MEC ENG 151B/250B: Convective Transport and Computational Methods

Project 2: Natural Convection Boundary Layer Flow

Professor Van P. Carey
UC Berkeley, Department of Mechanical Engineering
April 10, 2025

Larry Hui[1] and Derek Shah
SID: 3037729658

[1]University of California at Berkeley, College of Engineering, Department of Mechanical Engineering. Author to whom any correspondence should be addressed. email: larryhui7@berkeley.edu

Abstract

This project was completed as a requirement for MEC ENG 151B at the University of California at Berkeley. We present a numerical analysis of boundary layer flow caused by natural convection from a uniformly applied surface heat flux on a vertical plate. We utilized 3 common approaches to solve this problem. We started off implementing the Forward-Time-Central-Space (FTCS) finite-difference (FD) method to simulate the transient behaviour of the field variables over a 2D mesh. We were able to capture and visualize key transient phenomena including the leading-edge effect, various field variable surface plots, as well as the time at which our flow reached steady-state. Secondly, we used a more traditional similarity variables approach to transform the steady-state boundary layer equations into a Blasius-type ODE, which we solved using the fourth-order Runge-Kutta (RK4) method and state variables for simplification. This served as a benchmark to verify that the FD model results were accurate and demonstrated the self-similar nature of the flow. Lastly, an integral boundary layer analysis from the standpoints of momentum and energy conservation provided approximate expressions for the velocity and temperature profiles, which gave further physical insights into the scaling behavior of the boundary layer thickness.

These methods were then used to examine the heat transfer regarding the cooling of an electronics component mounted on a vertical circuit board, where we have both natural convection coupled with radiative heat transfer. Our solutions demonstrated convergence and similarities across the various methods, with consistent predictions of flow and thermal characteristics under varying operating conditions. In general, results confirm the accuracy and stability of the numerical methods used, and provide a solid foundation for analyzing convective heat transfer phenomena in applications.

Contents

List of Figures

1 Introduction and Theoretical Background

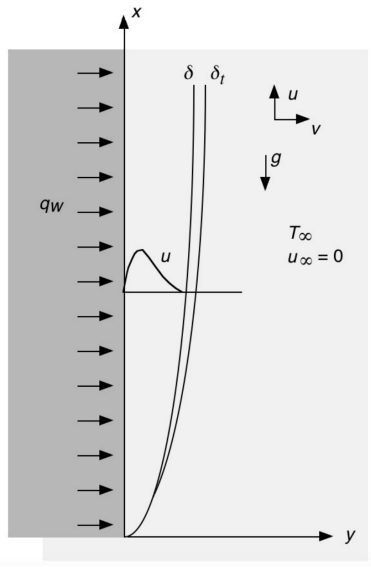

Figure 1: Natural Convection Boundary Layer Flow near a Surface with Uniform Heat Flux

This project will focus on understanding the natural convection boundary layer flow near a surface with a specified uniform heat flux as shown above in **Figure 1**. We consider three different ways of analyzing this type of two-dimensional convection heat transfer process. Invoking the usual boundary layer approximations, the Boussinesq approximations, neglecting pressure work and viscous dissipation, the governing equations, and the initial and boundary conditions for time-varying flow are:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \tag{1}$$

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = g\beta(T - T_\infty) + \nu\frac{\partial^2 u}{\partial y^2} \tag{2}$$

$$\frac{\partial T}{\partial t} + u\frac{\partial T}{\partial x} + v\frac{\partial T}{\partial y} = \alpha\frac{\partial^2 T}{\partial y^2} \tag{3}$$

$$\text{at } t = 0: \quad u = v = 0,\ T = T_\infty \quad \forall x, y \tag{4a}$$

$$\text{for } t > 0: \quad \text{at } y = 0: u = v = 0,\ \frac{\partial T}{\partial y} = -\frac{q_w}{k}, \quad \text{at } y = \infty: u = 0,\ T = T_\infty \tag{4b}$$

The initial and boundary conditions model the time-varying buoyancy-drive flow and heat transfer resulting from sudden application of the uniform heat flux at the surface at time $t = 0$, with zero u velocity and $T = T_\infty$ everywhere at $t = 0$.

