work, so the grid cells and their Jacobians (cell volume, J) will be constant. Moreover, directional dependence of thermal conductivity is not considered. Also, the density and thermal properties will be assumed to be constant for all of the problems considered here. The governing heat equation will then be linear with respect to the dependent variable, temperature. Since the equation is linear, the components of the coefficient matrix are comprised entirely of grid metric terms (refer again to Equation (2.41)), and once calculated, may be saved for the remainder of the time. For the case of a nonlinear problem, Newton’s method may be used in essentially the same manner as that for the flow solver, and the coefficient matrix will contain the temperature–dependent terms of specific heat, c, and conductivity, κ. The coefficient matrix will then need to be calculated each time step and, for that matter, each sub–iteration.

As an example, the one–dimensional, transient heat equation, with the assumptions mentioned above, is written as

$$\rho c\left(\frac{\Delta T_i^n}{\Delta\bar{\tau}}\right) - \kappa\left\{\left[J\left(\xi_x^2 + \xi_y^2 + \xi_z^2\right)\left(\frac{\partial T}{\partial\xi}\right)\right]_{i+1/2} - \left[J\left(\xi_x^2 + \xi_y^2 + \xi_z^2\right)\left(\frac{\partial T}{\partial\xi}\right)\right]_{i-1/2}\right\}^{n+1}$$

$$= 0 \ . \qquad (3.23)$$

The discretized space and temporal temperature derivatives in Equation (3.23), as well as the time step, are given as:

$$\left(\frac{\partial T}{\partial \xi}\right)_{i+1/2} = T_{i+1} - T_i \; ; \qquad \left(\frac{\partial T}{\partial \xi}\right)_{i-1/2} = T_i - T_{i-1}$$

$$\text{and} \qquad \Delta T_i^n = T_i^{n+1} - T_i^n \; ; \; \Delta\bar{\tau} = \frac{\Delta\tau}{J} \; . \qquad (3.24)$$

To make Equation (3.23) easier to follow, define the following coefficient matrix components:

$$A_i^+ \equiv J\left[\left(\xi_x^2 + \xi_y^2 + \xi_z^2\right)\right]_{i+1/2} \quad \text{and} \quad A_i^- \equiv J\left[\left(\xi_x^2 + \xi_y^2 + \xi_z^2\right)\right]_{i-1/2} \cdot \qquad (3.25)$$

In this case, the " + " and " – " superscripts are used simply to help identify which face, is being referred to relative to the cell center; no wave direction is implied here. Another thing to keep in mind is the fact that, this being a one–dimensional problem, no cross–derivative terms exist. In general, two– and three–dimensional problems will have grid metric terms including these cross derivatives. These additional terms would be defined in a manner similar to the A$^+$ and A$^-$. Using the above result, Equation (3.23) is expanded and re–written as

$$\rho c\left(\frac{\Delta T_i^{\,n}}{\Delta\bar{\tau}}\right) + \kappa\left(A_i^+ + A_i^-\right)T_i^{n+1} - \kappa A_i^+ T_{i+1}^{n+1} - \kappa A_i^- T_{i-1}^{n+1} = 0 \; . \qquad (3.26)$$

Converting Equation (3.26) to the so–called "residual" form (similar to that of the flow solver):

$$\left[\frac{\rho c}{\Delta\bar{\tau}} + \kappa\left(A_i^+ + A_i^-\right)\right]\Delta T_i^n - \kappa A_i^+ \Delta T_{i+1}^n - \kappa A_i^- \Delta T_{i-1}^n = -R_i^n \qquad (3.27)$$

$$\text{where} \qquad R_i^n = \kappa A_i^+\left(T_i - T_{i+1}\right)^n + \kappa A_i^-\left(T_i - T_{i-1}\right)^n \; .$$

A symmetric Gauss–Seidel iteration is used here again for solving the linear system of Equation (3.27). As with the flow solver, this procedure is comprised of multiple sets of forward and backward passes through the system

$$\left[D_i + L_i \cdot\right]^n \left[\Delta T_i^n\right]^{2p+1} = -R_i^n - \left[U_i^n \cdot\right]\left[\Delta T_i^n\right]^{2p} ,$$

$$\left[D_i + U_i \cdot\right]^n \left[\Delta T_i^n\right]^{2p+2} = -R_i^n + \left[L_i^n \cdot\right]\left[\Delta T_i^n\right]^{2p+1} , \qquad (3.28)$$

$$p = 0, 1, 2, \ldots, SGS\ LIMIT ,$$

with $$D_i \equiv \frac{\rho c}{\Delta \bar{\tau}} + \kappa\left(A_i^+ + A_i^-\right) \ ; \ L \equiv -\kappa A_i^- \ ; \ U \equiv -\kappa A_i^+ ,$$

and $$\left[\Delta T_i^n\right]^0 = [0] \ .$$

The implicit formulation of the linear heat equation is very stable, and good results have been achieved. Some of these results will be shown later in verification test cases.

Boundary Conditions

The boundary conditions for the flow solver are of the characteristic–variable type (CVBC). The principle of these conditions is to determine the values of the phantom–cell variables by way of the direction of wave propagation (i.e., into or out of the boundary). The possibilities of subsonic inflow or outflow and supersonic inflow or outflow exist for “permeable” boundaries. For solid wall, there is an impermeable CVBC or zero–pressure gradient (ZPG) condition assuming inviscid flow (i.e., non–zero tangential velocity at a solid boundary). For viscous flow, the no–slip condition is imposed (i.e., both normal and tangential velocity components are zero). In the original code developed by Arabshahi [1] and the more recent one by Cox [7], the thermal boundary conditions were limited to either the adiabatic–wall condition or the constant–temperature–wall condition. These have been modified to include specifications of spatially varying temperature or heat flux or the condition of thermal coupling (i.e., conjugate heat transfer) with the solid wall. A good description of the implementing if CVBC's may be found in Whitfield and Janus [39].

The means for specification of the boundary conditions for arbitrary geometries was developed in [1]. This method allows for a flexible construction of a multi–block grid; that is to say there is, in general, no restriction on how the grid is constructed (there is the one restriction of point–to–point continuity across block–to–block boundaries). As long as the boundary conditions are correctly specified in the input file, the code will apply the appropriate conditions (CVBC, solid wall with slip, no–slip solid wall, or block–to–block) to the corresponding boundary surface. This same methodology is used for the heat conduction solver and the coupling, if desired, of the flow solver and conduction solver.

The boundary conditions for the conduction solver consist of constant or variable temperature or heat flux and thermal coupling, either with other solid grid blocks or with fluid grid blocks. This thermal coupling will be discussed further in the next chapter.

CHAPTER IV

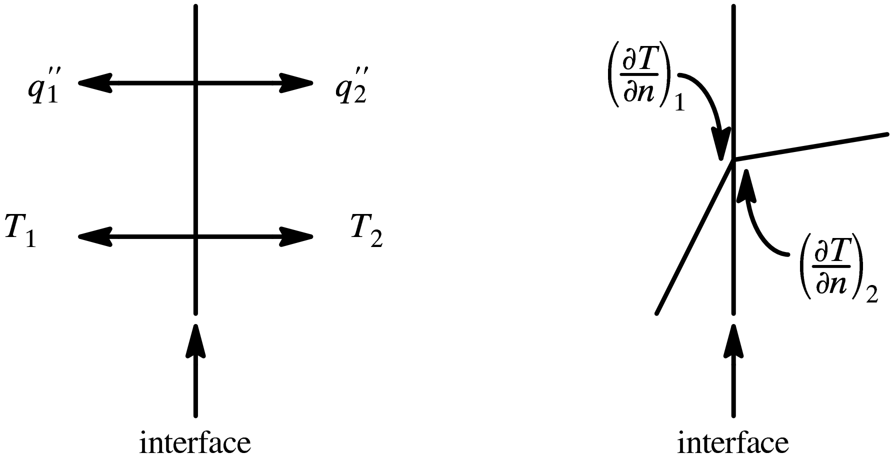
THERMAL COUPLING OF GRID BLOCKS

The impetus of this work is to expand the present CFD flow solver capability by including the physical effects of thermal communication between a fluid and a solid boundary. A study of physics of thermal energy transport has been necessary for this, as well as the consideration of numerical issues, especially at the interface. Like the fluid, the geometry of the solid may require multiple grid blocks to be constructed. Also, the solid may be composite in nature; i.e., the solid may consist of multiple materials. Accommodating this latter case requires multiple grid blocks within the solid geometry. Parallelization, though not attempted in this work, is yet another scenario requiring multiple grid blocks. Block–to–block communication of the flow solver alone has been addressed by Belk [4]. This work will address the block–to–block communications between the flow solver and the heat conduction solver and within conduction solver. Though the conduction problem is, in general, simpler than the flow problem (only a scalar variable is solved for at each grid cell), communication among grid blocks can offer problems.

The first topic to discuss is the criteria which must be met in order to successfully communicate (thermally) between grid blocks. Since each block is worked on separately, a coupling is necessary to insure accurate solutions. The two main conditions to be met are continuity of heat flux across the block interface and continuity of temperature values on the interface. That is, (with subscripts 1 and 2 to represent the two sides of an interface):

$$(q'')_1 = (q'')_2 \;;\; T_1 = T_2$$

The two conditions from the previous page are imposed on each grid cell along the common interface. Though the heat flux and temperature are continuous across the boundary, the temperature derivatives normal to the boundary will be discontinuous for solid–fluid interfaces and for composite solid–solid interfaces (see Figure 4.1 below). Of course, for a solid–solid interface with like material on both sides of the interface the temperature derivatives, too, will be continuous. For the case of discontinuous temperature derivatives, the ratio of these derivatives may be determined from the heat flux continuity (again, see Figure 4.1). An assumption that needs to be stated at the beginning is that of perfect contact along the interface of composite walls. That is, there are assumed to be no regions, even on a microscopic level, in which the two materials are not in contact. In reality, this is often not true. This assumption allows the continuity of heat flux to be applied at each cell along the length of the interface.


q_1''
q_2''
T_1
T_2
interface
$\left(\frac{\partial T}{\partial n}\right)_1$
$\left(\frac{\partial T}{\partial n}\right)_2$
interface

$$q_1'' = q_2'' \Rightarrow \kappa_1\left(\frac{\partial T}{\partial n}\right)_1 = \kappa_2\left(\frac{\partial T}{\partial n}\right)_2 \qquad \therefore \; \frac{(\partial T/\partial n)_1}{(\partial T/\partial n)_2} = \left(\frac{\kappa_1}{\kappa_2}\right)^{-1}$$

Figure 4.1 Continuous Heat Flux and Temperature (left); Discontinuous Temperature Derivatives (right)

Solid–Solid Block Coupling

The case of solid–solid block interfaces, with the blocks composed of both like and different materials, will now be discussed. The coupling of such blocks along their interfaces can be found entirely within that part of code written strictly for conduction problems. This conduction solver has been incorporated into the original CFD code to work either independently of, or in cooperation with the flow solver.

It should be noted that the temperature derivatives of each block at the interface may differ by orders of magnitude, depending on the ratio of the respective conductivities. This fact can certainly affect the grid spacings on the respective sides of the interface, in that one side may require much tighter spacing to adequately resolve the derivative. Adequate resolution of the temperature derivative is obviously necessary for accuracy, but grid spacing may also be required to help with stability.

With regard to the transient behavior of conduction within a solid, the property of thermal diffusivity is of key importance. Thermal diffusivity, α, is defined as

$$\alpha = \frac{\kappa}{\rho c} \quad . \tag{4.1}$$

The product of ρc in the denominator is often referred to as the thermal capacitance or thermal "mass". It plays a role similar to that of inertial mass in Newton's 2nd law; i.e., the greater the value of thermal mass, the more time is required for the thermal effects to penetrate through the solid. For a composite wall, this means that one one side of the interface will "advance" at a different rate than the other, thermally speaking. The difference in these rates of advancement depends directly on the ratio of the respective diffusivities. In other words, for a given time step interval, the temperature field of one side of the interface will change by a greater, or lesser, amount than the other, depending on the ratio of diffusivities across the interface.

The diffusivity can also be viewed to act in much the same way as the wave speed for the convective terms. The dimensionless CFL number for wave–type equations is defined as the ratio of (a*t)/L, where “a” is some wave speed, “t” is time, and “L” some value of length; similarly the dimensionless Fourier number, Fo, of diffusion problems is defined as the ratio of $(\alpha^* t)/L^2$. For both computational wave and diffusion problems, the CFL and Fourier numbers, respectively, are measures of how fast a particular problem is being advanced; these two numbers may also be gauges of stability. Consequently, thermal diffusivity is very important for transient problems, as well as steady–state problems that are solved using the unsteady equation, as is often the case.

Composite Slab

For the composite solid wall, the code is constructed such that the lower–numbered block uses the interface temperature supplied by the higher–numbered block as a Dirichlet boundary condition. Based on this interface temperature and that of the center of the cell adjacent to the interface for the lower–numbered block, a 1st–order, one–sided difference is calculated to approximate the temperature derivative at the interface. Using this approximated value for the temperature derivative and the conductivity, κ, of the material in the lower–numbered block, a heat flux is calculated and placed in the proper memory location for use by the higher–numbered block as a Neumann boundary condition. The memory allocation for the block–to–block conduction problem is set up in the same manner and using the same array as that formed by Arabshahi [1] for block–to–block situations within the flow solver.

Again, the temperatures used for this block–to–block communication are from the previous time step, so an explicit (i.e., lagged) boundary condition exists. Because of this “explicitness” at the interface, a stability restriction will exist, limiting the size of the time step to be used. As an example, the orders of accuracy along a constant–i interface are $[(\Delta\tau), (\Delta\xi), (\Delta\eta)^2, (\Delta\zeta)^2]$. Increasing the time accuracy from 1st to 2nd–order by means of

3–step, backward differencing can allow for increased values of the time step size while maintaining numerical stability.

The general algorithm for the conduction problem on a composite wall or slab is shown in the figure below.

```
Do ib = 1, no. of blocks
   Obtain interface BC's (temp. for lower no. block, q" for higher no. block)
   Obtain "regular" BC's
   If lower no. block then
      Calculate q" and send to higher no. block
   End If
   Calculate RHS of equation
   Calculate LHS of equation, if necessary
   Do isgs = 1, SGS LIMIT
       Forward sweep
       Backward sweep
   End Do
     $T^{n+1} = T^n + \Delta T^n$
   If higher no. block then
       Send to $T^{n+1}$ lower no. block at interface for next time step
   End If
 End Do
```

Figure 4.2 Basic Algorithm for Explicit Coupling of Composite Solid Blocks

Homogeneous Slab

A homogeneous wall is defined as one of like material, and, thus, like thermal properties. In this case, the primary reason for dividing the grid into multiple blocks is that of geometrical complexity. Even though the single block grid would be best (assuming the computer has sufficient memory), the geometry often makes such a grid impractical. The question of block–to–block interfaces then becomes quite important.

The same procedure described in the previous section and illustrated in Figure 4.2 could be used for the homogeneous wall as well. However, because of the common material on both sides of the interface, there will be no discontinuity in temperature derivatives at the block–to–block boundary. Therefore, one might think that there is no need to pack grid points perpendicularly to the boundary for the purpose of improving accuracy. As it turns out, packing at the boundary is still needed. The problem of heat conduction in a solid is a boundary–value problem; i.e., the solution in the domain will be influenced by all boundaries of the domain. A block–to–block boundary treated explicitly means there is a "barrier" within the domain which keeps all of the grid cells from being correctly influenced by all of the boundaries. And the commonality of thermal properties means this diffusion process progresses at the same pace for both sides of the interface. This combination of explicit block–to–block boundary and same rate of advance of the physical problem makes for a very "unforgiving" situation. Any errors in calculation of the interface heat flux/temperature are quickly amplified; a non–physical solution results in relatively few time–step iterations. So, a very tight packing may be required in order to make the interface solution stable; either that or a time step value that is possibly so small as to be impractical.

Interface grid packing with like material on either side is undesirable, especially with a structured grid. As with fluid blocks, this packing can possibly "propagate" to the outer grid boundary. A way to avoid the requirement of tight grid packing at the block–to–block boundary is through the choice of the temperature used for the boundary condition of the interface's lower–numbered block. Suppose for a moment that the block interface did not exist. The cell boundary coinciding with the block interface would use the temperature at the cell centers on either side of the cell boundary and a 2nd–order, central–difference approximation to calculate the heat flux at the cell boundary. Assume, now, that the interface boundary is again present; rather than use an interface temperature supplied by the higher–numbered block, use the temperature from the cell centers adjacent to the interface from this higher–numbered block. This will still result in an explicit boundary, so there will still be the so–called conditional stability. However, the stability is increased some while avoiding an excessively tight grid spacing along the interface boundary.

Both of the previously discussed ways to treat the interface boundary of a homogeneous wall or slab are based on the procedure for the composite wall (i.e., the algorithm in Figure 4.2), and both of the possible treatments result in an explicit block–to–block boundary and an accompanying stability problem. A method in which a homogeneous solid block could have implicit interface boundaries would be most desirable. As it turns out, such a method is possible while making a relatively minor modification to the algorithm presented in Figure 4.2.

Let us first look at a single–block, 2–D slab such as that on the left side of Figure 4.3. The i–coordinate increases from left to right, the j–coordinate from bottom to top. Therefore, the forward sweep is "to the right and up", the backward sweep "to the left and down". For the single block case, the coefficient matrix of the LHS will apply to all grid cells in the interior domain; the entire domain is solved simultaneously. Suppose now that the single–block grid is divided into two blocks. The forward sweep is done on block 1, but instead of calculating a heat flux at the interface and passing that to block 2 for use as a boundary condition, the ΔT^n values for the cells adjacent to the interface are passed to block 2. The code then goes to block 2 and completes the forward sweep of the grid using the information passed from block 1. This procedure is then reversed for the backward sweep. The backward sweep is performed on block 2, and the ΔT^n values for its cells adjacent to the interface are passed to block 1. The code goes to block 1 using the information from block 2 and completes the backward sweep of the grid. The entire procedure constitutes a single Gauss–Seidel iteration, which is executed for the prescribed number of times.

The procedure described above can be accomplished without much effort by rearranging the looping structure of the algorithm of Figure 4.2. The position of the Gauss–Seidel and blocking loops is reversed; i.e., the Gauss–Seidel loop is now "outside", rather than "inside" the blocking loop. This new algorithm is shown in Figure 4.4. A very similar procedure was described by Belk [4] for block–to–block communication of fluid blocks (without the Gauss–Seidel iteration, only one forward–backward sweep combination was performed at that time). Also, a communication with Pankajakshan [40] revealed that a similar procedure is used for a parallelized, structured flow solver, similar in that the values passed from block to block are from the most recent Gauss–Seidel iteration (this, of course, is necessary in a parallel code since the numerous blocks cannot "wait" to receive the most recently updated data from all of the neighboring blocks). So the procedure just described is based somewhat on work that has been done by others for different purposes than that of thermal block–to–block communication.

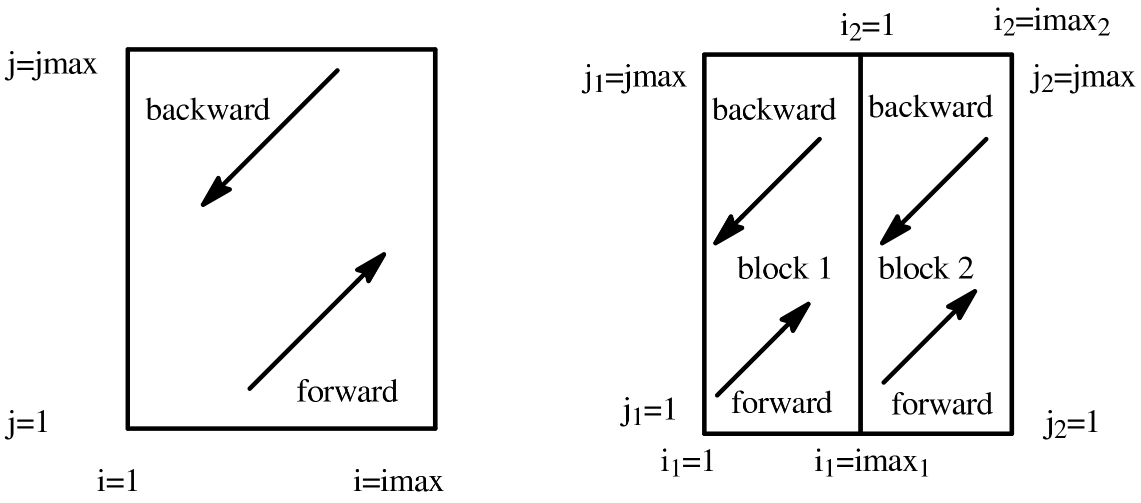


Figure 4.3 Implicit Treatment of Multiple–Block Grids

In summary, the block–to–block thermal communication of solid grid blocks can be prone to problems. The heat conduction problem is one in which all points within the domain are related, to a greater or lesser degree, and any hindrances to adequate communication may result in numerical instability. The block–to–block boundaries are usually the source of these instabilities understandably, and the greatest attention has to be given to these boundaries. The ratio of thermal conductivities plays a role in the amount of grid packing required normal to the boundary; the ratio of thermal diffusivities gives a measure of how “fast” the respective blocks change or react to changes in temperature, heat flux, and so on.

Fluid–Solid Block Coupling

The case issues involved in the thermal coupling of fluid and solid grid blocks are to a large degree the same as that of the composite wall described previously. Continuity of wall heat flux and temperature are maintained, but temperature derivatives at the interface are discontinuous; often differing by orders of magnitude. For example, the conductivity of aluminum, often used for aerospace structures, is 237 W/(m–K) at 300K; air at the same temperature and atmospheric pressure has a conductivity of only 0.0263 W/(m–K). The ratio of the conductivity of aluminum to that of air in this case is just over 9000. Even for a non–metallic solid with a much lower conductivity, such as fused quartz (κ = 1.38 W/(m–k)), the conductivity ratio is still in excess of one order of magnitude, at approximately 52. Thermal diffusivities, on the other hand, often differ by less than an order of magnitude (97.1e–06 m^2/sec and 22.5e–06 m^2/sec for aluminum and air, respectively, at the same conditions). All of these values are taken from Appendix A of Incropera and DeWitt [41].

Grid packing on the fluid side of the interface is usually going to be much tighter than on the solid side, especially if the fluid is in a gaseous state. Of course, tight point spacing and relatively high grid–point density are usually required for viscous flows anyway, especially for high–Reynolds number flows (i.e., flows with very thin velocity boundary layers and, thus, very large velocity derivatives at the interface).

The question to be addressed is which type of boundary condition, heat flux or temperature, is applied to which side of the fluid–solid boundary. In the case of the composite wall interface, the lower–numbered block used the temperature supplied from the higher–numbered block as a boundary condition. The lower–numbered block calculated a wall heat flux and supplied that valued to the higher–numbered block for its boundary condition. Block numbering does not determine the type of interface boundary condition for the respective blocks of the coupled fluid–solid case. The code is constructed such that the solid blocks supply the interface temperature to the fluid blocks; the fluid blocks calculate a wall heat flux and send that value to the solid blocks for use as their boundary condition.

The sequential nature of the code means that grid blocks are worked on in an ascending order; i.e., the block loop starts with block 1 and progresses to the “n^{th}” block. The previous paragraph described how the fluid and solid blocks communicate in the coupled problem. Like the composite wall, these interface conditions are explicit in nature. For these conditions to be consistent (or “synchronized” as described by Belk [4]), all fluid grid blocks need to be worked on before any of the solid blocks. Certainly the most straightforward way of doing this is to have all fluid blocks numbered smaller than any solid block. This numbering scheme is easily accomplished for relatively small, simple grids. For grids of higher block number counts and greater complexity the requirement of all fluid blocks to have smaller numbers than all solid blocks could become impractical.

An easy modification is made to the existing code so that an “identification” is given to each grid block with respect to its type; each block has a logical variable that identifies it as a fluid block (fluid flow with no thermal coupling to a solid boundary), a fluid/ solid block (fluid flow with thermal coupling to a solid boundary), or a solid block (heat conduction coupled either to other solid blocks or to fluid/solid blocks). This identification takes place in the initialization subroutine, while reading the boundary conditions of the respective blocks. A block loop is marched through within the main (time) iteration loop of the code, and all the fluid blocks are worked on before any solid blocks, using this identification variable. A second block loop is then executed (within the same time iteration loop), but only the solid blocks are worked on in this loop. A Newton sub–iteration loop, if necessary, is placed between the time loop and the block loops, as shown in Figure 4.5, for example. Therefore, all fluid blocks, and the associated wall heat fluxes, will be calculated at any coupled fluid–solid interfaces before any solid blocks are executed, regardless of block numbering. This identification system also means that uncoupled problems (i.e., either the flow solver or the heat conduction solver alone) may be simulated while using the same code. The type of problem to be worked on is described entirely within the input file by the definition of the boundary conditions. A greatly simplified version of this algorithm is shown in Figure 4.5 on the following page.

```
Do time–step iteration
    Do Newton iteration
        Do isgs = 1, SGS LIMIT
            Do ipass = 1, 2
                Do ib = 1, no. of blocks
                    If (isgs=1 and ipass=1) then
                        Obtain BC's
                        Calculate RHS of equation
                        Calculate LHS of equation, if necessary
                    End If
                    If (ipass=1) then
                        Obtain $\Delta T^n$ from lower–numbered block(s)
                        Forward sweep
                        Pass $\Delta T^n$ to higher–numbered block(s)
                    Else If (ipass=2) then
                        Obtain $\Delta T^n$ from higher–numbered block(s)
                        Backward sweep
                        Pass $\Delta T^n$ to lower–numbered block(s)
                    End If
                End Do (block loop)
            End Do (ipass loop)
        End Do (isgs loop)
        $T^{n+1} = T^n + \Delta T^n$
    End Do (Newton loop)
End Do (time–step loop)
```

Figure 4.4 Basic Algorithm for Implicit Coupling of Homogeneous Solid Blocks

```
Do time–step iteration
    Do Newton iteration
        Do ib = 1, no. of blocks
            If fluid block then
                Simulate fluid flow
            End If
        End Do
        Do ib = 1, no. of blocks
            If solid block then
                Simulate heat conduction
            End If
        End Do
    End Do (Newton Loop)
 End Do (time–step loop)
```

Figure 4.5 Basic Algorithm for Explicit Coupling of Fluid and Solid Blocks

CHAPTER V

RESULTS AND DISCUSSION

Validation of the Heat Conduction Solver

The heat conduction solver is a new addition to the overall CFD code and needs to have some degree of validation before use in conjunction with the flow solver. Some relatively simple test cases on one– and two–dimensional grids will be used to demonstrate that the conduction code is working properly. These test cases all assume constant thermophysical properties, which give a linear heat equation. Also, these tests have analytical solutions against which comparisons can be made.

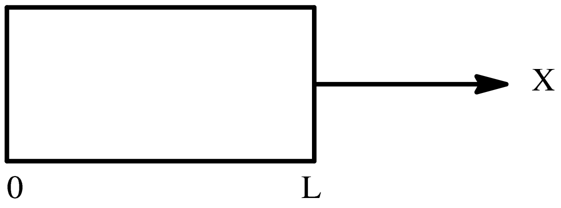
One–Dimensional Test Cases

The first test case is that of a one–dimensional, transient conduction problem. This test case is an example problem taken from the text of Incropera and DeWitt [41]. The example in the text is that of a semi–infinite slab of pure copper exposed to a constant radiative heat flux boundary condition. The analytical solution of such a problem is known, and will be presented later.

For the purpose of code validation , a model of a “bar” of rectangular cross section is used instead of the semi–infinite slab. This bar is horizontal with the left face having the radiative heat flux boundary condition and the remaining five grid surfaces treated as adiabatic. Though this bar is of finite length, it is reasoned to behave like the semi–infinite slab, at least for some period of time after the start of the problem.

Since this bar model is of finite length, the analytical solution for the semi–infinite slab is not directly applicable. The analytical method for solving the non–homogeneous, boundary–value problem in a finite, Cartesian space is taken from the text of Özisik [42]; this method is based on the Fourier Integral Transform method of solving partial differential equations (PDE's). The general expression of this problem is:

$$
\begin{aligned}
&\frac{1}{\alpha}\frac{\partial T}{\partial t} = \frac{\partial^2 T}{\partial x^2} + \frac{g(x,t)}{\kappa} \ ; \ 0 \le x \le L \ , \ t > 0 \ , \\
-\ &\kappa_1 \frac{\partial T}{\partial x} + h_1 T = f_1(t) \ ; \ x = 0 \ , \ t > 0 \ , \\
&\kappa_2 \frac{\partial T}{\partial x} + h_2 T = f_2(t) \ ; \ x = L \ , \ t > 0 \ , \\
&T = F(x) \ ; \ 0 \le x \le L \ , \ t = 0 \ .
\end{aligned}
\tag{5.1}
$$



where $\kappa_1, \kappa_2, h_1, h_2, f_1, f_2$ are prescribed boundary properties and conditions

The general analytic solution for this problem is:

$$
T(x,t) = \sum_{m=0}^{\infty} e^{-\alpha\beta_m^2 t} \cdot K(\beta_m,x) \cdot \left[\overline{F}(\beta_m) + \int_{t'=0}^{t} e^{\alpha\beta_m^2 t'} \cdot A(\beta_m,t')dt' \right] . \tag{5.2}
$$

There are a number of terms in the above equation that need to be further defined. The integral transform method works by transforming the PDE to an ordinary differential equation (ODE) in terms of some new transformed variable. This new ODE takes the form of the Sturm–Liouville eigenvalue problem; the β_m's are the eigenvalues of this problem. The term $K(\beta_m,x)$ is defined as the transform kernel and depends on the combination of boundary conditions (as do the β_m's) for the given problem. The general "boundary conditions of the third kind" above in Equation (5.1) mean that there are nine possible combinations of $K(\beta_m,x)$ and β_m's (see pages 50–51 of Özisik [42] for a listing of these combinations). A volumetric heat source (or sink) is represented by g(x,t). The terms $\overline{F}(\beta_m)$ (initial condition) and $A(\beta_m,t)$ (boundary conditions) are defined as

$$\begin{aligned} \overline{F}(\beta_m) &\equiv \int_{x'=0}^{L} K(\beta_m,x')\cdot F(x')dx' \\ A(\beta_m,t) &\equiv \alpha\left[\left[\frac{K(\beta_m,x)}{\kappa_1}\right]_{x=0}\cdot f_1(t) + \left[\frac{K(\beta_m,x)}{\kappa_2}\right]_{x=L}\cdot f_2(t)\right] \end{aligned} \tag{5.3}$$

A separate code was written to evaluate Equation (5.2). Results from this code are taken to be the analytical solution that will be compared to the results from the heat–conduction solver. For this specific test case, the following assumptions are made to the general problem of Equation (5.2):

$$\begin{aligned} &g(x,t) = h_1 = h_2 = f_2\,(t) = 0, \\ &\kappa_1 = \kappa_2 = 1, \\ &f_1\,(t) = c1 = \text{constant}, \\ &F(x) = c2 = \text{constant}. \end{aligned} \tag{5.4}$$

By specifying κ_1 and κ_2 to equal 1, temperature derivatives rather than heat fluxes are given at the boundaries. The boundary condition combination is that of a Neumann condition (BC of the second kind) at each boundary. This combination means the transform kernel will be

$$K(\beta_m,x) = \sqrt{\frac{2}{L}}\cos(\beta_m x) \;\; ; \;\; K(\beta_0,x) = \sqrt{\frac{1}{L}}\cos(\beta_0 x)$$

where the eigenvalues are the positive roots of

$$\begin{aligned} &\sin(\beta_m L) = 0 \\ &\therefore \beta_m = \frac{m\pi}{L} \;\; , \;\; m = 0, 1, 2, \ldots \end{aligned}$$

The remaining specifications of physical length of the rectangular bar, diffusivity, and initial and boundary conditions for the example problem are

bar length, L, = 1 m,
κ = 401 W/(m–K), c = 385 J/(kg–K), ρ= 8933 kg/m^3 (pure copper),
α = 117e–06 m^2/sec,
$q''|_{x=0}$ = 300 kW/m^2 => $\partial T/\partial x|_{x=0} = f_1 \cong -748$ K/m,
$T_{initial}$ = 293.15 K (20 °C).

These values along with the desired locations (x and t) for the temperature are input for the approximation of the analytical solution, approximate in that the summation cannot be carried to infinity. The question of the value, N, at which to truncate the summation was answered by simple trial–and–error type iteration. Various values for N ranging from 100 to 10000 were tried. The N–value of 10000 was picked, based on the fact that by this point no appreciable difference was noticed by increasing N.

A simple grid of size 101 x 2 x 2 was used by the heat conduction code. The spacing was uniform, giving a grid–cell length of 0.01 m; the grid width and height were set to 0.075 m. A time step of 1 second was used, resulting in a grid–cell Fourier Number of 1.17 (based on a length of 0.01 m and the above physical properties); this is just over twice the explicit limit of 0.5. The test lasted for 120 seconds. The time and space accuracies are 1st– and 2nd–order, respectively, for all cases, unless otherwise noted.

The analytic solution of one–dimensional heat conduction within a semi–infinite slab with constant heat flux, q_0'', at the exposed boundary is given by:

$$T(x,t) = \frac{2q_0''\sqrt{\frac{\alpha t}{\pi}}}{\kappa}\exp\left(-\frac{x^2}{4\alpha t}\right) - \frac{q_0'' x}{\kappa}\, erfc\left(\frac{x}{2\sqrt{\alpha t}}\right) + T_{initial} \, , \qquad (5.5)$$

where $$erfc\ w \equiv 1 - erf\ w \ ; \ erf\ w \equiv \frac{2}{\sqrt{\pi}}\int_0^w e^{-v^2}dv \ .$$

A set of tables will now be constructed comparing the results from the above equation with those of Equation (5.2) and the heat–conduction solver at three different spatial locations and five different points in time (24–second intervals, t = 0 not included). Values of the error function, erf, for use in evaluating the exact solution are taken from Appendix B.2 of Incropera & DeWitt [41]. The temperature values are in degrees Celsius.

Table 5.1 Comparison of Analytical and Approximate Solutions for 1–D, Transient Conduction Test Case; x = 0 m from Heat–Flux Boundary

time –>	24 sec	48 sec	72 sec	96 sec	120 sec
semi–∞ slab	64.74	83.26	97.48	109.5	120.0
finite bar	64.72	83.25	97.47	109.5	120.0
numerical	64.60	83.17	97.40	109.4	120.0

Table 5.2 Comparison of Analytical and Approximate Solutions for 1–D, Transient Conduction Test Case; x = 0.1 m from Heat–Flux Boundary

time –>	24 sec	48 sec	72 sec	96 sec	120 sec
semi–∞ slab	24.73	34.68	44.57	53.85	62.50
finite bar	24.74	34.69	44.59	53.86	62.51
numerical	24.88	34.74	44.60	53.85	62.50

Table 5.3 Comparison of Analytical and Approximate Solutions for 1–D, Transient Conduction Test Case; x = 0.2 m from Heat–Flux Boundary

time –>	24 sec	48 sec	72 sec	96 sec	120 sec
semi–∞ slab	20.12	21.73	25.17	29.46	34.24
finite bar	20.13	21.81	25.18	29.49	34.25
numerical	20.18	21.91	25.27	29.56	34.31

Table 5.4 Comparison of Analytical and Approximate Solutions for 1–D, Transient Conduction Test Case; x = 0.3 m from Heat–Flux Boundary

time –>	24 sec	48 sec	72 sec	96 sec	120 sec
semi– ∞ slab	20.00	20.11	20.63	21.84	23.52
finite bar	20.00	20.11	20.69	21.90	23.69
numerical	20.00	20.14	20.74	21.96	23.73

The entries in each of the above tables show very good agreement between the exact values and the approximate results. The results from the approximation of Equation (5.2) suggest that this method will serve well as the analytical solution for tests that follow. Results from the conduction solver indicate that the code works properly for this test case. The results are given to four significant figures to help show just how closely the different methods compare. Accuracy to this number of significant figures is usually not available, three figures of accuracy is more common. Many temperature measurement devices have resolution to one decimal point, at best, so these results agree well within that range of measurement accuracy. One thing to note, however, is the fact that the temperatures from the approximation of Equation (5.2) and those from the heat–conduction solver are higher than those of the semi–infinite slab solution, except at x = 0. This difference is more noticeable at points farther from the left boundary and at the larger time values. This suggests that the adiabatic right boundary (x = 1 m) is having some slight effect; this effect would certainly become more pronounced if the simulation were continued. Further comparison of the solution of Equation (5.2) and the conduction solver is shown in the Figure 5.1.

A great amount of detail is being given to this relatively simple test case. The purpose is to demonstrate that both the approximation of Equation (5.2) and the heat–conduction solver compare well with a mathematically exact solution. Other tests will compare only the results of the approximate solution of (5.2) (or its two–dimensional “version”) to those of the conduction solver, with the approximate solution of (5.2) then considered to be the analytical or exact solution.

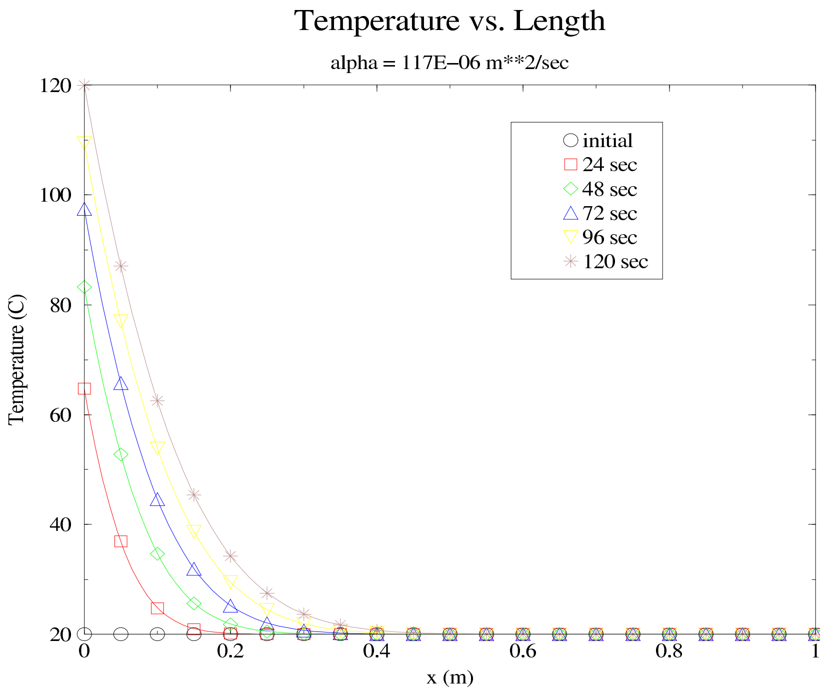


Figure 5.1 Temperature vs. X for 1–D, Transient Test Case; Fourier Transform Method (Symbols) and Heat–Conduction Solver (Curves)

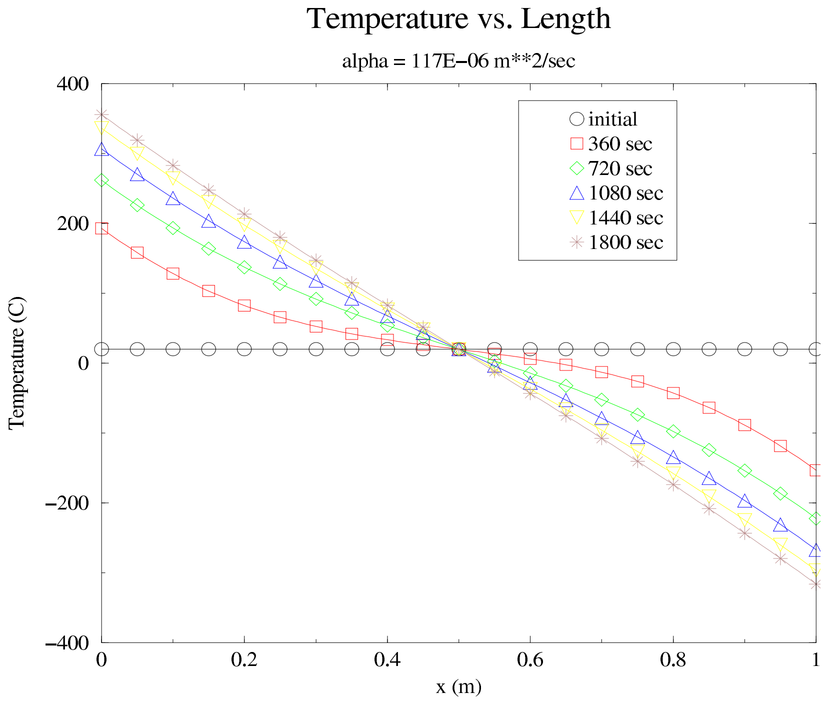


Figure 5.2 Temperature vs. X for 1–D, Steady–State Test Case; Fourier Transform Method (Symbols) and Heat–Conduction Solver (Curves)

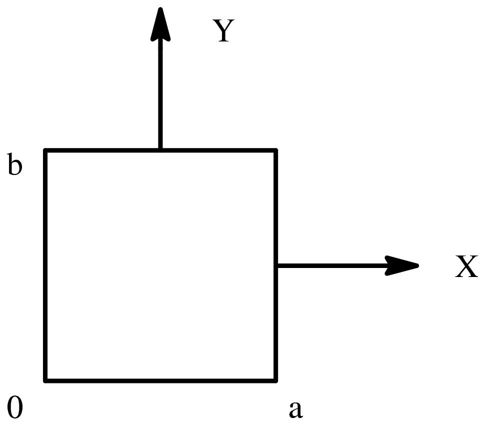
Another simple test was conducted using the same grid and problem parameters from before. This new test was run until a steady–state was reached. The previous transient test, if continued, would never reach a steady–state; the thermal effects would finally penetrate to the right boundary at x = 1 m and the temperature within the bar would then continue to increase. For this new test, a heat flux was specified to be exiting the right boundary and of the same magnitude as that entering at the left boundary. The test goes through a transient and reaches a steady–state which gives a linear plot of temperature vs bar length. The test run was for 1800 seconds, with results presented at intervals of 360 seconds (see Figure 5.2). Though the temperature range is not very realistic, the plots make physical sense, with each side mirroring the other about the center.

The initial validation of the heat–conduction solver for both a transient and a steady–state, one–dimensional problem is complete. The results from the conduction solver and from the approximated Fourier integral transform method (approximated by truncation of the infinite series) compare very well with the exact solution of the transient case. The next test cases are meant to further validate the conduction solver by simulating transient and steady–state problems in two space dimensions.

Two–Dimensional Test Cases

The Fourier integral transform method will be treated as the exact solution for the following cases. Expanding this to higher dimensions is relatively straightforward and the similarities with the one–dimensional expression are obvious. Referring to Chapter Two of Özisik [42], the general two–dimensional, time–dependent, non–homogenous boundary–value problem and the corresponding analytical expression for temperature within the domain are given on the following page. Like the one–dimensional case, $K(\beta_m,x)$ and $K(\nu_n,y)$ are the transform kernels, the β_m's and ν_n's are the eigenvalues, $\overline{F}(\beta_m,\nu_n)$ accounts

$$
\begin{aligned}
&\frac{1}{\alpha}\frac{\partial T}{\partial t} = \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{g(x,y,t)}{\kappa} \;\; ; \;\; 0 \le x \le a \;,\; 0 \le y \le b \;,\; t > 0, \\
&-\kappa_1 \frac{\partial T}{\partial x} + h_1 T = f_1(y,t) \;\; ; \;\; x = 0 \;,\; t > 0, \\
&\kappa_2 \frac{\partial T}{\partial x} + h_2 T = f_2(y,t) \;\; ; \;\; x = a \;,\; t > 0, \\
&-\kappa_3 \frac{\partial T}{\partial y} + h_3 T = f_3(x,t) \;\; ; \;\; y = 0 \;,\; t > 0, \\
&\kappa_4 \frac{\partial T}{\partial y} + h_4 T = f_4(x,t) \;\; ; \;\; y = b \;,\; t > 0, \\
&T = F(x,y) \;\; ; \;\; 0 \le x \le a \;,\; 0 \le y \le b \;,\; t = 0.
\end{aligned}
\tag{5.6}
$$



$$
T(x,y,t) = \sum_{m=0}^{\infty}\sum_{n=0}^{\infty} e^{-\alpha(\beta_m^2+\nu_n^2)t} \cdot K(\beta_m,x) \cdot K(\nu_n,y) \cdots
\tag{5.7}
$$

$$
\cdots \left[\overline{F}(\beta_m,\nu_n) + \int_{t'=0}^{t} e^{\alpha(\beta_m^2+\nu_n^2)t} \cdot A(\beta_m,\nu_n,t')\,dt' \right].
$$

for the initial condition and $A(\beta_m,\nu_n,t)$ accounts for the boundary conditions. $\overline{F}(\beta_m,\nu_n)$ and $A(\beta_m,\nu_n,t)$ (for $g(x,y,t) = 0$) are defined on the next page.