1.1 Work Division and Tasks

The tasks for this project were distributed among both team members to ensure efficient collaboration and thorough completion of all required components. Each member contributed to different aspects of the project as follows:

1. The determination of the values of the constants C_1 and C_2 in the similarity analysis; the derivation using the integral equations and functional forms; the determination of the relations for the velocity field, $u(x, y/\delta)$, the temperature field $T(x, y/\delta)$ and the boundary layer thickness $\delta(x)$—including predicting and plotting u and T profiles—were done by **Larry Hui** and reviewed by **Derek Shah**. All of this work is documented in a step-by-step approach in Section 1.2.

2. The program development (including description of code features and implementation of coding) was mainly done by **Derek Shah** who implemented in MATLAB both the explicit Forward-Time-Central-Space (FTCS) algorithm as well as the code used to solve ODE's numerically, namely the 4^{th} order Runge-Kutta scheme (RK4). **Larry Hui** helped with debugging issues and describing code features.

3. Running computations for different cases with different geometries and parameter sets was done by **both of us**; this sped up this section of work and ensured that we could double check the results and make sure they aligned with our intuition.

4. Analysis of the results was done by **Derek Shah** who processed, interpreted, and computed results and the relevant plots. The Electronic Device Cooling question was also done by both of us, with **Derek Shah** implementing the finite difference formulation and **Larry Hui** answering question qualitatively.

5. Write-up of results and conclusions are done by both **Larry Hui** and **Derek Shah**. Derivations and code development are all integrated into one cohesive document. The conclusions were drawn based on numerical findings, and recommendations for further improvements were discussed.

This project is split into 4 tasks which are briefly described below.

1. **Task I**: This task deals with a Finite-Difference Numerical Simulation of Natural Convection. We derive and discretize the governing boundary layer equations for both velocity and temperature fields using finite-difference approximations. An explicit FTCS algorithm is implemented on a 51×51 node mesh to simulate the transient natural convection induced by a suddenly applied uniform heat flux. From this numerical scheme, we generate 3D surface plots of the temperature and u-velocity fields. We further analyze key transient phenomena—such as the leading-edge effect and the time required for the surface temperature to attain 99% of its steady state—by examining variations in the field variables along the surface. A mathematical formulation for the numerical analysis of boundary layer flow (**Task I**) is found in Section 1.2.1. Further discussion and results can be found in Section 3.1.

2. **Task II**: This task deals with similarity transformation analysis. This task reformulates the steady-state boundary layer equations using similarity transformation variables. By replacing the original spatial variables and field functions with their similarity counterparts, the partial differential equations are reduced to a system of ordinary differential equations. The resulting boundary value problem is solved via a fourth-order Runge-Kutta method. The similarity solution is then used to transform the finite-difference results for comparison, allowing us to validate the consistency of the numerical predictions and examine the self-similar behavior of the flow. A mathematical description i.e. parts (2.1) and a subset of (2.2) can be found in Section 1.2.2.

3. **Task III**: This task is about performing an integral boundary layer analysis. Here, we develop an integral method by applying momentum and energy balances over a finite control volume. Assumed functional forms for the velocity and temperature profiles are substituted into the integral equations to derive relationships for the scaling exponents and coefficients governing the x-dependence of the solution. This approach yields explicit expressions for the $u(x, y/\delta)$ and $T(x, y/\delta)$ fields, which are then compared with both the finite-difference and similarity solutions. The integral analysis provides an approximate yet physically insightful representation of the boundary layer behavior. A complete description of this task can be found in Section 1.2.3.

4. **Task IV**: We extend the preceding analyses to a practical application involving the cooling of an electronic device mounted on a vertical circuit board. In this scenario, natural convection is coupled with radiative heat transfer by modifying the boundary conditions to account for thermal radiation exchange with a surrounding enclosure. We assess the maximum allowable heat generation rate based on a prescribed surface temperature limit and evaluate the effect of gravitational variations (including lunar gravity) on the cooling performance. This integrated analysis addresses both the convective and radiative contributions to heat rejection from the device. Results and discussion are found in Section 3.4.