A steady–state problem will be the first two–dimensional test case. The physical domain is a square 1 m x 1 m with $g(x,y,t) = \kappa_1 = \kappa_2 = f_4(x,t) = 0$ and $h_1 = h_2 = 1$; also,

$$\overline{F}(\beta_m,\nu_n) \equiv \int_{x'=0}^{a}\int_{y'=0}^{b} K(\beta_m,x') \cdot K(\nu_n,y') \cdot F(x',y')dx'dy'$$

$$\begin{aligned}\frac{A(\beta_m,\nu_n,t')}{\alpha} \equiv {} & \left(\frac{K(\beta_m,x)}{\kappa_1}\right)_{x=0} \cdot \int_{y'=0}^{b} K(\nu_n,y') \cdot f_1(y',t') \cdot dy' + \cdots \\ \cdots + {} & \left(\frac{K(\beta_m,x)}{\kappa_2}\right)_{x=a} \cdot \int_{y'=0}^{b} K(\nu_n,y') \cdot f_2(y',t') \cdot dy' + \cdots \\ \cdots + {} & \left(\frac{K(\nu_n,y)}{\kappa_3}\right)_{y=0} \cdot \int_{x'=0}^{a} K(\beta_m,x') \cdot f_3(x',t') \cdot dx' + \cdots \\ \cdots + {} & \left(\frac{K(\nu_n,y)}{\kappa_4}\right)_{y=b} \cdot \int_{x'=0}^{a} K(\beta_m,x') \cdot f_4(x',t') \cdot dx'\end{aligned} \qquad (5.8)$$

$f_1(y,t) = f_2(y,t) = 300$ K, $f_3(x,t) = -50$ K/m, $\kappa_3 = \kappa_4 = 1$ and $h_3 = h_4 = 0$. These parameter values indicate that the boundaries of x = 0 and x = a (i.e., 1) are constant Dirichlet conditions of 300 K. The boundaries of y = 0 and y = b are constant Neumann conditions with a temperature derivative of –50 K/m (out flow of heat) at y = 0 and an adiabatic condition at y = b. The initial condition is a uniform temperature of 300 K. Since $\kappa_1 = \kappa_2 = 0$, the first two terms in the above definition of $A(\beta_m,\nu_n,t')$ will be undefined. A substitution is made whenever $\kappa = 0$ and requires the derivative of the transform kernel with respect to the spatial variable associated with the particular boundary. For the present example, the two required substitutions are defined as

$$\left(\frac{K(\beta_m,x)}{\kappa_1}\right)_{x=0} \rightarrow \frac{1}{h_1}\left(\frac{dK(\beta_m,x)}{dx}\right)_{x=0} \text{ and } \left(\frac{K(\beta_m,x)}{\kappa_2}\right)_{x=a} \rightarrow -\frac{1}{h_2}\left(\frac{dK(\beta_m,x)}{dx}\right)_{x=a}$$

The boundary conditions of this example are Dirichlet–Dirichlet in the x–direction and Neumann–Neumann in the y–direction. The resulting transform kernels are:

$$K(\beta_m,x) = \sqrt{\frac{2}{a}}\sin(\beta_m x),$$
$$K(\nu_n,y) = \sqrt{\frac{2}{b}}\cos(\nu_n y) \;;\; K(\nu_0,y) = \sqrt{\frac{1}{b}}\cos(\nu_0 y) \tag{5.9}$$

where the eigenvalues are the positive roots of

$$\sin(\beta_m a) = 0 \quad \text{and} \quad \sin(\nu_n b) = 0$$
$$\therefore \beta_m = \frac{m\pi}{a} \;,\; m = 0,1,2,... \;;\; \nu_n = \frac{n\pi}{b} \;,\; n = 0,1,2,... \tag{5.10}$$

This example will reach a steady state because of the (constant and balanced) boundary conditions. Temperature plots vs the x–direction (at some constant y–direction value) will be symmetric about the centerline of the slab. Also, to further demonstrate the effect of the magnitude of thermal diffusivity on the physical time scale of the problem, the diffusivity for copper will be increased by a factor of 1000 from 0.000117 to 0.117 (m^2/sec). Diffusivity values of this magnitude are not realistic for most common metallic solids (please refer to Appendix A of Incropera and DeWitt [41]). However, this is of no consequence for the purpose of this test case in that no comparisons are being made to an actual experiment. This test is only for further verification of the heat–conduction solver, and so good agreement between the solver and Equation (5.7) is all that is sought.

Another small code (similar to that for Equation (5.2)) was written to approximate Equation (5.7). In the one–dimensional case, the summation was truncated at m = 10000. For this two–dimensional case, the summations over m and n will each be carried to 5000. This will compromise the solution a little, but it is felt necessary in order to have reasonable computer run times. For example, if the summations over m and n are both carried to 10000, then the inner most loop (summation over n) will be executed 10^8 times for each point in space and time where a solution is desired. A run time of approximately 16 minutes on an SGI R10000 processor for 21 different values of x, 3 different values of y, and only 1 value of time, t. Summing m and n to 10000 would increase this run time by roughly a factor of 4, so that about an hour would be required to get temperature results for each value of time, with the same number of x and y values.

The heat–conduction solver uses a grid of size 71 x 71 x 2 to discretize the 1 m x 1 m physical domain; equal spacing gives grid cells that are approximately 1.43 cm x 1.43 cm with a depth of 1 cm. A minimum time step of 0.01 seconds is used to march the solution to its steady state. Thermal diffusivity is not an input, per se, for the conduction solver. The density, specific heat, and thermal conductivity are input for the given solid material. The conductivity and specific heat of aluminum (167 W/(m–K) and 883 J/(kg–K), respectively) were used, and the density was adjusted to a value of approximately 1.62 kg/m^3 to match the diffusivity value for the analytical method above. A diffusivity of 0.117 m^2/sec, time step of 0.01 sec, and cell length L = 0.0143 m results in a Fourier number of about 5.73, which is roughly 23 times the explicit–limit value of 0.25 for two–dimensional problems. The boundaries at x = 0 and x = 1 m are specified to be at the constant temperature of 300 K. The boundary at y = 0 is a constant heat flux of –8350 W/m^2 (this will match the temperature derivative specified for the analytical problem); that at y = 1 m is an adiabatic wall.

The plots for this steady–state example are shown in the next three figures. Each is a graph of temperature vs x–direction at constant y–direction values of 0, 0.5 and 1.0 m, respectively. Each curve or group of symbols is for a different point in time; these plots show the transient of this problem as it advances to the steady state. As before, the symbols represent the analytical results (Equation (5.7)), and the solid curves are for the conduction solver results. All of the plots have the same minimum and maximum values of temperature to give a visual impression of the relative change experienced at the different y–direction values within the slab.

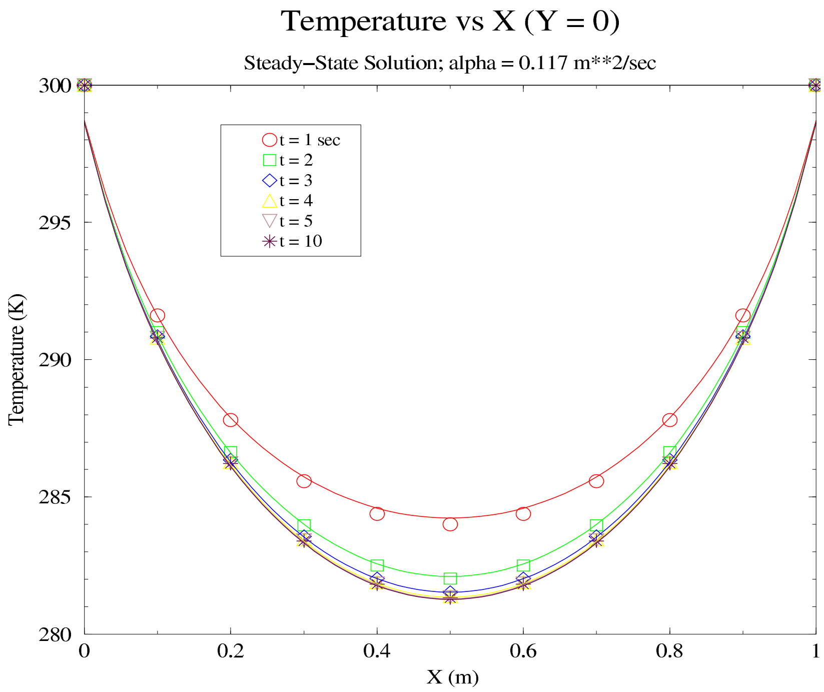


Figure 5.3 Temperature vs. X for 2–D, Steady–State Test Case; Fourier Transform Method (Symbols) and Heat–Conduction Solver (Curves); Y = 0 m

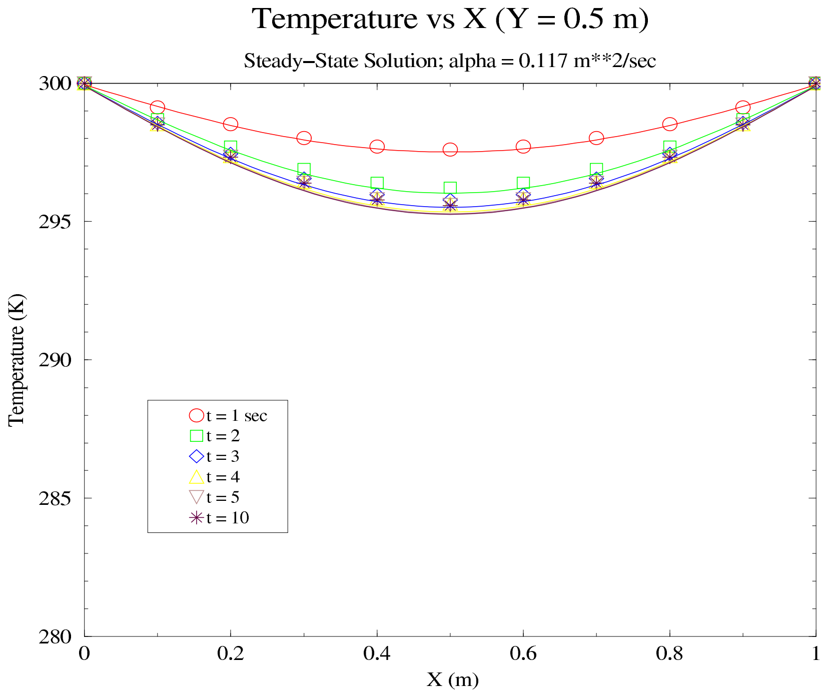


Figure 5.4 Temperature vs. X for 2–D, Steady–State Test Case; Fourier Transform Method (Symbols) and Heat–Conduction Solver (Curves); Y = 0.5 m

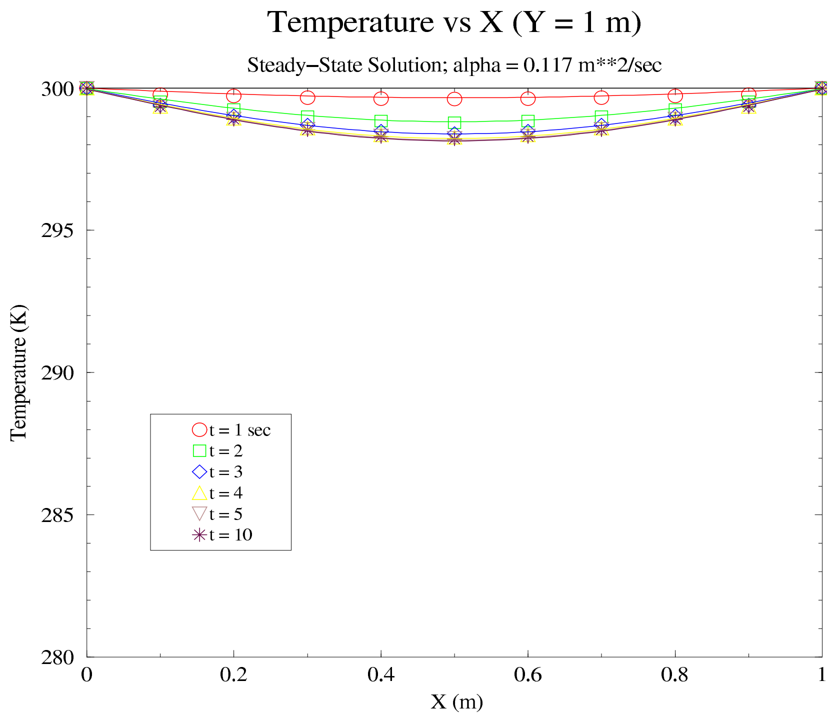


Figure 5.5 Temperature vs. X for 2–D, Steady–State Test Case; Fourier Transform Method (Symbols) and Heat–Conduction Solver (Curves); Y = 1.0 m

The greatest change in temperature takes place at the boundary of y = 0, due to the boundary condition; heat is flowing out of the slab, so the temperature drops to a minimum at the midpoint of the x–direction. Similar patterns are seen at y = 0.5 m and y = 1 m, but to lesser degrees. Very little change is seen at the boundary of y = 1 m due to its adiabatic condition. In all plots, the majority of the change takes place in the first three seconds; the temperature drop is less than one degree from three to ten seconds. The solutions of the conduction solver disagree with the specified the condition of 300 K at the x = 0 and x = 1 m boundaries. This is due to the simple averaging (implemented for the plotting software) of the cell–centered and interface temperatures surrounding a given boundary node. The plots at y = 0.5 m and y = 1 m agree better at these boundaries, because there is much less temperature change at the x–boundaries of these y–locations. Also notice that the temperature derivatives at the x–boundaries interior to y = 0.5 m and y = 1 m are much smaller than at y = 0. Since the two x–boundaries are maintained at a constant temperature, the heat flux will adjust accordingly, resulting in the smaller magnitude of the x–wall temperature derivatives.

In summary, these results show good agreement between the results of conduction solver and Equation (5.7). The maximum discrepancy appears to be no worse than 0.5 degrees in magnitude at any point, and this seems acceptable considering the use of first–order time accuracy and a moderate time–step for the conduction solver.

Next an unsteady, two–dimensional case was run. The parameters used were similar to those before, the exception being the boundary condition at x = 1 m. Before, this boundary condition was set to a constant temperature of 300 K; this will be changed to a Neumann condition that is a sinusoidal function of time (same at all y–locations). The magnitude of the derivative will be the same as that for the y = 0 boundary from the prior case, with a frequency of 1.25 Hz (i.e. $dT/dx|_{x=1} = 50 \sin \omega t$ K/m , $\omega = 2.5\pi$ $\sec^{-1}$).

The change in the boundary condition will also mean a change in the analytical expression for T(x,y,t). The transform kernels remain unchanged, as do the ν–eigenvalues; the change is to the β–eigenvalues. The new β–eigenvalues are the positive roots of:

$$\cos(\beta_m a) = 0 \Rightarrow \beta_m = \left[(2m + 1)\pi\right]/(2a) \ , \ m = 0, 1, 2, ... \qquad (5.11)$$

Yet another small code to approximate Equation (5.7) is needed as a result of the new time–dependent boundary condition. The summations will again be stopped at 5000 for both the m and n eigenvalue parameters.

The same grid will be used for the conduction solver. The x = 1 m boundary condition was changed, with the heat flux at this boundary set to $q''_w|_{x=1} = 8350 \sin \omega t$ W/m^2 , $\omega = 2.5\pi$ $\sec^{-1}$. The time step was reduced from 0.01 to 0.002 seconds, resulting in a Fourier number of about 1.15, and the test run from t = 0 to t = 5 seconds.

Figure 5.6 through Figure 5.8 show temperature is plotted vs x–direction at y–direction values of 0, 0.5 and 1 m, respectively. The pattern of the plots is similar at these different y locations, with the most noticeable effect at the y = 0 location and a diminishing effect away from this boundary. As before, each plot uses the same temperature scale to help show this decreasing effect at the interior y–values.

At time values of 1 and 5 seconds, sin (ωt) = 1; therefore the heat flux will be exiting the boundary at its maximum magnitude (8350). This is visible in each plot by the equal negative slope for both of these time curves. The opposite is true at t = 3 seconds where sin (ωt) = –1 giving a heat flux of maximum magnitude entering the boundary. The boundary is adiabatic at the times of 2 and 4 seconds, where sin (ωt) = 0. Both the 2– and 4– second temperature curves are very nearly perpendicular to the y–axis at x = 1. The results of the analytical and numerical solutions agree very well, and they also behave as expected with respect to the specified boundary conditions.

A time step of 0.002 seconds was used by the conduction solver for this case. A quick test case was run to see the effect of a much larger time step: this step value was increased to a value of 0.02 seconds, resulting in a Fourier number of approximately 11.5. The code behaved well with respect to stability, but the time accuracy of the results is noticeably (and expectedly) worse than before. Figure 5.9 illustrates this point for the y = 0 boundary.

The good results from both the 1–D and 2–D test cases suggest that the heat–conduction solver works well. The implicit nature of the solver provides good numerical stability; this was demonstrated in both cases with Fourier numbers significantly larger than the explicit limit. The conduction solver used only one grid block for each of these test cases. The next step is to address the thermal block–to–block communication if more than one grid block is required for a solid region.

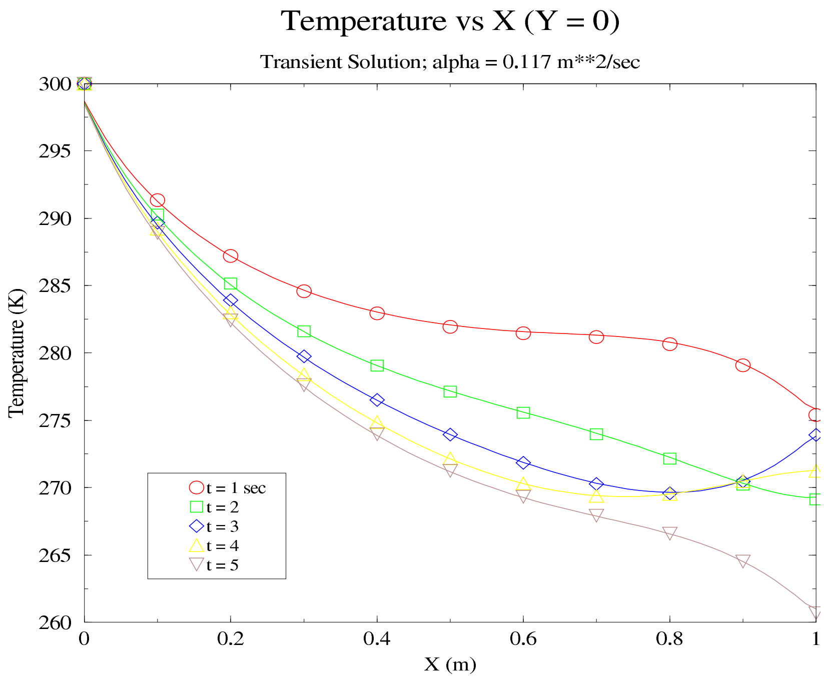


Figure 5.6 Temperature vs. X for 2–D, Transient Test Case; Fourier Transform Method (Symbols) and Heat–Conduction Solver (Curves); Y = 0 m, Time Step = 0.002 Seconds

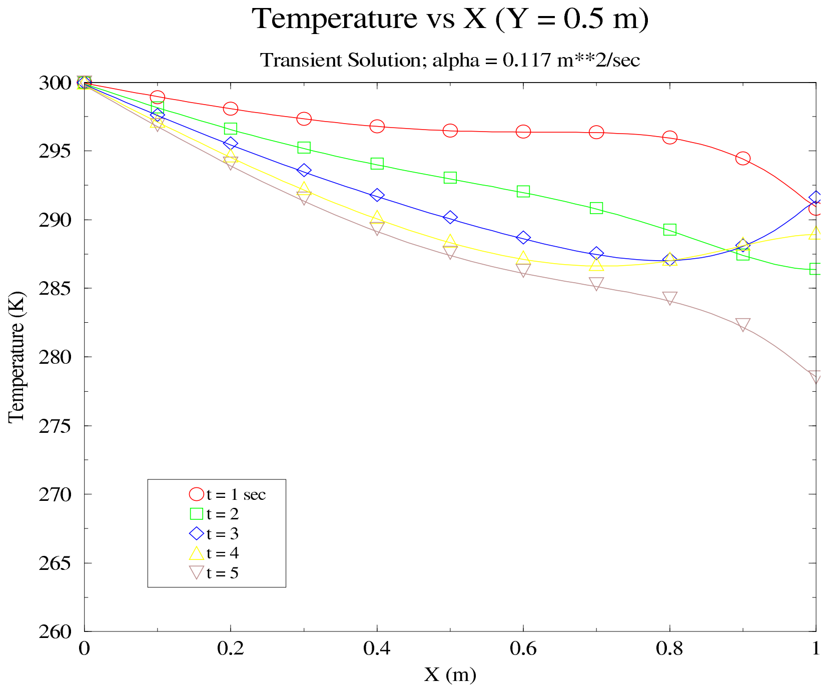


Figure 5.7 Temperature vs. X for 2–D, Transient Test Case; Fourier Transform Method (Symbols) and Heat–Conduction Solver (Curves); Y = 0.5 m, Time Step = 0.002 Seconds

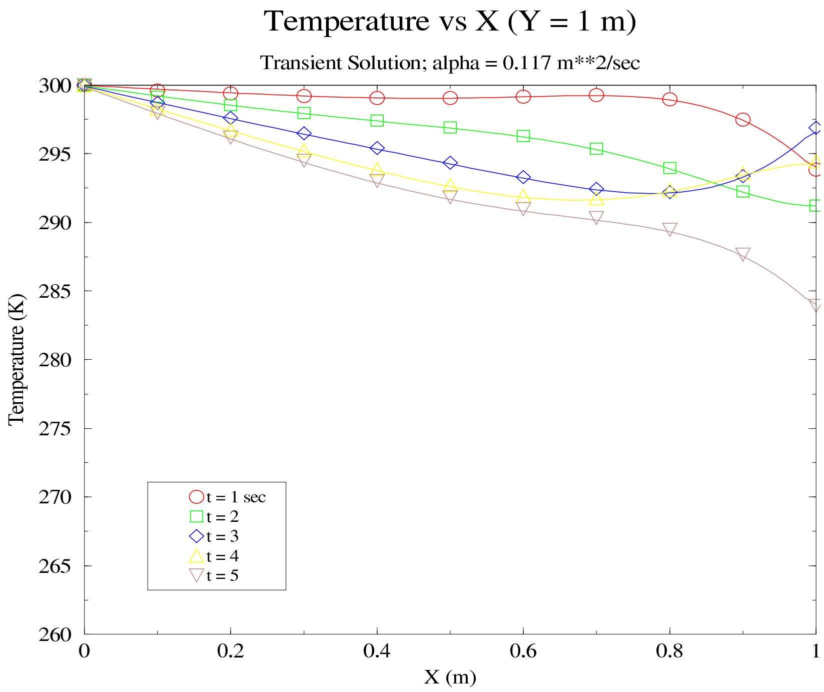


Figure 5.8 Temperature vs. X for 2–D, Transient Test Case; Fourier Transform Method (Symbols) and Heat–Conduction Solver (Curves); Y = 1.0 m, Time Step = 0.002 Seconds

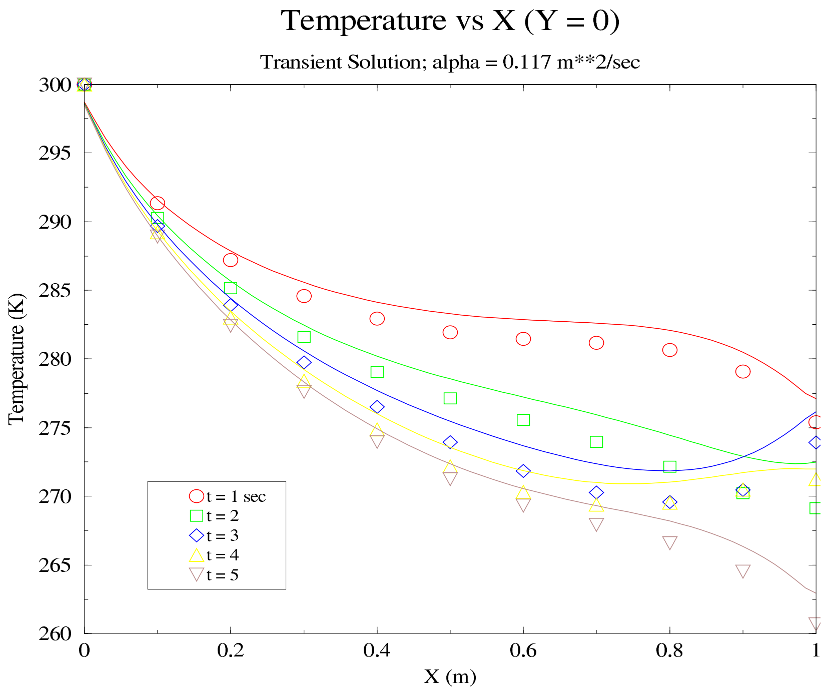


Figure 5.9 Temperature vs. X for 2–D, Transient Test Case; Fourier Transform Method (Symbols) and Heat–Conduction Solver (Curves); Y = 0 m, Time Step = 0.02 seconds

Block–to–Block Communication for Composite Slab

The same general two–dimensional grid from the previous section will be used to investigate the block–to–block communication of both the composite and homogeneous slabs. The same basic boundary conditions from before will also be used. The two key factors at the block–to–block boundaries are the thermal diffusivities and the grid spacing on each side.

As mentioned in the previous chapter, the thermal diffusivity is a measure of how rapidly thermal effects spread through a material. A composite slab is composed of at least two different materials, and, consequently, at least two different diffusivity values. This, of course, means that each different part or component of the slab evolves at a different rate, depending on the diffusivity value. The ratio of the respective diffusivities at a given boundary determines the physical behavior of the problem, moreover it directly affects the numerical stability.

Again, the same simple 1 m x 1 m geometry from before will be used; any changes will be in grid size and spacings at the interface, the location of which is x = 0.5 m. The material properties of silver and commercial bronze will be used for this case. The values for the material properties are: κ_{silver}= 429 W/(m–K), c_{silver} = 235 J/(kg–K), ρ_{silver}= 10500 kg/m^3 and κ_{bronze}= 52 W/(m–K), c_{bronze} = 420 J/(kg–K), ρ_{bronze}= 8800 kg/m^3. These values give diffusivities of approximately α_{silver} = 1.74e–04 m^2/sec and α_{bronze} = 1.4e–05 m^2/sec. As before, the boundaries of x = 0 and x = 1 m are kept at a constant temperature of 300K; the y = 0 boundary has a heat flux of –8350 W/m^2 and an adiabatic condition is prescribed y = 1 m.

The basic procedure for block–to–block communication of composite slabs was described in the previous chapter. Lower–numbered blocks take boundary temperatures from higher–numbered blocks and calculate a heat flux at the boundary. This heat flux is then passed to the higher–numbered block for use as its boundary condition. The ratio of diffusivities at an interface is $R_\alpha = \alpha_{lower} / \alpha_{higher}$.

As a first example, suppose that $R_\alpha > 1$; the diffusivity of the lower–numbered block is then the greater of the two and that block will experience a greater change than the higher–numbered block, assuming the same grid size on both sides of the interface for some given time step. In this methodology the lower–numbered block is the "initiator" of action at the interface; the higher–numbered block takes the supplied boundary condition and then provides a "response" (in the form of a interface temperature) to the lower block. For the case of $R_\alpha > 1$, the lower–numbered block provides a boundary condition that the higher block can not adequately respond to. The temperatures along the interface will diverge quickly to non–physical values. The greater the value of R_α, the worse is this problem.

Another way to look at this situation is in terms of the grid–cell Fourier number at the interface, and this is where interface grid spacing (and thus cell size) comes into play. Initially, the grid spacings were specified as being the same at the interface (though non–uniform for the grid block itself). By increasing the spacing in the lower–numbered block (of higher diffusivity), the grid–cell Fourier number is reduced. The lower the block 1 grid–cell Fourier number relative to that of block 2, the more stable the interface communication was found to be.

Assume block 1 to consist of silver and block 2 of bronze; the value of R_α is then approximately 12.4. A grid spacing of 0.001 m is given for either side of the interface with equal grid sizes of 41 x 71 x 2; an initial time–step value of 0.2 seconds is specified. The calculation aborts in less than 10 iterations after a negative (absolute) temperature is calculated. Reducing the time step by a factor of 10 still fails to get the solver past 10 iterations. In fact, continuing to make order–of–magnitude reductions in the time step gains little. However, by increasing the size of the cell on the silver side of the interface, some success is found. By increasing the interface spacing of the silver block to 0.024 m (the maximum i–dimension value is also reduced from 41 to 36), a time step of 0.2 could be used. Of course, this increasing of cell size should be held somewhat in check so as to maintain as much possible accuracy of the temperature derivative at the interface.

Plots of temperature vs x–direction, for various y–direction values and a given time, are shown in Figure 5.10 through Figure 5.12. Pictures of temperature contours for two different times are given in Figure 5.13 and Figure 5.14. All of these figures give a good view as to how each side of the interface progresses and the abrupt change in slope of the temperature plots at the interface.

As a short side note, there is nothing special about the value of 0.2 for time step. The purpose here is to gain some idea of what is necessary to get a relatively high value of time step and still maintain numerical stability with these explicit block–to–block boundaries. As will be seen later, the time step values required for fluid–solid thermal coupling are often governed by the fluid aspect of the problem; this is certainly true in high–speed flows. If time step values on the order of 0.01 seconds or greater can be obtained for solid block–to–block grids, then numerical stability at these boundaries should be of little concern in fluid–solid coupling problems. The required time steps for the fluid aspect of the problem are often at least an order of magnitude smaller than that for the solid.

Assume now that the blocking arrangement from the previous page is reversed, with block 1 being bronze and block 2 being silver. The value for R_α is now less than one. Assume, too, that the grid spacings on either side of the interface are again equal (0.001 m with grid sizes of 41 x 71 x 2). Block 1 will pass a heat–flux boundary condition to block 2 as before. In this case, however, block 1 has the smaller diffusivity and will naturally progress at a slower thermal rate compared to block 2. That is, the grid–cell Fourier numbers of block 1 are already lower than those of block 2 without any changes in cell size needed.

Simulation runs with time steps of 1 second were successful using 1st–order time accuracy. Time–step runs of 2 seconds were successful by increasing time accuracy to 2nd–order.

Another favorable aspect of the bronze–silver blocking arrangement has to do with adequate resolution of the interface derivatives. Since the conductivity of bronze is much less than that of silver (i.e., $R_\kappa = \kappa_{lower}/\kappa_{higher} < 1$), the temperature derivative on the bronze–side of the interface will be much greater than that of the silver side. Therefore, if sufficient grid spacing is specified for resolution of the derivative on the bronze side, equal spacing on the silver side will be more than enough for resolution of its derivative.

Another small test was run in which the spacing on the silver side of the interface was set to double that of the bronze side (i.e., 0.001 m for bronze, 0.002 m for silver). This test was just to see what, if any, effect would result, since the equal spacing of 0.001 m on the silver side was not required from the derivative resolution standpoint. The stability deteriorated, and a smaller time step than before was required. Doubling the grid size reduced the Fourier number of the silver–side grid cells, which is a less stable situation.

Since the action of the previous paragraph destabilized the interface, another small test was tried in which the silver–side grid spacing at the interface was the reduced from 0.001 m to 0.0005 m. Now the Fourier number of the grid cells of the silver block (at the interface) was increased by roughly a factor of 4 over that of the 0.001–spacing, thus driving the respective Fourier numbers farther apart to the more stable condition. These results were quite interesting; time step values as high as 20 seconds were executed using 1st–order time accuracy with no problems. Figure 5.15 through Figure 5.17 compare the results of bronze–silver blocking arrangement carried at a time of 1,000 seconds and three different values of the y–direction. The first run used a time step of 1 second (1,000 iterations) with equal grid spacing of 0.001 m on both sides of the interface. The second and third runs are at time steps of 10 (100 iterations) and 20 (50 iterations) seconds, respectively. The results do not seem to differ greatly, considering the difference in time step size.

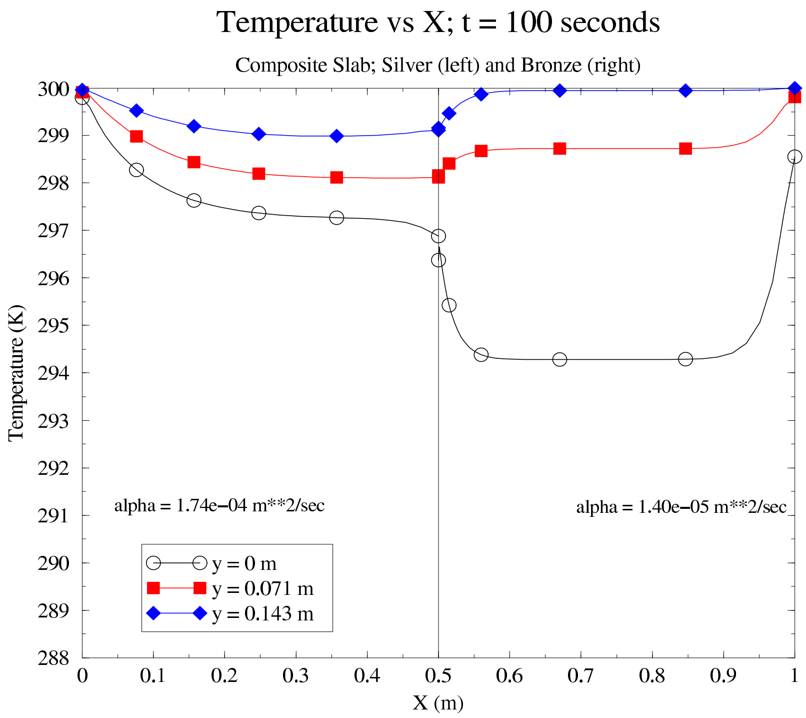


Figure 5.10 Temperature vs. X for 2–D, Silver–Bronze Composite Slab at Time = 100 Seconds

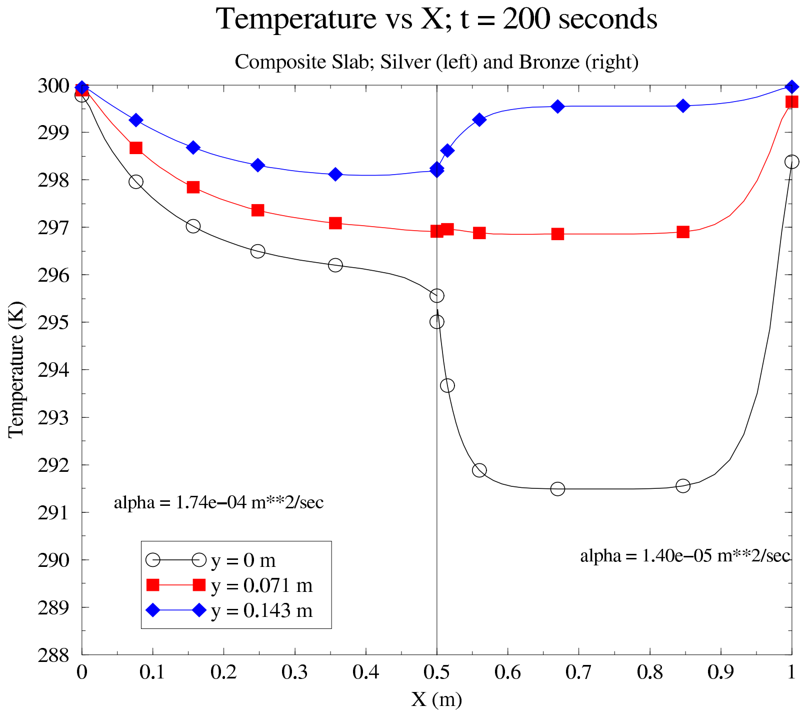


Figure 5.11 Temperature vs. X for 2–D Silver–Bronze Composite Slab at Time = 200 Seconds

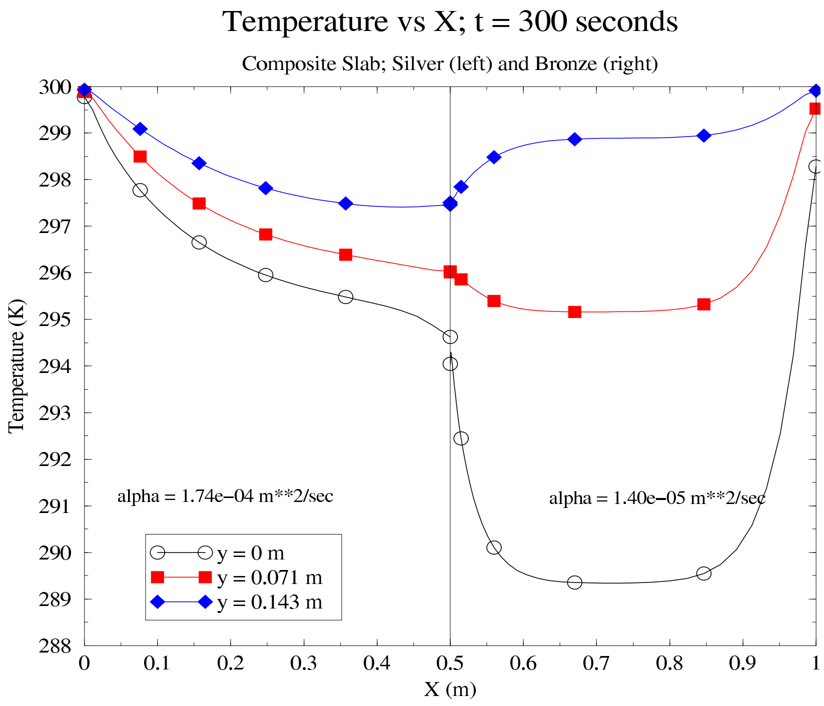


Figure 5.12 Temperature vs. X for 2–D Silver–Bronze Composite Slab at Time = 300 Seconds

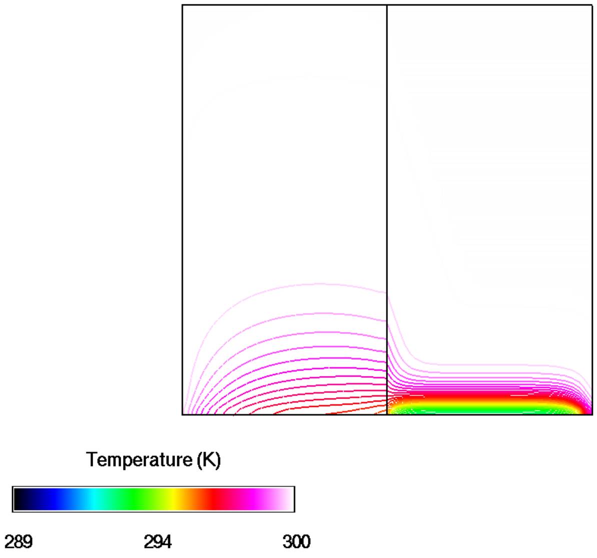


Figure 5.13 Temperature Contours for 2–D Silver–Bronze Composite Slab at Time = 150 Seconds

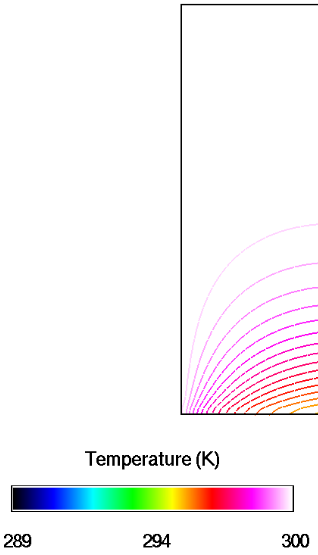


Figure 5.14 Temperature Contours for 2–D Silver–Bronze Composite Slab at Time = 300 Seconds

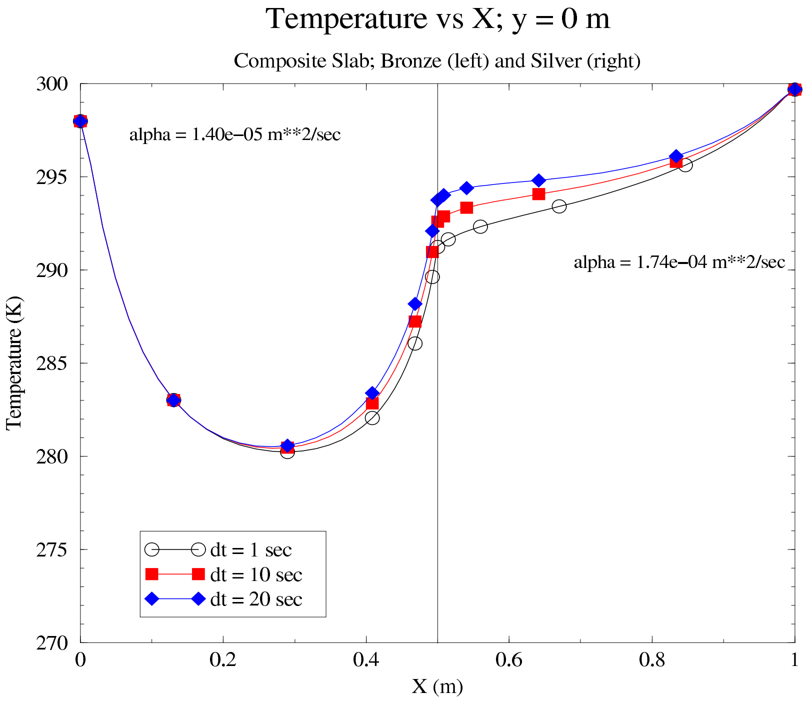


Figure 5.15 Temperature vs. X for 2–D Bronze–Silver Composite Slab at Y = 0 m, Time = 1000 Seconds

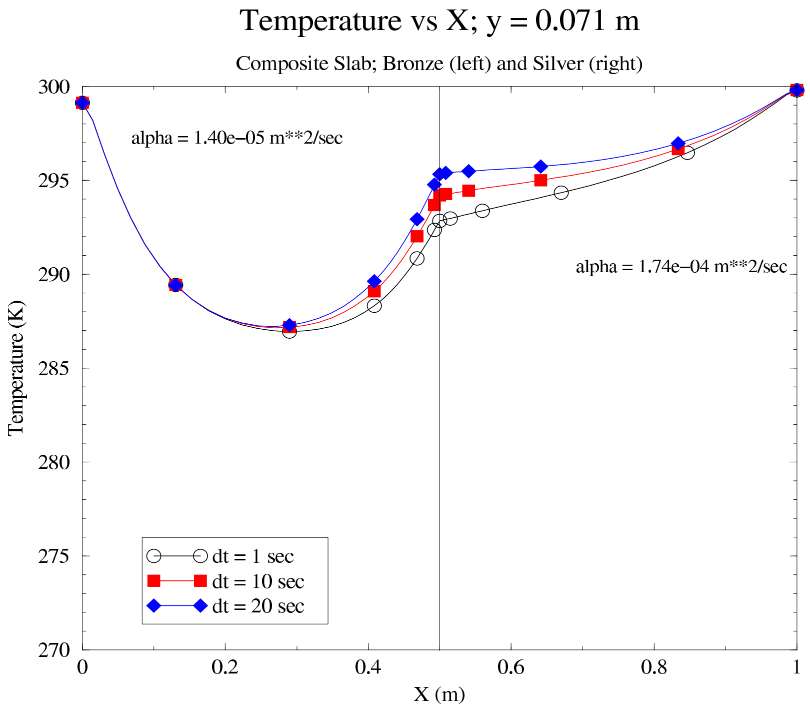


Figure 5.16 Temperature vs. X for 2–D Bronze–Silver Composite Slab at Y = 0.071 m, Time = 1000 Seconds

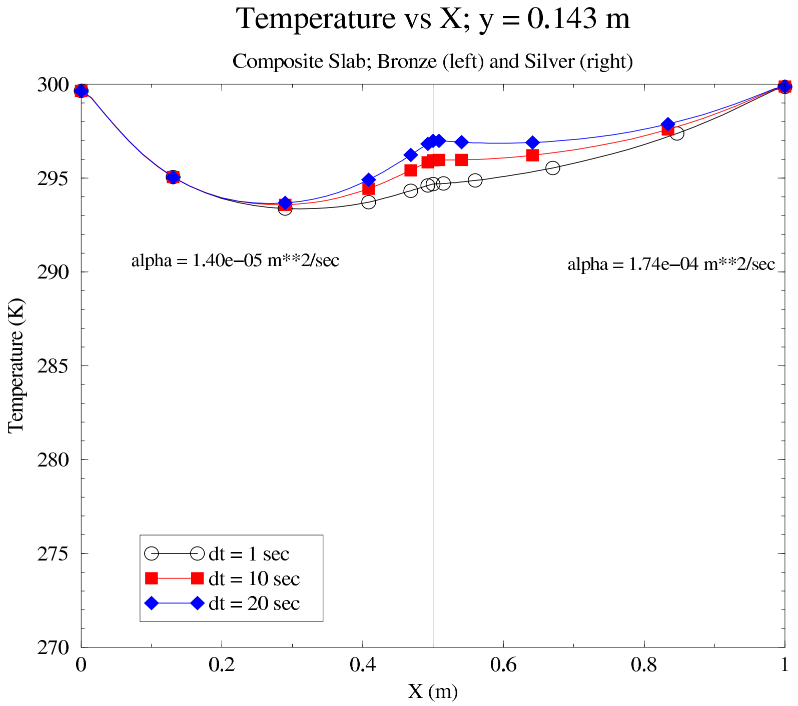


Figure 5.17 Temperature vs. X for 2–D Bronze–Silver Composite Slab at Y = 0.143 m, Time = 1000 Seconds

The previous example using bronze and silver as the two materials gives a relatively large difference between both the diffusivities and conductivities. Another example using materials with properties closer in value will be examined to gain some idea of this effect. The materials of aluminum and copper are chosen for this case. The nominal property values are: $\kappa_{copper} = 401$ W/(m–K), $c_{copper} = 385$ J/(kg–K), $\rho_{copper} = 8933$ kg/m^3 and $\kappa_{aluminum} = 237$ W/(m–K), $c_{aluminum} = 903$ J/(kg–K), $\rho_{aluminum} = 2702$ kg/m^3. The resulting diffusivity values are approximately α_{copper} = 1.17e–04 m^2/sec and $\alpha_{aluminum}$ = 9.7e–05 m^2/sec. The ratios of both diffusivity and conductivity (copper/aluminum) are between 1 and 2. With diffusivities so close in value, the respective Fourier numbers at the interface will be closer in value than previously. The same grid size as before is used, as well as an equal interface grid spacing of 0.001 m. Also, the initial and boundary conditions from before are used. The aluminum block is designated block 1, the copper one block 2.