1.2 Theoretical Background

This section aims to give a brief overview to all the mathematical formulation and analytical approaches used throughout the project to model and understand natural convection boundary layer flow over a vertically heated surface. The solutions to some of the tasks are also shown here, namely **Task II** and **Task III**.

We will start with the numerical finite-difference formulation, followed by similarity analysis, and integral solutions. Each method offers unique insights into the physical behavior of the system—ranging from time-dependent simulations capturing transient startup phenomena, to reduced-order similarity solutions describing steady-state behavior, and approximate integral formulations providing closed-form expressions for flow characteristics. Together, these approaches form a comprehensive framework for analyzing buoyancy-driven convection and validating numerical results across varying levels of complexity.

1.2.1 Numerical Analysis Formulation

In this first task, the system of equations, initial conditions and boundary conditions is to be solved on the finite-difference mesh shown in the diagram below.

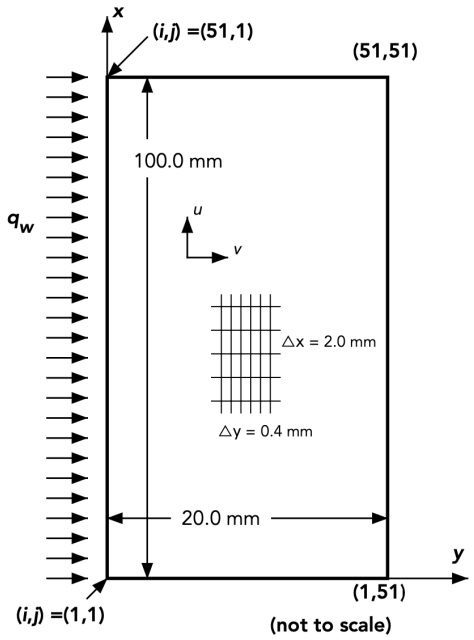

Figure 2: Finite Difference Mesh for Boundary Layer Analysis

First-order upwind differences are used for the first derivatives in the convection terms. Otherwise, backward differences are used for first derivatives. For second order derivatives, central differences are used. The resulting finite difference equations are

$$\frac{T'_{i,j} - T_{i,j}}{\Delta t} + u_{i,j}\frac{T_{i,j} - T_{i-1,j}}{\Delta x} + v_{i,j}\frac{T_{i,j+1} - T_{i,j}}{\Delta y} = \alpha\frac{T_{i,j+1} - 2T_{i,j} + T_{i,j-1}}{(\Delta y)^2} \tag{5}$$

$$\frac{u'_{i,j} - u_{i,j}}{\Delta t} + u_{i,j}\frac{u_{i,j} - u_{i-1,j}}{\Delta x} + v_{i,j}\frac{u_{i,j+1} - u_{i,j}}{\Delta y} = g\beta(T'_{i,j} - T_\infty)\frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{(\Delta y)^2} \tag{6}$$

$$\frac{u'_{i,j} - u'_{i-1,j}}{\Delta x} + \frac{v'_{i,j} - v'_{i,j-1}}{\Delta y} = 0 \tag{7}$$

We rearrange the above equations to solve for u, v and T at the next time step

$$T'_{i,j} = T_{i,j} + \left[\alpha\frac{T_{i,j+1} - 2T_{i,j} + T_{i,j-1}}{(\Delta y)^2} - u_{i,j}\frac{T_{i,j} - T_{i-1,j}}{\Delta x} - v_{i,j}\frac{T_{i,j+1} - T_{i,j}}{\Delta y}\right]\Delta t$$

$$u'_{i,j} = u_{i,j} + \left[g\beta(T_{i,j} - T_\infty) + \nu \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{(\Delta y)^2} - u_{i,j} \frac{u_{i,j} - u_{i-1,j}}{\Delta x} - v_{i,j} \frac{u_{i,j+1} - u_{i,j}}{\Delta y} \right] \Delta t \tag{8}$$

$$v'_{i,j} = v'_{i,j-1} - \left[\frac{u'_{i,j} - u'_{i-1,j}}{\Delta x} \right] \Delta y$$