The bronze–silver example demonstrated that the greater the difference in Fourier numbers at the interface, the more stable the problem. The diffusivities, and thus naturally the Fourier numbers, for this case of copper and aluminum are much closer than before. This fact suggests that the maximum achievable time steps will be smaller than previously, and this is indeed the case. A time–step value of approximately 0.05 seconds turns out to be the maximum attainable for this grid and its accompanying initial and boundary conditions with 1st–order time accuracy (lower–numbered block is aluminum, higher–numbered one is copper). Increasing the time accuracy to 2nd–order allows the time step to be increased to roughly that of 0.075 seconds. Figure 5.18 through Figure 5.20 shows that the thermal effects are progressing, as expected, at almost the same pace throughout the solid. Also, the change in the temperature plot at the interface is much less abrupt due to the relatively close conductivity values.

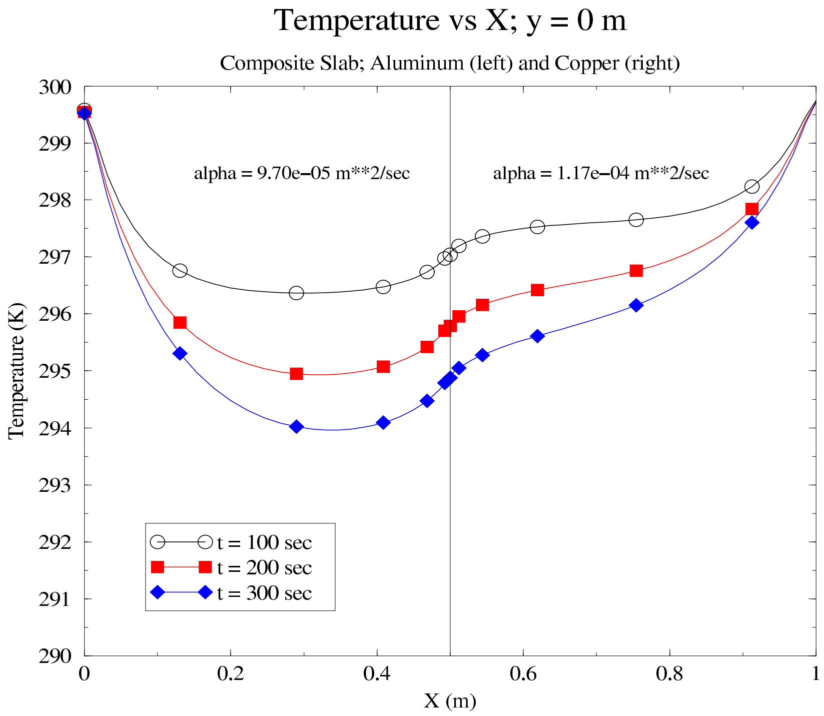


Figure 5.18 Temperature vs. X for 2–D Aluminum–Copper Composite Slab at Y = 0 m

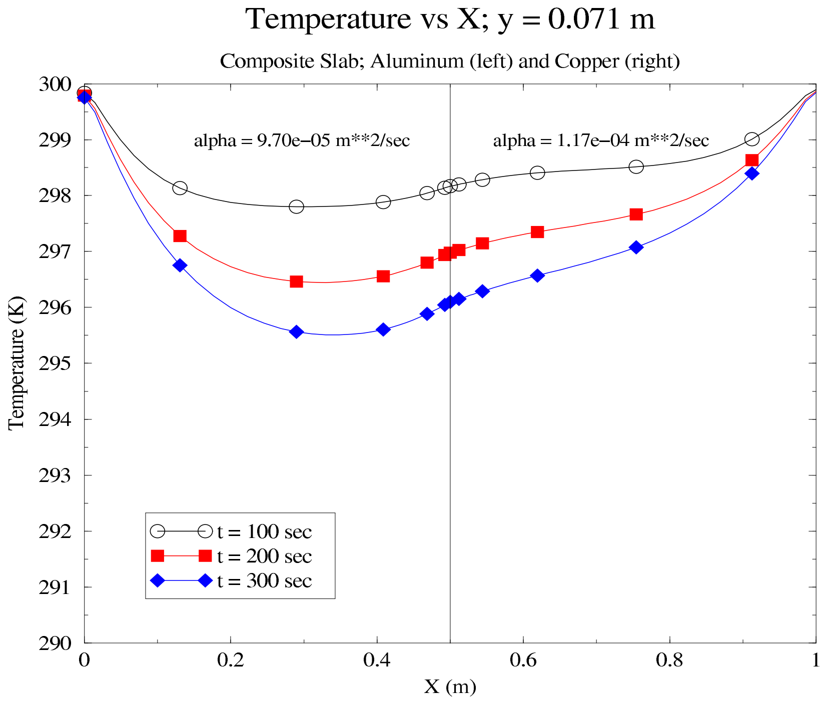


Figure 5.19 Temperature vs. X for 2–D Aluminum–Copper Composite Slab at Y = 0.071 m

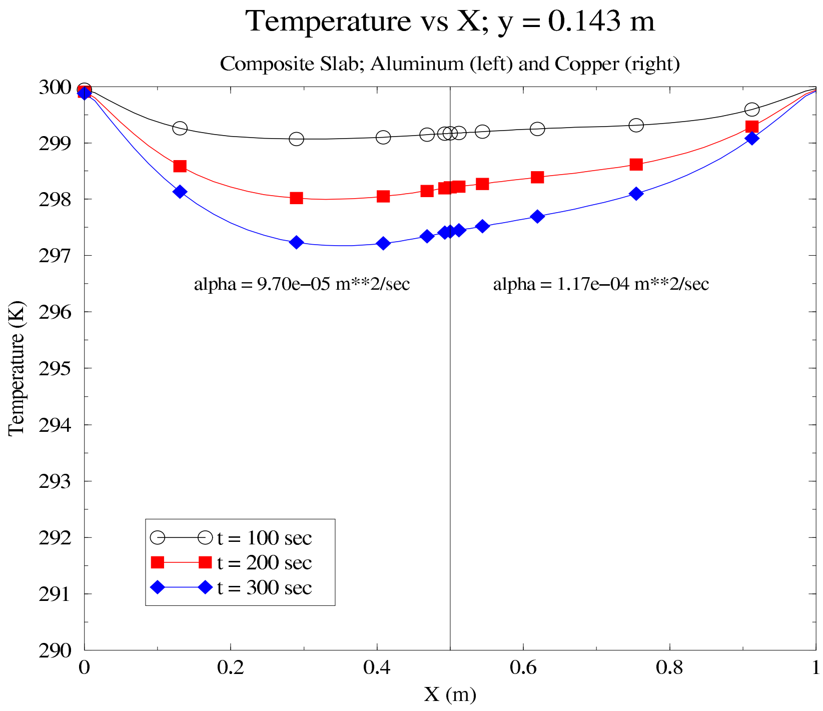


Figure 5.20 Temperature vs. X for 2–D Aluminum–Copper Composite Slab at Y = 0.143 m

In order to increase the difference in Fourier numbers, the grid spacing on the copper–side of the interface was changed from 0.001 m to 0.0005 m as in the previous example. This change provided practically no increase in the maximum time step. The copper–side spacing was further reduced to 0.00025 m. This allowed the time step to be increased to 0.075 seconds using 1st–order time accuracy and 0.125 seconds for 2nd–order. This is an improvement, but still it shows that these two materials behave so alike one another that increasing the time step by changing cell size is of limited practical use.

The above example of the aluminum–copper composite slab illustrates the increased difficulty in making simulations without having to resort to extremely small time steps. The closer the diffusivity ratio, R_α, gets to 1, the more difficult is the task of finding a useable time step. The "worst–case scenario" for this block–to–block communication methodology is then that of a composite slab consisting of materials having equal diffusivities (which is highly unlikely), or that of the homogeneous slab, which will be addressed next.

Block–to–Block Communication for Homogeneous Slab

Based on the previous discussion, one could infer that the homogeneous slab, consisting of more than one block, would be very difficult to treat. Indeed, small time steps and very tight spacing at the block–to–block boundary might be required. If the time step is specified to be very small, the code may be used in an explicit manner; however, this may not be practical. Tight grid spacing is not necessarily required due to the fact that the temperature derivative will be continuous (because of equal conductivities) at a block–to–block boundary within a homogeneous slab. Tight grid spacing is not desired, if not needed.

A slight modification of the composite–slab methodology was tried and met with some success. A boundary heat flux was still calculated by the lower–numbered block and passed to the higher–numbered block for use as its boundary condition. And this heat–flux calculation was still based on a temperature (from the previous time–step iteration) provided by the higher–numbered block to the lower–numbered one. However, in this case the higher–numbered block passed the cell–center temperature from its interface grid cells rather than a temperature on the interface itself (briefly described in Chapter 4). The approximation of the temperature derivative used for calculating the heat flux boundary condition becomes a second–order central difference, using cell–center temperatures on each side of the interface. Grid–cell Fourier numbers for this method are still rather limited for stability reasons, however. This methodology may be used without necessarily having to alter grid spacing at the the block–to–block interface (especially if the solid is being coupled to a fluid flow requiring small time steps).

The method described above was used when first testing the coupling of the heat–conduction solver to the flow solver, and this way of treating the interface worked fine for the time steps involved (these values were small due to the fluid part of the problem). There was still a desire to reduce the concern of time–step limitations due to possible solid block–to–block interfaces, since it is felt that a significant number of fluid–solid problems will involve a homogeneous solid. A basic description of a different method of block coupling within the Gauss–Seidel iteration loop was given in the previous chapter. The strength of this method is that it should give the same results as if there were only one solid grid block; this allows for larger time–step values.

As a quick reminder, this “Gauss–Seidel coupling” primarily consists of reversing the orders of the grid–block loop and the Gauss–Seidel loop (i.e., the block loop is now contained within the Gauss–Seidel loop rather than vice–versa). Also, the part of the solution vector (that is, the ΔT^n values for each grid cell) along the interface is passed from block to block instead of heat–flux or temperature boundary conditions.

Testing of this procedure was accomplished by using the same basic two–dimensional slab grid (along with the same initial and boundary conditions) as that for the 2–D validation of the conduction solver, since both analytical and single–block numerical results are already available for comparison. The 1–block grid size from the validation case is 71 x 71 x 2; this will be divided at the x = 0.5 point into two blocks of 36 x 71 x 2 (evenly spaced). Only the unsteady problem from above will be discussed (time step = 0.002 seconds with a Fo = 1.15).

The comparison is made to the numerical results from the validation case. If the code works correctly, there should be no difference between the 1–block and 2–block results. The 1–block case compared very well with the analytical solution, so comparison of the numerical results of the 1–block and 2–block cases should be sufficient. Figure 5.21 through Figure 5.23 show both the 1–block and 2–block results, with the 1–block numerical results represented by the symbols and the 2–block results by the curves: the solutions are the same. The values plotted in these figures are cell–centered values and have been averaged to get values for the grid points. This is done for post–processing software and the present averaging at boundaries causes the slight “kink” in the curve at the interface. The actual cell–centered temperatures written to the re–start files (after 2500 time–step iterations) have practically no difference between the 1–block and 2–block cases.

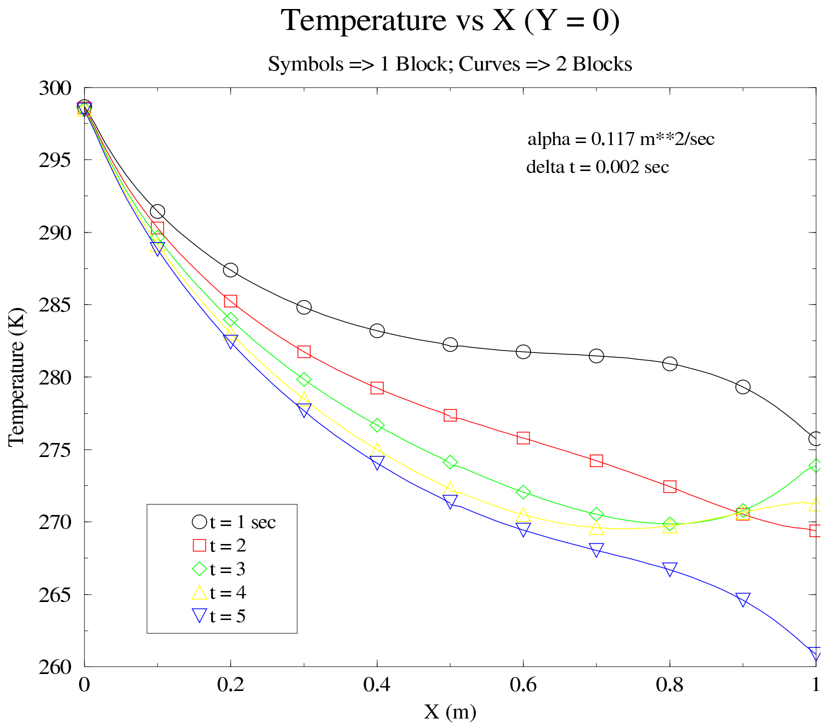


Figure 5.21 Comparison of 1–Block (Symbols) and 2–Block Implicit Coupling (Curves) Solutions for 2–D Transient Test; Y = 0 m, Time Step = 0.002 Seconds

Another quick test was conducted to help further demonstrate the stability. Just as before in the validation case, the time step was increased by an order of magnitude to 0.02 seconds (Fo = 11.5); the results are given in Figure 5.24. In this figure, the two–block temperature curves are compared to the one–block solutions (keeping in mind that the one–block results used a time step of 0.002 seconds). The code proved to be stable with results very much like those in Figure 5.9, which demonstrate the effect of time step on accuracy. So, this method of “Gauss–Seidel coupling” seems to allow the homogeneous solid to be treated relatively easily, at least for this simple 2–D geometry, and there appears to be no restriction on time step. The code has been constructed so that the setting of an input parameter will determine whether to execute the Gauss–Seidel loop either inside or outside of the grid–blocking loop.

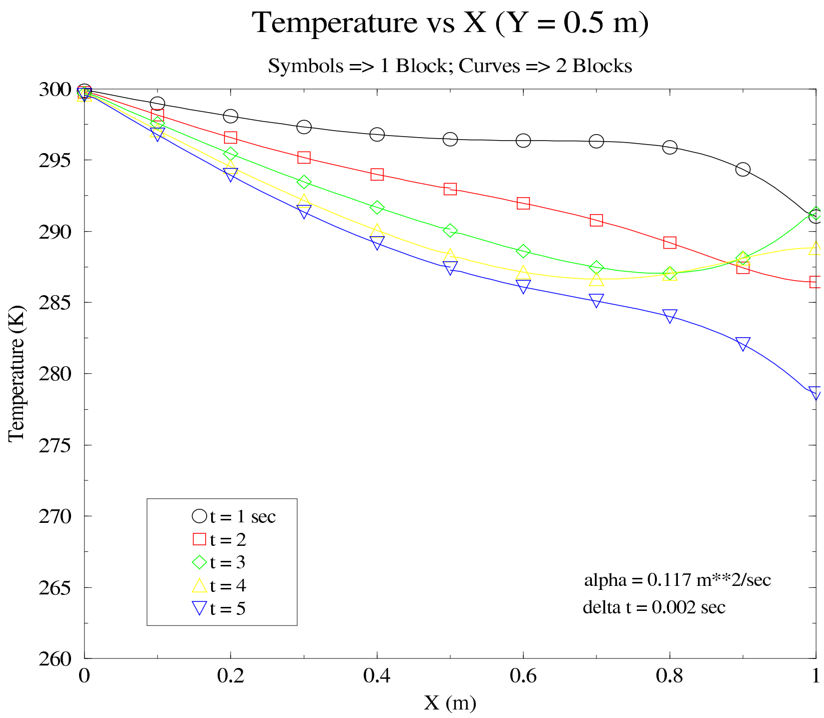


Figure 5.22 Comparison of 1–Block (Symbols) and 2–Block Implicit Coupling (Curves) Solutions for 2–D Transient Test; Y = 0.5 m, Time Step = 0.002 Seconds

This will complete the results and discussion pertaining solely to the heat–conduction solver. In summary, it has been found that the block–to–block thermal communication among solid blocks can be a potential source of numerical problems, in spite of the fact that the heat conduction problem on a single block is not computationally challenging. The next group of results will concern the actual coupling of the heat–conduction solver to the flow solver.

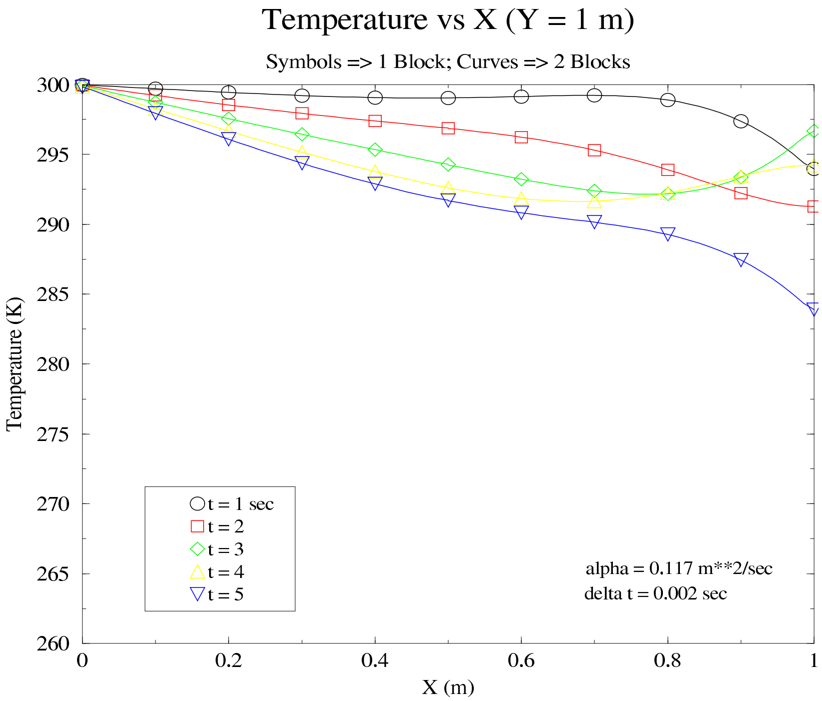


Figure 5.23 Comparison of 1–Block (Symbols) and 2–Block Implicit Coupling (Curves) Solutions for 2–D Transient Test; Y = 1.0 m, Time Step = 0.002 Seconds

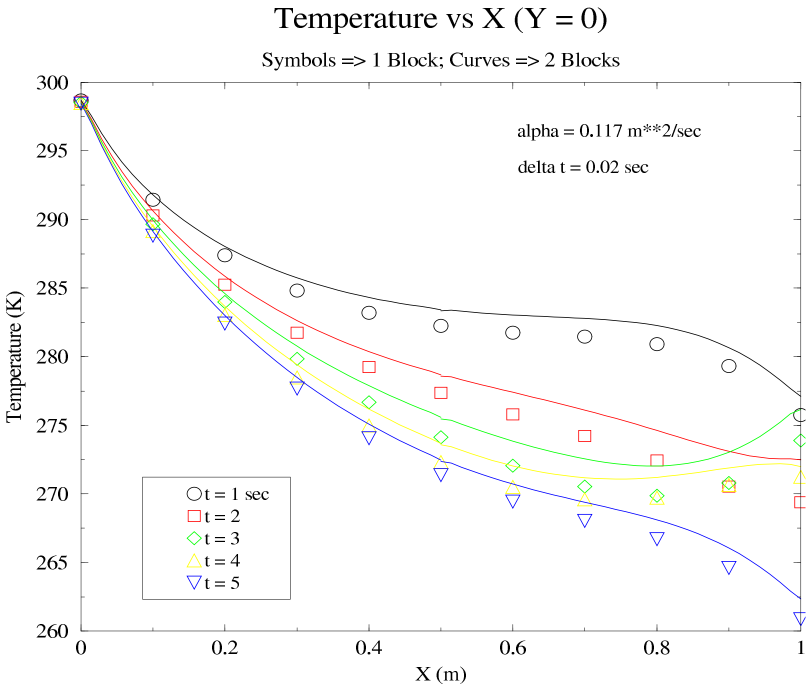


Figure 5.24 Comparison of 1–Block (Symbols) and 2–Block Implicit Coupling (Curves) Solutions for 2–D Transient Test; Y = 1.0 m, Time Step = 0.02 Seconds

Coupling of Heat–Conduction Solver and Flow Solver

Steady Laminar–Flow Over a Flat Plate

The methodology for linking the two solvers is basically the same as that for the composite slab. After all, a fluid region and a solid region differ only in their material makeup and the fact that a fluid block has convective flux in addition to the diffusive flux crossing grid–cell boundaries. At the fluid–solid interface, diffusive energy flux is the concern just as for the composite slab.

As previously discussed, the fluid blocks calculate heat flux (regardless of block number) and pass this to the solid blocks; the solid blocks return an interface temperature to the fluid blocks. Within any given time step iteration (or Newton sub–iteration), the fluid blocks are all processed first, and the heat fluxes sent to the appropriate solid boundaries. The solid blocks are then all solved. This is an explicit boundary condition for a fluid–solid interface, but it allows for consistent communication in case of multiple fluid–solid boundaries.

The first test of this fluid–solid coupling is a flat plate in laminar flow. The plate will be of finite thickness, but thin enough that the thermal mass is small; therefore, the fluid will be exposed, for all practical purposes, to a constant–temperature wall condition. The convection heat transfer solution to this problem is given entirely in terms of dimensionless parameters; these are the Nusselt Number, $\mathrm{Nu_x}$, the Reynolds Number, $\mathrm{Re_x}$, and the Prandtl Number, Pr [41]. The expression relating these parameters is

$$Nu_x = 0.332\sqrt{\mathrm{Re}_x}\,\mathrm{Pr}^{1/3} \ ; Nu_x = \frac{hx}{k}. \tag{5.12}$$

The convection coefficient, h, may be expressed as a function of x:

$$h(x) = \frac{Nu_x k}{x} \Rightarrow h(x) \to \infty; \ x \to 0. \tag{5.13}$$

The grid for this case consists of two fluid blocks and one solid block. The first fluid block extends upstream of the plate's leading edge for a short distance; the second fluid block is in contact with the solid block. Both fluid blocks are oriented with increasing k–indices in the streamwise direction and increasing i–directions normal to and away from the fluid–solid interface. The solid block has increasing i–indices in streamwise direction and increasing j–indices normal to and toward the fluid–solid interface. The purpose of the different index orientation is to help test the generality of the blocking arrangement.

The grid dimensions are given in the table below. The physical dimensions of the plate are a length of 0.1778 m (7 inches), and a thickness of 2e–04 m (0.008 inches). Initial spacing away from the boundary on the fluid side is 7e–06 m and 1.27e–05 m on the solid side. Initial spacing in the streamwise direction is 1.27e–05 m at the leading edge and 2.5e–05 m at the trailing edge.

Table 5.5 Block Numbers and Dimensions for Flat–Plate Grid

Block Number	Grid Dimension
1 (fluid)	51 x 2 x 26
2 (fluid)	51 x 2 x 181
3 (solid)	181 x 7 x 2

The solid block is initialized to a constant temperature of 170 K with a constant temperature of 200 K applied to the j = 1 boundary (opposite the fluid–solid boundary where j = 7). Except for the fluid–solid boundary, all others of the solid are treated as adiabatic walls. The plate is made thin enough that the fluid–solid boundary should reach an effectively constant temperature of 200 K. The fluid blocks are initialized to a free–stream Mach Number of 0.15 and free–stream temperature of 161 K; this temperature, and also the plate length, come from an unrelated test case. The material properties of silver are used for the plate, and a constant time step of 0.002 seconds applied.

The simulation was run for 5,000 iterations (10 seconds). It was felt that this time length would be sufficient to reach a steady state, given the thickness of the plate. As it turns out, this was more than enough; interface temperatures near the leading edge (where the greatest temperature difference was seen) had settled out after the first 1,000 iterations.

Figure 5.25 is a plot of the Nusselt Number vs streamwise direction, x. The circular symbols are for the theoretical solution given by Equation (5.12). The solid curve is from the coupled flow/heat–conduction solver and agrees well with the theoretical prediction. The actual heat flux values that are passed from the fluid block to the solid block and the x values of points along the boundary are output to a file. This file is then input to a small code that evaluates the convection coefficient along the boundary by means of Newton's law of cooling. Reynolds Number and Prandtl Number are calculated using fluid properties evaluated at the so–called "film temperature" (the simple average of the wall and free–stream temperatures, 180.5 K in this case; Prandtl Number, too, is at this temperature). Nusselt Number is then calculated using the convection coefficient, the x–position, and the film–temperature conductivity.

A plot of the numerical and theoretical convection coefficients vs the x–direction is given in Figure 5.26. The most disagreement here is in the region between the leading edge and x = 0.05 m, where the numerical values are slightly less than the theoretical (i.e., the curve appears to be at the "edge" of the circles representing theoretical values). This stands to reason in that the temperature values here are slightly less than 200 K due the conduction within the solid. Figure 5.27 is a close–up view of the temperature contours near the leading edge of the plate. The maximum temperature difference in the plate is just over 0.1 K. The "cold spot", as much as it can be called that here, is at the leading edge.

A second flat–plate test case was conducted with the same basic conditions as the one just described. In this case, though, the thickness of the plate was increased to 0.05 m. Moreover, the plate conductivity was reduced by a factor of 100, in order to see a noticeable temperature difference in the direction normal to the fluid–solid interface. The plate density and specific heat were both reduced by a factor of 10 so that the plate diffusivity will remain the same as before. The combination of thicker plate and smaller conductivity (and thus larger temperature derivative) means that the number of points in the j–direction needs to be increased (this was set at 51, so the solid grid block is now 181 x 51 x 2). Since the plate is much thicker and the conductivity smaller, a higher temperature condition must be specified on the boundary opposite the fluid–solid interface (j = 1), so that the temperature along this interface can be approximately 200 K; this new temperature condition is set to 203.5 K. The same time step of 0.002 seconds is used and the code is executed for 20,000 iterations. The temperatures along the fluid–solid interface by this point have changed less than 0.05 K since the iteration count of 18,000 (i.e., the last 4 seconds).

Figure 5.28 and Figure 5.29 are plots of the Nusselt Number and convection coefficient (vs x) for this thicker flat plate. The shapes of the respective curves are very close to those of the thin–plate case, and the agreement is close to the theoretical. And as for the thin plate, the primary difference between the numerical and theoretical results is in the region of x = 0.025 to x = 0.05 m. A greater temperature drop occurs across (i.e., normal the fluid–solid interface) due to the smaller conductivity. Therefore, a smaller heat flux (relative to the theoretical) is experienced in the leading–edge region, and the convection coefficient curve is less than the theoretical here.

Figure 5.30 is a view of the temperature contours within the plate; the general contour pattern is very similar to the thin plate with the "cold spot" at the leading edge. The interface temperature reaches a maximum of approximately 199.6 K and is within 0.1 K of this value over roughly the last 0.018 m (0.75 inches) of the plate. The contour pattern shows that heat flux in the solid (normal to the contours) has components both tangential and normal to the interface.

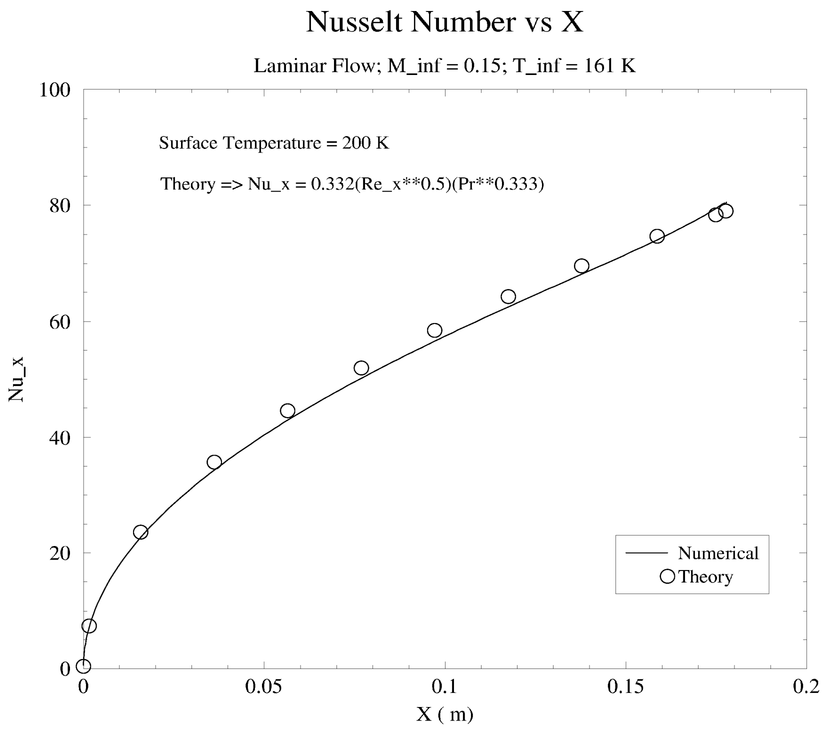


Figure 5.25 Nusselt Number vs X for Thin Flat–Plate Test

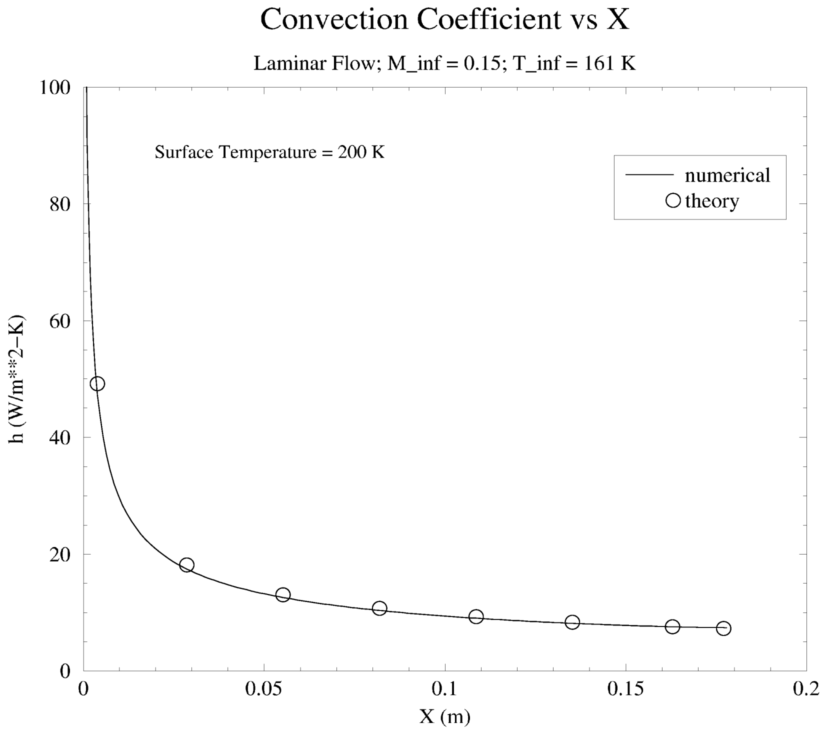


Figure 5.26 Convection Coefficient vs X for Thin Flat–Plate Test

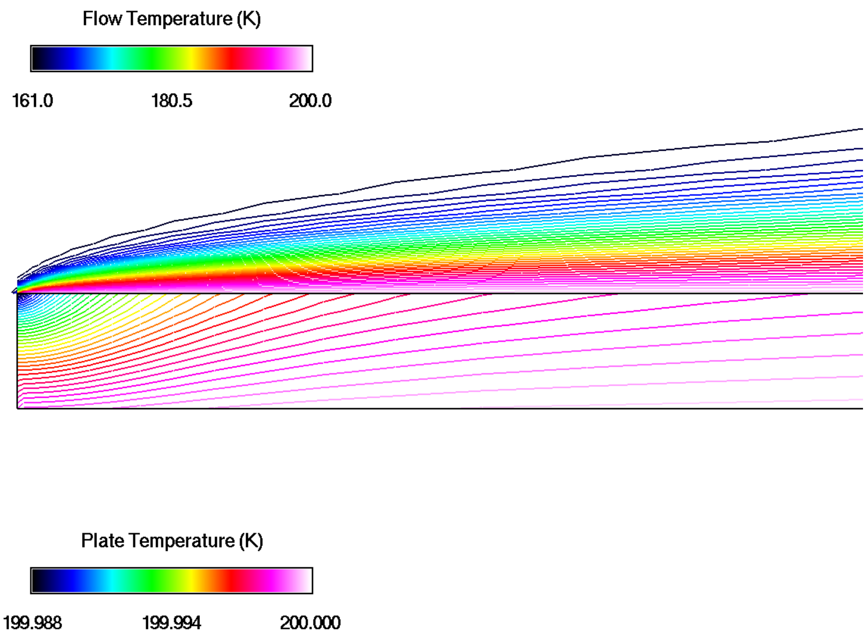


Figure 5.27 Temperature Contours for Thin Flat–Plate Test

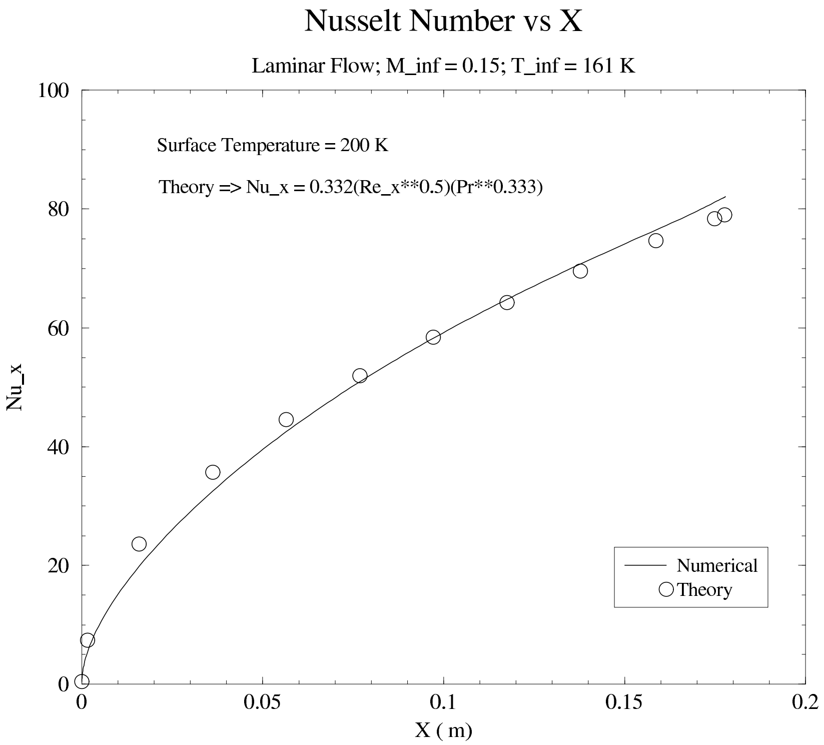


Figure 5.28 Nusselt Number vs X for Thick Flat–Plate Test

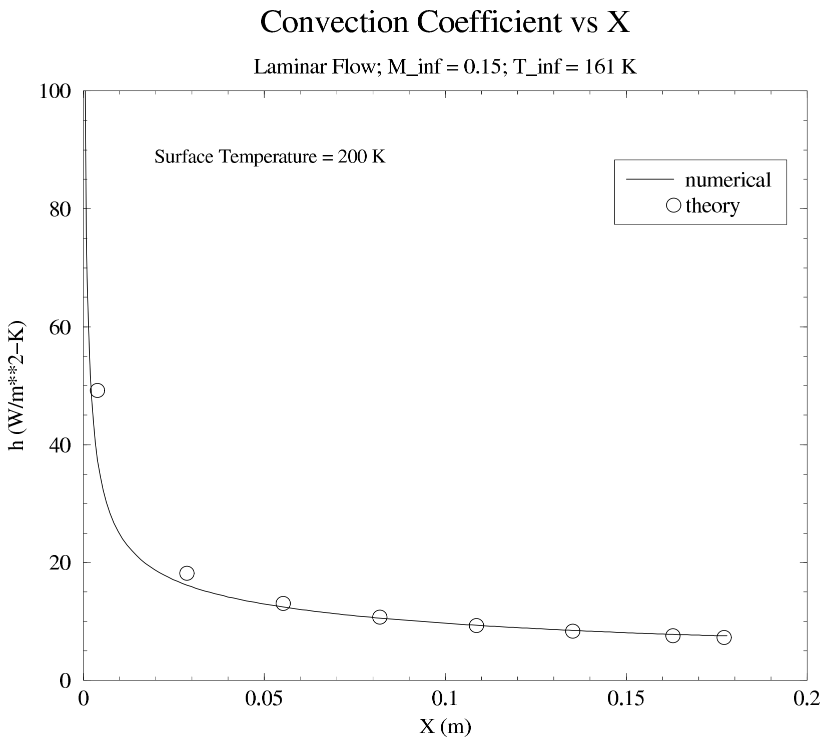


Figure 5.29 Convection Coefficient vs X for Thick Flat–Plate Test

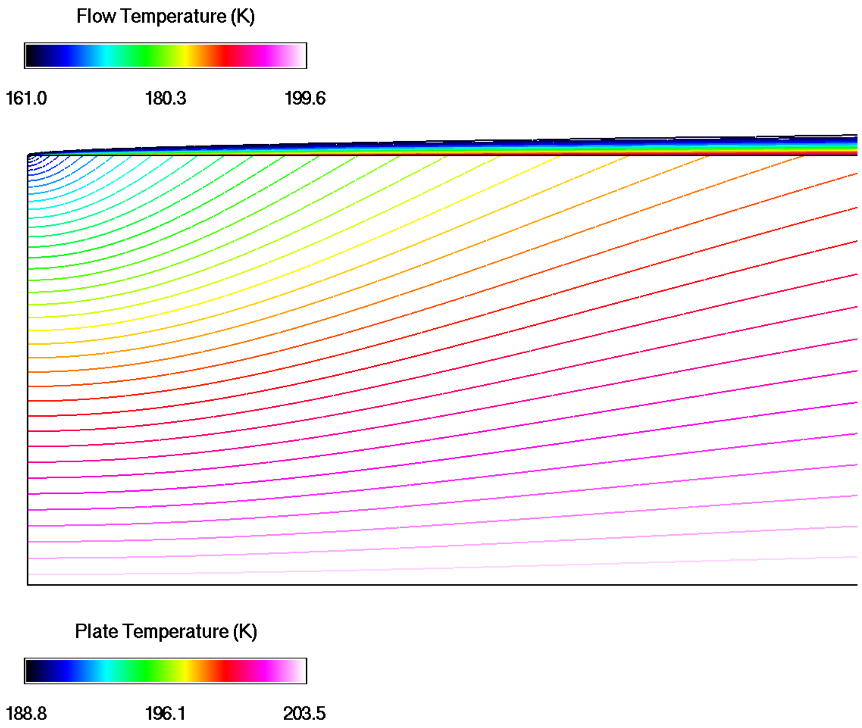


Figure 5.30 Temperature Contours for Thick Flat–Plate Test

The results from these two simple flat–plate test cases are encouraging. Thermal communication across the fluid–solid interface seems to be accounted for well in this fundamental steady–state problem.

Transient Two–Dimensional Blunt–Nosed Wedge

A recent paper by Rahaim, *et al.* [16] provides numerical results of conjugate heat transfer problems along with their own experimental data with which to compare. Temperature measurements within the solid were recorded and plotted vs time. One of the experiments conducted in their research was that of a blunt–nosed wedge in Mach 3 flow. This experiment is the basis of the first transient test in this study.

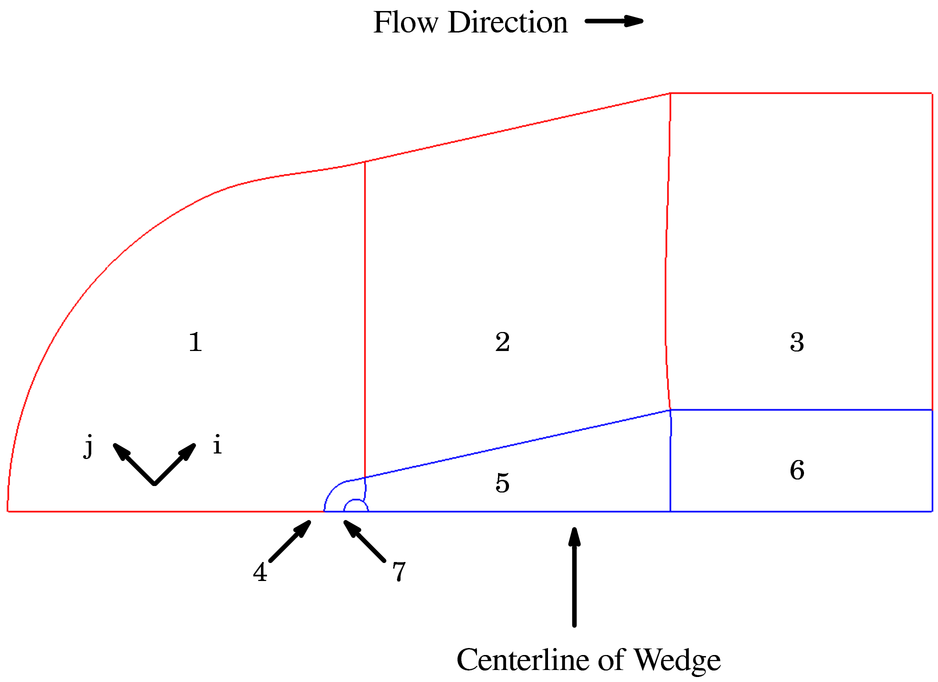


Figure 5.31 Grid Blocking and Numbering for 2–D Blunt–Nosed Wedge

The wedge itself is 3 inches in length, a base height of 1 inch, and a 12.5 degree half–angle. The grid built for this model only included the top half; a symmetry condition was assumed along the grid line extending upstream from the nose of the wedge. Also, the grid is only two–dimensional, which is an approximation, but end conditions were not specified in Rahaim [16] and the complexity of a full three–dimensional simulation was felt to be beyond the level of this test case. The wedge used in the experiment is made of aluminum and the physical properties are the thermal conductivity κ = 167 W/(m–K), the density ρ = 2790 kg/m^3, and the specific heat c = 883 J/(kg–K); the diffusivity is about 68e–06 m^2/sec.

The grid consists of three fluid blocks and four solid blocks. Initial spacing normal to the fluid–solid interface is about 2.54e–06 m for the fluid blocks and twice that for the solid blocks. The figure on the previous page gives the outline and arrangement of the grid blocks, while the dimensions of the blocks are given in the following table.

Table 5.6 Block Numbers and Dimensions for Blunt–Nosed Wedge Grid

Block Number	Grid Dimension
1 (fluid)	71 x 66 x 2
2 (fluid)	71 x 66 x 2
3 (fluid)	26 x 66 x 2
4 (solid)	71 x 31 x 2
5 (solid)	71 x 56 x 2
6 (solid)	26 x 56 x 2
7 (solid)	96 x 11 x 2

The block–to–block communication within the solid was carried out explicitly; i.e., there was no Gauss–Seidel coupling for these blocks. A heat–flux boundary condition was supplied to the higher numbered blocks from the lower–numbered blocks. The higher–numbered blocks then pass the cell–centered temperature (from the cell adjoining the interface), rather than the interface temperature, back to the lower–numbered block for use in the next time step. This procedure gives a limited stability condition, but the time steps mandated by the fluid aspect of the problem are such that the stability is of no concern.

Figure 5.32 is a plot of the temperature at a point within the wedge; the location of the point is approximately 0.1 inches from the nose and along the centerline. The curve with circular symbols represents thermocouple values recorded by Rahaim [16]. Numerical results by Rahaim were obtained using two different flow solver, one of which (NASFLO [43]) was developed on the basis of the work of Whitfield [2]: its predictions are shown by the curve with square symbols. The curve with diamonds gives the results from the present code (QCHEQNS). The agreement between the two sets of numerical results is good for the first second of time in the simulation. After that, the results from QCHEQNS start to tend away from the other numerical results and toward the experimental values. Normally, this would be looked at favorably. However, the QCHEQNS–results are departing from the trend demonstrated by the other code. The results presented in Rahaim [16] are given for a time of 11 seconds; the present plot is limited to two seconds because the QCHEQNS–results continue to trend away from the numerical results of Rahaim [16].

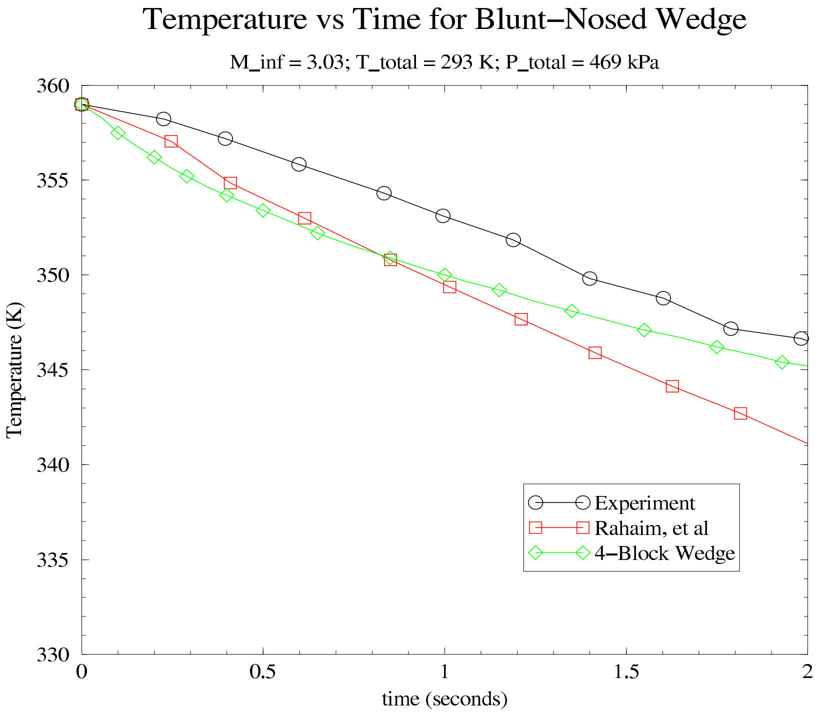


Figure 5.32 Temperature vs. Time at X = 0.1 Inches Interior from Nose of Wedge 4–Block Grid for Heat–Conduction Solver

The most probable cause of this disagreement is the boundary conditions for the wedge. The paper by Rahaim did not specify the conditions used on the base of the wedge or at the two end faces (the constant–k boundaries for this case). This is a two–dimensional grid assumed to be a cross–sectional slice at the center of the wedge; for this first attempt at the problem, these boundaries were set to adiabatic wall conditions. During the early part of the run, however, these boundary conditions would probably have little effect, and this is felt to be the reason for the good agreement within the first few seconds.

Another test case was tried with this configuration to see if a change in the boundary condition on the base of the wedge would have any effect. For this second case, the wedge (solid) grid was modified such that it was only one block. The three fluid blocks were combined into one, also. This change was made to see if the block–to–block boundaries had any effect on the disagreement just discussed.

The wedge base was arbitrarily set to a constant temperature of 300 K and maintained at this value of the first second of the run. After one second, the base temperature was further reduced to 220 K, in an effort to improve the agreement with the NASFLO solution.

Figure 5.33 is the same as the previous one, except now the results from QCHEQNS with a 1–block fluid and 1–block solid grid are added. This plot shows practically no difference between the 4–block and 1–block wedge grids through t = 0.5 seconds. After that, the 1–block curve can be seen to be slightly below that of the 4–block grid curve. This effect can be seen more between t = 1 and t = 2 seconds, though the difference is still small. The agreement between the 1–block and 4–block curves in the early part of the run suggests that the block–to–block boundaries of the 4–block grid are working appropriately and that the base boundary condition has yet to be felt. Where the base effect can be noticed, it is seen to be small. So the question of appropriate wedge boundary conditions still remains, and the best answer probably lies with a full three–dimensional simulation including the end and base flows. These tests have, however, given some insight into the problem.

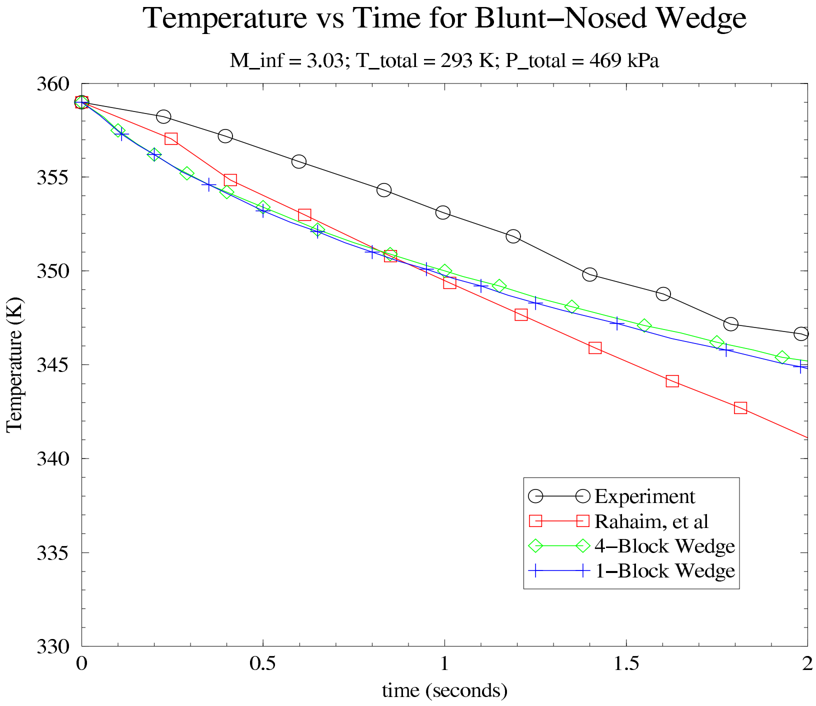


Figure 5.33 Temperature vs. Time at X = 0.1 Inches Interior from Nose of Wedge 4–Block and 1–Block Grids for Heat–Conduction Solver

Figure 5.34 through Figure 5.36 show the temperature contours within the wedge for different times for the 4–block grid. The solid black lines indicate the block boundary locations; the temperature contours have good continuity across these boundaries. The view is concentrated on the nose half of the wedge, since this is where the greatest changes are occurring. The contours show that the heat flow is mostly normal to the nose of the wedge. As the solution evolves, the contours take a shape indicating that the primary heat flow is still toward the nose region and the temperature continues to drop; this stands to reason, as the boundary temperatures downstream of the nose will increase due to the energy collected from upstream and also from viscous dissipation in the boundary layer. The nose will continue to be the area of lowest temperature for the wedge, because energy is not being supplied through any of the other boundaries (practically the same as the leading edge of the flat plate shown before). Consequently, the heat flow tends to become more and more in the direction of the nose.

Figure 5.37 shows the temperature in the external flow about the wedge. The blunt–nosed wedge causes a detached bow shock, which is easily seen upstream of the nose. The fluid temperature approaches the stagnation temperature of 293 K (behind the shock and along the symmetry boundary), except right at the interface of the wedge; the fact that this stagnation temperature is below the temperature of the wedge is the reason for the nose being the coolest portion of the wedge.

Transient Two–Dimensional Total–Temperature Probe

The final test case that assumes the coupling of the flow solver with the heat–conduction solver is based on a total–temperature measuring probe. The description and tests of this device are given by Buttsworth, *et al.* [44]. Figure 5.38 gives a view of the overall grid structure. It consists of 4 fluid blocks and 1 solid block; the dimensions are given in the Table 5.7 following the figure . Initial spacing at the nose of the probe for the fluid block is 3e–06 m and 6e–06 m for the solid block modeling the probe.

Again, a two–dimensional approximation is being made to what is indeed a three–dimensional problem; the actual measuring device is composed of two cylindrical probes in close proximity, with nearly–spherical ends (one of these probes is heated, while the other is not).

The test time is just over 0.5 seconds, and it is reasoned that little effect in the circumferential direction (with respect to heat flow) will occur in this time. Also, only one of the probes will be modeled since the measuring end of each experiences very nearly the same condition as the other. Therefore, a two–dimensional grid with adiabatic conditions is applied to the constant–k surfaces (again in the plane of flow) and a symmetry plane along the centerline is used.

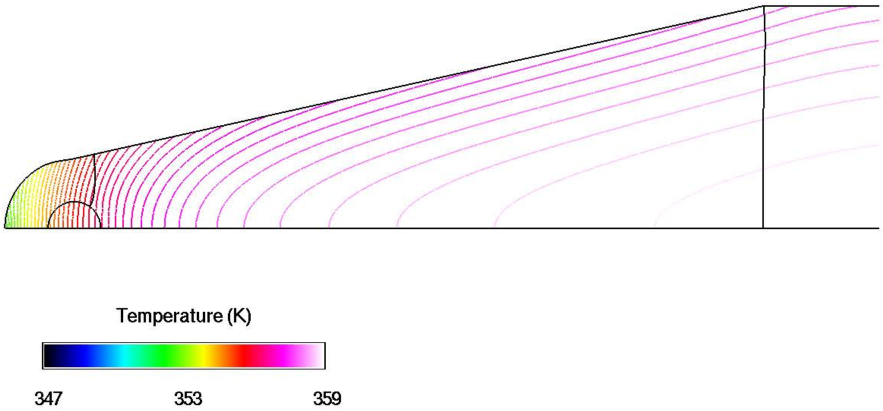


Figure 5.34 Temperature Contours for 4–Block Blunt–Nosed Wedge, Time = 0.33 Seconds

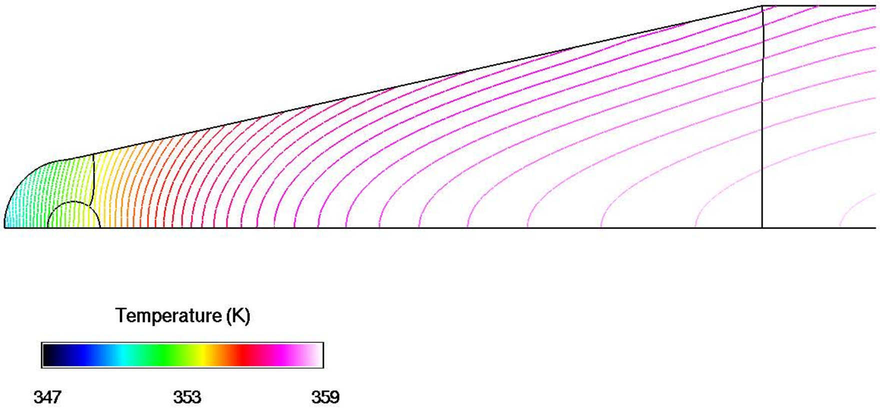


Figure 5.35 Temperature Contours for 4–Block Blunt–Nosed Wedge, Time = 0.65 Seconds

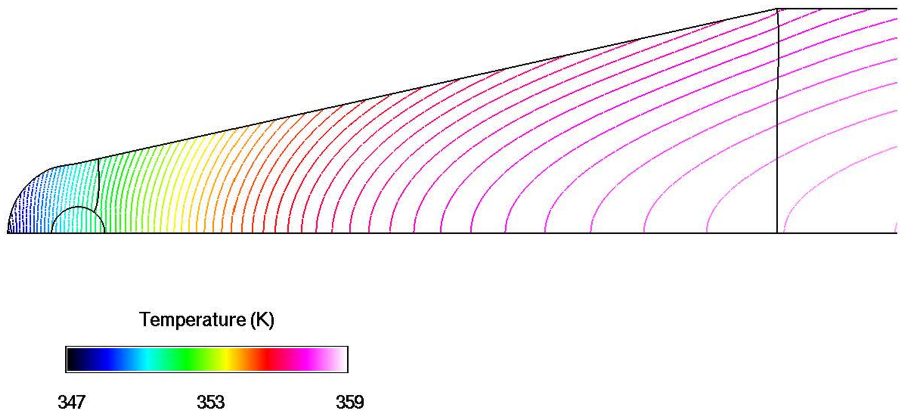


Figure 5.36 Temperature Contours for 4–Block Blunt–Nosed Wedge, Time = 1.0 Second

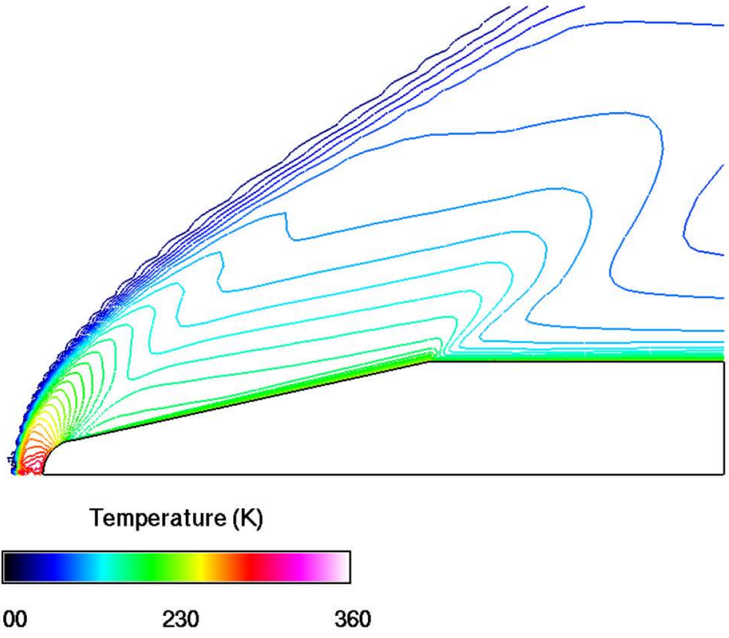


Figure 5.37 Temperature Contours of Flow External to Blunt–Nosed Wedge

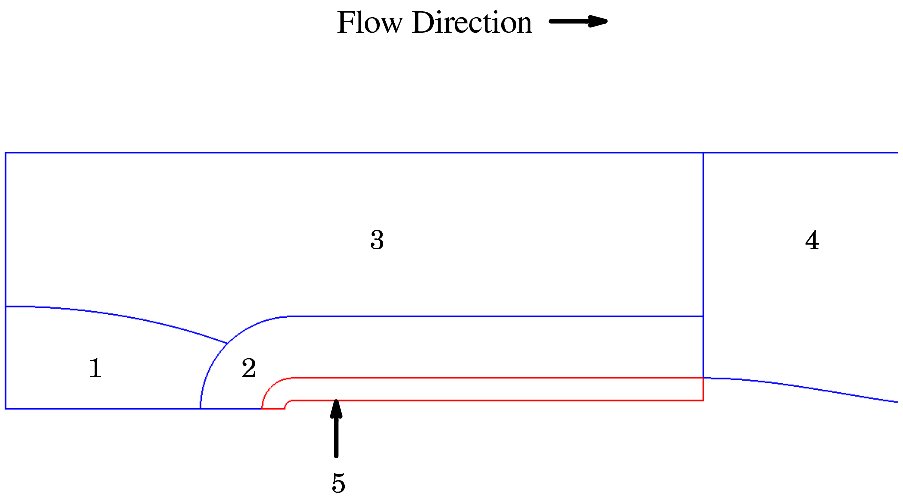


Figure 5.38 Grid Blocking and Numbering for 2–D Total–Temperature Probe

Table 5.7 Block Numbers and Dimensions for Total–Temperature Probe Grid

Block Number	Grid Dimension
1 (fluid)	41 x 21 x 2
2 (fluid)	136 x 41 x 2
3 (fluid)	156 x 31 x 2
4 (fluid)	71 x 71 x 2
5 (solid)	136 x 31 x 2

The flow solver is impulsively started with a free–stream Mach Number of 0.38. Free–stream pressure is set to approximately 91.7 kPa, free–stream temperature to 285 K. These values correspond to total pressure and temperature of 1 atmosphere and 293 K, respectively. The time step is set to 2e–05 seconds and maintained at this value for the entire run; the length of the run was 0.5 seconds. Three Newton sub–iterations were executed for the fluid block (none for the solid), and the time accuracy was kept at 1st–order.

Viscous fluxes for the j–direction (normal to the fluid–solid interface) were calculated with turbulent eddy viscosities included for the blocks nearest the probe.

The solid block was initialized to a constant temperature of 410 K in accordance with Buttsworth [44]. All boundaries except that in contact with the fluid were treated as adiabatic. Note that a portion of the inner boundary of the solid is off of the centerline. This accounts for the fact that part of the probe is hollow to allow space for a small electric heater to be inserted. The probe is made of fused quartz and the thermophysical properties for this (at 400 K) are conductivity, κ = 1.51 W/(m–K), density, ρ = 2220 kg/m^3 , and specific heat, c = 905 J/(kg–K). The conductivity and specific heat were adjusted at various times within the run by linear interpolation with these same values at 300 K (κ = 1.38 W/(m–K) and c = 745 J/(kg–K), respectively).

Temperature values at the nose of the probe in the fluid block (number 2) were recorded at various time levels. These values were taken from the files used to view the solution. The difference between the temperatures at the surface and just off the surface (i.e., j = 1 and j = 2) was taken to calculate a 1st–order temperature derivative normal to the probe surface. The product of this approximate derivative and the conductivity of air was taken to get a heat flux at the nose of the probe. The conductivity of air was adjusted at 10–degree intervals by linear interpolation between the values at 400 K and 350 K (0.0338 and 0.03 W/(m–K)). These property values for quartz and air were again taken from the appendices of Incropera and DeWitt [41].

Figure 5.39 is a plot of the nose temperatures vs time. The time is listed from 0 – 600 ms to agree with the plots from Buttsworth [44] (actual flow was not initiated until 100 ms, at which point a diaphragm was ruptured). The numerical values agree with experiment quite well for the first 50 – 100 ms. Disagreement after that point continues to increase for the duration of the run. Though the quantitative comparison of the numerical and experimental temperatures is not perfect, qualitative results are in good agreement.

Notice that the trend of the numerical curve matches that of the experiment with no abrupt changes.

Figure 5.40 plots the heat flux at the nose (calculated as described above) vs time. As can be expected from the temperature plot, the agreement is the best in the 100 – 200 ms time frame. Beyond that point, the numerical results disagree with experiment. The numerical values predict a higher heat flux than experiment. This is physically consistent with the temperature plot: the higher–than–experiment nose surface temperature will have a greater temperature difference with the incoming air flow, and, thus, a higher heat flux. As with the temperature plot, the encouraging aspect of this plot is that the shape of the numerical curve matches very well with that of the experiment. Even though the agreement is not perfect, the numerical and experimental values are relatively close.

Figure 5.41 through Figure 5.43 give a view of the temperature contours around and within the probe. These pictures are focused on the nose region of the probe, and the solid black lines mark the edges of the probe grid. The flow can be seen to separate near where the leading–edge curvature turns fully downstream (this separation region continues to grow and extend downstream with time). The contours within the probe evolve in a manner similar to that of the wedge: the heat flow takes on an early pattern of being primarily in the upstream direction toward the nose. The temperatures can be seen to continue to drop at the nose and within the probe. The multi–dimensional effect of the energy diffusion can certainly be seen.

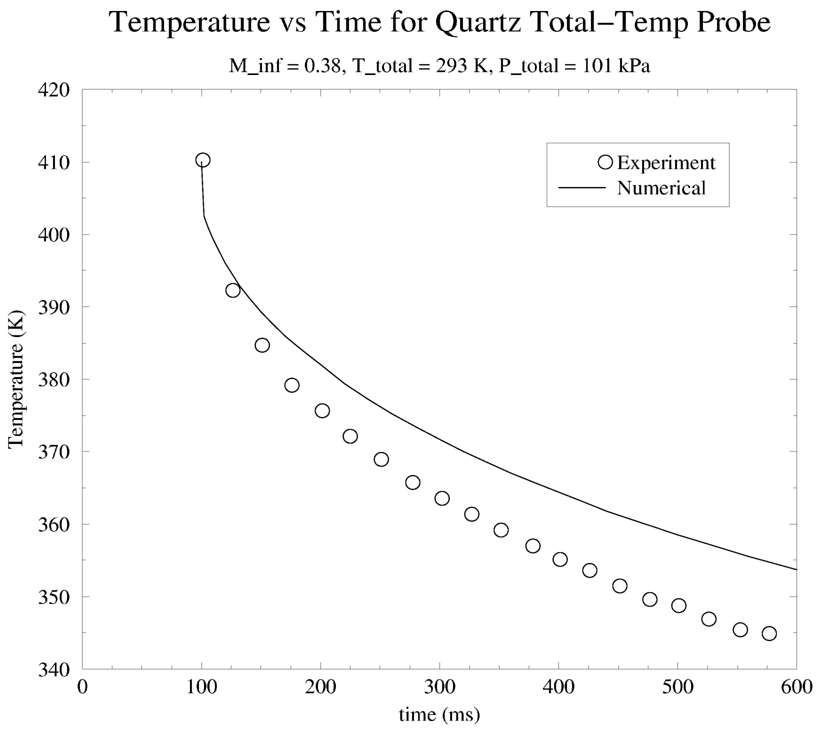


Figure 5.39 Temperature vs. Time at Nose of Total–Temperature Probe

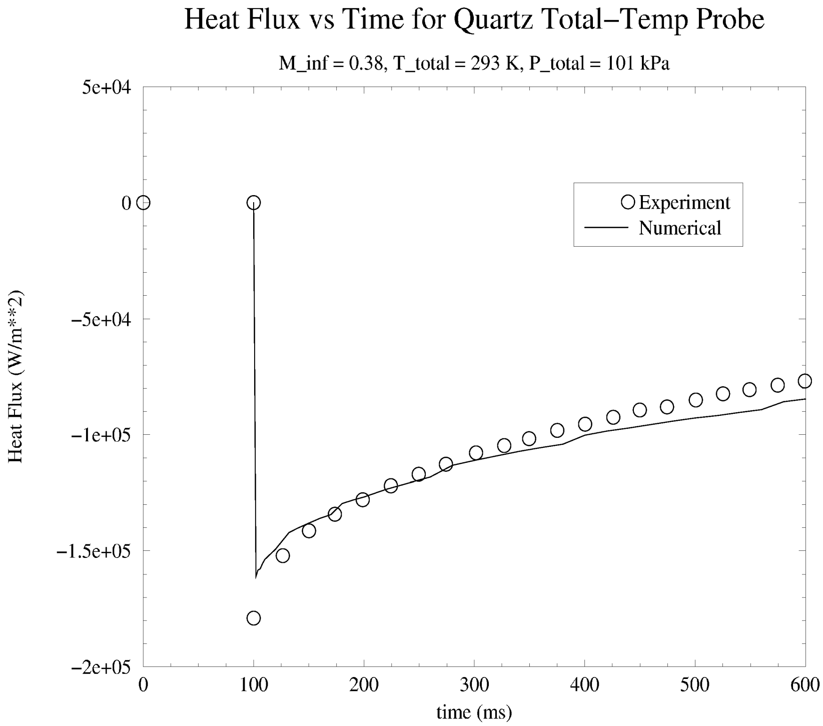


Figure 5.40 Heat Flux vs Time at Nose of Total–Temperature Probe

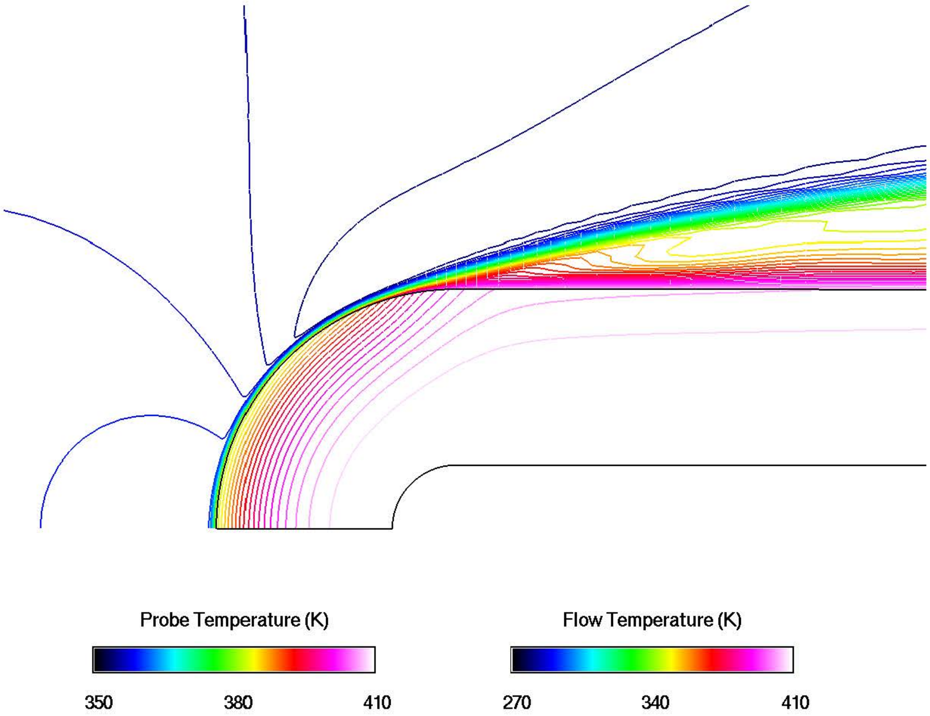


Figure 5.41 Temperature Contours for Total–Temperature Probe, Time = 200 ms

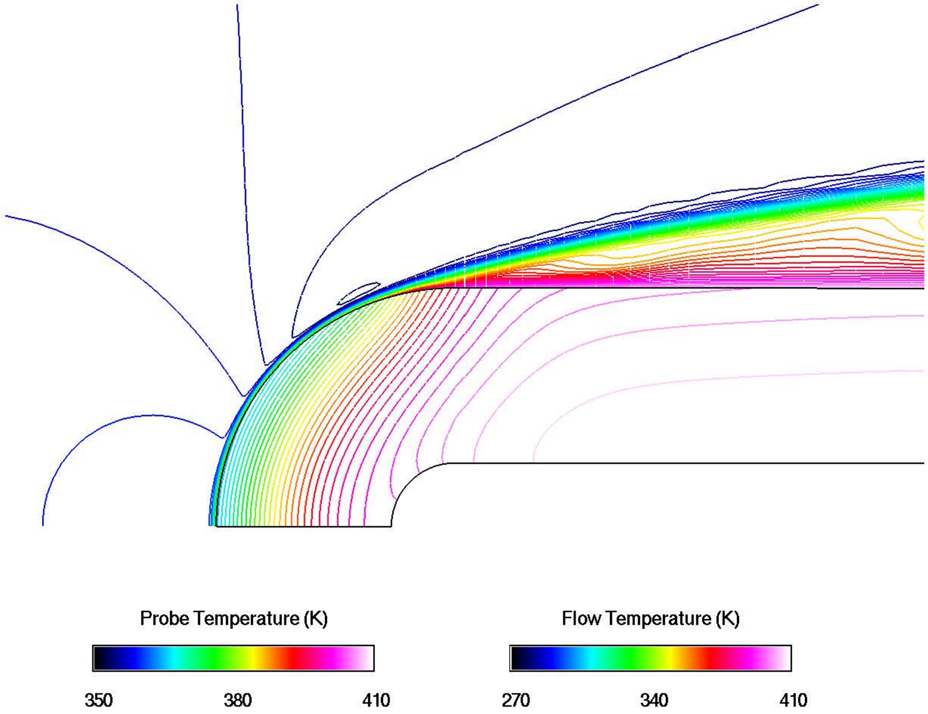


Figure 5.42 Temperature Contours for Total–Temperature Probe, Time = 400 ms

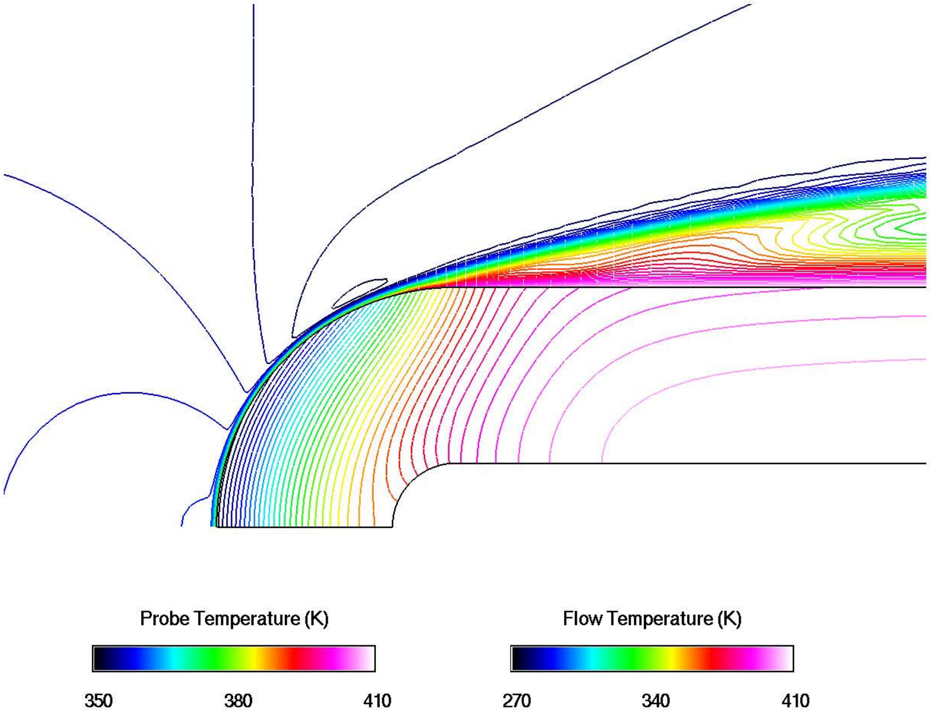


Figure 5.43 Temperature Contours for Total–Temperature Probe, Time = 600 ms

CHAPTER VI

SUMMARY AND CONCLUSIONS

A numerical study of the coupling of conduction and convection heat transfer on multi–block, structured grids has been presented. A portion of this study involved the development of a heat–conduction code for the solution of the time–dependent, multi–dimensional heat equation. Another aspect of the study was the coupling of this heat–conduction solver with an existing flow solver.

The primary focus of this study has been on thermal communication of the heat conduction terms across block–to–block grid boundaries, treated in a time–accurate context. Each grid block has been solved individually and then coupled with neighboring grid blocks, either fluid or solid, by means of the continuity of heat flux (i.e., thermal energy) and temperature at the respective block–to–block grid interface. This somewhat loose coupling is a potential source of stability problems at the block–to–block interfaces, depending on the ratio of the Fourier Numbers of each block at the interface. The Fourier Number can be viewed as a non–dimensional time (like the CFL number for wave–type problems) and is a good measure of the time–dependent nature of the diffusion of the thermal energy within the individual blocks.

If the materials on either side of an interface have a substantial difference in their respective thermal diffusivities, then these materials may be viewed to be decoupled by their very nature. The degree of this decoupling, of course, depends on the difference in their diffusivities. Consequently, it is not surprising that materials with a greater difference in Fourier Numbers, often due to the difference in diffusivities, are more stable with respect to the block–to–block thermal communication that has been outlined here. Conversely, the more closely the values of Fourier Number across an interface, the more prone to stability problems is the interface, especially when solid blocks are communicating with other solid blocks (i.e., the pure diffusion problem). In these cases, either smaller, more–restrictive time steps are required, or a more coupled treatment (such as the "Gauss–Seidel" coupling) is needed. For these reasons, a substantial portion of this study was devoted to the conduction within and communication between solid grid blocks.

Many other numerical and physical aspects of the general conjugate heat transfer problem have not been addressed. The thermophysical properties of the solid materials were always treated as constant, thus resulting in a linear heat equation for the solid grid blocks. Also, the property of thermal expansion was not accounted for. These are important factors that can certainly make a difference in the accuracy of the predictions. However, the potential impact on cell Fourier Numbers (thermophysical properties affecting the diffusivity and thermal expansion affecting the cell size) does not change the fact that the difference in (or ratio of) the Fourier Numbers across an interface is of primary importance. Including variable thermophysical properties and possible expanding cell sizes simply means the difference in Fourier Numbers across the interface will not be constant. Also, radiation is not being accounted for, and there are a number of problems in which this mode of heat exchange can play a prominent role.

The thermal communication between solid and fluid blocks proved to be of relative ease with respect to stability. Much of this is due to the fact that the time scale for fluid problems is often much less than that for the diffusion in the solid. Therefore, using a minimum time–stepping based on the fluid problem is small enough that possible stability problems with, and within, the solid are not encountered. If a solid could possibly be contained within one grid block (either because of a simple solid geometry or perhaps through the use of unstructured gridding for more complicated geometries), then the question of stability problems between solid grid blocks becomes moot; the coupled fluid–solid problem itself has not been a computational challenge for the cases studied.

However, in the case of very low–speed flows, or perhaps free–convection problems, diffusion within the fluid could play a more significant role than in higher–speed, forced convection flows. Time scales within the fluid and solid(s) may then become closer in magnitude, in which case the fluid–solid communication might become more of a concern. This may be an area that requires future consideration.

In summary, a heat–conduction solver has been written, verified, validated, and coupled to an existing flow solver. Though some important aspects have not been addressed, this study has offered some good insight into the potential concerns and physical behavior of coupled conduction–conduction and conduction–convection heat transfer problems.

REFERENCES

[1] Arabshahi, A., “A Dynamic Multiblock Approach to Solving the Unsteady Euler Equations about Complex Configurations”, Ph.D. Dissertation, Mississippi State University, Mississippi, May 1989.

[2] Whitfield, D.L., “Implicit Upwind Finite Volume Scheme for the Three–Dimensional Euler Equations”, Engineering and Industrial Research Station Report, MSSU–EIRS–ASE–85–1, Mississippi State University, Mississippi State, MS, September 1985.

[3] Whitfield, D.L., Janus, J.M., and Simpson, L.B., “Implicit Finite Volume High Resolution Wave–Split Scheme for Solving the Unsteady Three–Dimensional Euler and Navier–Stokes Equations on Stationary or Dynamic Grids”, Engineering and Industrial Research Station Report, MSSU–EIRS–88–2, Mississippi State University, Mississippi State, MS, February 1988.

[4] Belk, D.M., “Three–Dimensional Euler Equations Solutions on Dynamic Blocked Grids”, Ph.D. Dissertation, Mississippi State University, August 1986.

[5] Simpson, L. B., “Unsteady Three–Dimensinal Thin–Layer Navier–Stokes Solutions on Dynamic Blocked Grids”, Ph.D. Dissertation, Mississippi State University, Mississippi, December 1988.

[6] Gatlin, B., “An Implicit, Upwind Method for Obtaining Symbiotic Solutions to the Thin–Layer Navier–Stokes Equations”, Ph.D. Dissertation, Mississippi State University, Mississippi, August 1987.

[7] Cox, C.F., “An Efficient Solver for Flows in Local Chemical Equilibrium”, Ph.D. Dissertation, Mississippi State University, Mississippi, December 1992.

[8] Lau, S.C., Ong, L.E., and Han, J.C., “Conjugate Heat Transfer in Channels with Internal Longitudinal Fins”, *Journal of Thermophysics and Heat Transfer*, Vol. 3, No. 3, July–August 1989, pp. 303–308.

[9] Yu, W–S, Lin, H–T, and Hwang, T–Y, “Conjugate Heat Transfer of Conduction and Forced Convection Along Wedges and a Rotating Cone”, *International Journal of Heat and Mass Transfer*, Vol. 34, No. 10, 1991, pp. 2497–2506.

[10] Pozzi, A., and Lupo,M., “The Coupling of Conduction with Forced Convection Over a Flat Plate”, *International Journal of Heat and Mass Transfer*, Vol. 32, No. 7, 1989, pp. 1207–1213.

[11] Trevino, C., Espinoza, A., and Mendez, F., “Steady–State Analysis of the Conjugate Heat Transfer Between Forced Counterflowing Streams”, *Journal of Thermophysics and Heat Transfer*, Vol. 10, No. 3, July–August 1996, pp. 476–483.

[12] Joubert, P., and Le Quere, P., “Numerical Study of Thermal Coupling Between Conductive Walls and a Boussinesq Stratified Fluid”, *Numerical Heat Transfer, Part A*, Vol. 16, 1989, pp. 489–506.

[13] Bernier, M.A., and, “Conjugate Conduction and Laminar Mixed Convection in Vertical Pipes for Upward Fow and Uniform Wall Heat Flux”, *Numerical Heat Transfer, Part A*, Vol. 21, 1992, pp. 313–332.

[14] Shope, F.L., “Conjugate Conduction–Convection Heat Transfer with a High–Speed Boundary Layer”, *Journal of Thermophysics and Heat Transfer*, Vol. 8, No. 2, April–June 1994, pp. 275–281.

[15] Janus, J.M., and Newman, J.C., “Aerodynamic and Thermal Design Optimization for Turbine Airfoils”, *AIAA–2000–0840*, January, 2000.

[16] Rahaim, C.P., Kassab, A.J., and Cavalleri, R.J., “Coupled Dual Reciprocity Boundary Element/Finite Volume Method for Transient Conjugate Heat Transfer”, *Journal of Thermophysics and Heat Transfer*, Vol. 14, No. 1, January–March 2000, pp. 27–38.

[17] Sondak, D.L., and Dorney, D.J., “Simulation of Coupled Unsteady Flow and Heat Conduction in Turbine Stage”, *Journal of Propulsion and Power*, Vol. 16, No. 6, November–December 2000, pp. 1141–1148.

[18] Chang, K.C., and Payne, U.J., “Numerical Treatment of Diffusion Coefficients at Interfaces”, *Numerical Heat Transfer, Part A*, Vol. 21, 1992, pp. 363–376.

[19] Shyy, W., and Burke, J., “Study of Iterative Characteristics of Convective–Diffusive and Conjugate Heat Transfer Problems”, *Numerical Heat Transfer, Part B*, Vol. 26, 1994, pp. 21–37.

[20] Ruiz, O.E., and Black, W.Z., “A Conservative Iterative–Based Zonal Decomposition Scheme for Conduction Heat Transfer Problems”, *Journal of Heat Transfer*, Vol. 121, February 1999, pp. 169–173.

[21] Janus, J.M., “Advanced 3–D CFD Algorithm for Turbomachinery”, Ph.D. Dissertation, Mississippi State University, Mississippi, May 1989.

[22] Chen, J.P., “Unsteady Three–Dimensional Thin–Layer Navier–Stokes Solutions for Turbomachinery in Transonic Flow”, Ph.D. Dissertation, Mississippi State University, Mississippi, December 1991.

[23] Thompson, J.F., Warsi, Z.U.A., and Mastin, C.W., Numerical Grid Generation, Foundations and Applications, Elsevier Science Publishing Co., Inc., 1985, pp. 95–134.

[24] White, F.M., Viscous Fluid Flow, McGraw–Hill, Inc., 1991, pp. 23–31.

[25] Warsi, Z.U.A., Fluid Dynamics, Theoretical and Computational Approaches, CRC Press, Inc., 1993, pp. 59–59.

[26] Baldwin, B.S., and Lomax, H., "Thin–Layer Approximation and Algebraic Model for Separated Turbulent Flows", *AIAA–78–257*, January, 1978.

[27] Briley, W.R., and McDonald, H., "Solution of the Multidimensional Compressible Navier–Stokes Equations by Generalized Implicit Method", *Journal of Computational Physics*, Vol. 24, 1977, pp. 372–397.

[28] Vanden, K.J., and Whitfield, D.L., "Direct and Iterative Algorithms for the Three–Dimensional Euler Equations", *AIAA Journal*, Vol. 33, No. 5, May 1995, pp. 851–858.

[29] Cinnella, P., "Flux–Split Algorithms for Flows with Non–Equilibrium Chemistry and Thermodynamics", Ph.D. Dissertation, Virginia Polytechnic Institute and State University, Virginia, December 1989.

[30] Godunov, S.K., "Finite Difference Method for Numerical Comutation of Discontinuous Solutions of the Equations of Fluid Dynamics", *Mat. Sbornik*, Vol. 47, No. 3, 1959, pp. 271–306. Translated as JPRS 7225 by the US Department of Commerce, November 1960.

[31] Hirsch, C., Numerical Computation of Internal and External Flows, Volume 2, John Wiley and Sons, Ltd., 1990, pp. 443–472.

[32] Roe, P.L., "Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes", *Journal of Computational Physics*, Vol. 43, 1981, pp. 357–372.

[33] Quirk, J.J., "A Contribution to the Great Riemann Solver Debate", *NASA CR 191409, ICASE Report 92–64*, December 1992.

[34] Harten, A., Lax, P.D., and van Leer, B., "On Upstream Differencing and Godunov–Type Schemes for Hyperbolic Conservation Laws", *SIAM Review*, Vol. 25, 1983, pp. 36–61.

[35] Osher, S.R., and Chakravarthy, S., "Very High Order Accurate TVD Schemes", *ICASE Report 84–44*, September 1984.

[36] Whitfield, D.L., "Newton–Relaxation Schemes for Nonlinear Hyperbolic Systems", *Engineering and Industrial Research Station Report, MSSU–EIRS–ASE–90–3*, Mississippi State University, Mississippi, October 1990.

[37] Beam, R.M., and Warming, R.F., "An Implicit Finite–Difference Algorithm for Hyperbolic Systems in Conservation Law Form", *Journal of Computational Physics*, Vol. 22, 1976, pp. 87–110.

[38] Dennis, J.E., Jr., and Schnabel, R.B., Numerical Methods for Unconstrained Optimization and Nonlinear Equations, Prentice–Hall, Inc., 1983, pp. 86–92.

[39] Whitfield, D.L., and Janus, J.M., “Three–Dimensional Unsteady Euler Equations Solution Using Flux Vector Splitting”, *AIAA–84–1552*, June, 1984.

[40] Pankajakshan, R., Private Communication, December, 2000.

[41] Incropera, F.P., and DeWitt, D.P., Fundamentals of Heat and Mass Transfer, John Wiley and Sons, Inc., 1985, pp. 222–227.

[42] Özisik, M.N., Boundary Value Problems of Heat Conduction, Dover Publications, Inc., 1989, pp. 43–124.

[43] Cavalleri, R.J., and Tiarn, W., “CFD Evaluation of an Advanced Thrust Vector Control Concept”, *AIAA–90–100*, January 1990.

[44] Buttsworth, D.R., Jones, T.V., and Chana, K.S., “Unsteady Total Temperature Measurements Downstream of a High–Pressure Turbine”, *Journal of Turbomachinery*, Vol. 120, October 1998, pp. 760–767.