The heat flux boundary condition at $y = 0$ is represented in finite difference form as

$$\frac{-q_w}{k} = \frac{T_{i,2} - T_{i,1}}{\Delta y} \tag{9}$$

The explicit Forward-Time-Central Space (FTCS) algorithm is to be used to advance the temperature and velocity fields in time. To implement it, a computer program must be created that executes the following operations:

Algorithm 1 Forward-Time-Central Space (FTCS)

1: Create arrays to store the old (unprimed) and new (primed) field variables at each node

$$u_{i,j},\ v_{i,j},\ T_{i,j},\ u'_{i,j},\ v'_{i,j},\ T'_{i,j} \quad \text{where } 1 \le i \le 51 \text{ and } 1 \le j \le 51.$$

2: Initialize old and new fields everywhere to the values

$$u_{i,j} = 0, \quad v_{i,j} = 0, \quad T_{i,j} = T_\infty$$
$$u'_{i,j} = 0, \quad v'_{i,j} = 0, \quad T'_{i,j} = T_\infty$$

3: Compute the fields at the next time step at all non-boundary points

$$T'_{i,j} = T_{i,j} + \left[\alpha \frac{T_{i,j+1} - 2T_{i,j} + T_{i,j-1}}{(\Delta y)^2} - u_{i,j} \frac{T_{i,j} - T_{i-1,j}}{\Delta x} - v_{i,j} \frac{T_{i,j+1} - T_{i,j}}{\Delta y} \right] \Delta t$$
$$u'_{i,j} = u_{i,j} + \left[g\beta(T_{i,j} - T_\infty) + \nu \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{(\Delta y)^2} - u_{i,j} \frac{u_{i,j} - u_{i-1,j}}{\Delta x} - v_{i,j} \frac{u_{i,j+1} - u_{i,j}}{\Delta y} \right] \Delta t$$
$$v'_{i,j} = v'_{i,j-1} - \left[\frac{u'_{i,j} - u'_{i-1,j}}{\Delta x} \right] \Delta y$$

4: The temperatures along the wall at $y = 0$ at the next time step are computed as

$$T'_{i,1} = \frac{q_w}{k} \Delta y + T'_{i,2}$$

5: If time has exceeded the specified limit, exit the algorithm. If not, store new field values in old field variables at each point

$$u_{i,j} = u'_{i,j}, \quad v_{i,j} = v'_{i,j}, \quad T_{i,j} = T'_{i,j}$$

and return to step [3].

1.2.2 Similarity Analysis

For the purposes of this project, we define the similarity variables as follows

$$\eta \triangleq \frac{y}{x} \left(\frac{g\beta q_w x^4}{5k\nu^2} \right)^{1/5} \quad , \quad \psi \triangleq 5\nu \left(\frac{g\beta q_w x^4}{5k\nu^2} \right)^{1/5} F(\eta) \quad , \quad H(\eta) \triangleq \frac{T - T_\infty}{q_w x/k} \left(\frac{g\beta q_w x^4}{5k\nu^2} \right)^{1/5} \tag{10}$$

we can show that the replacement of y, u, v, and T with transform variables satisfies the continuity equation and converts the *steady forms* of the u-momentum and energy equations, and associated boundary conditions to

$$F''' - C_{2.1} F'^2 + C_{2.2} F F'' + H = 0 \tag{11}$$

$$H'' + \Pr(4FH' - F'H) = 0 \tag{12}$$

$$F(0) = 0, \quad F'(0) = 0, \quad H'(0) = -1 \tag{13a}$$

$$F'(\infty) = 0, \quad H(\infty) = 0 \tag{13b}$$

where $C_{2.1}$ and $C_{2.2}$ are integer numerical constants i.e. $C_{2.1}, C_{2.2} \in \mathbb{Z}$.

Task II.i Continuity Equation: Firstly, we show that the replacement of y, u, v, and T with the transform variables described above satisfy **Eq. 11–13b**.

Proof. By invoking the Boussinesq approximations, density is taken to be constant when studying flow implying incompressibility as the material derivative of ρ is zero i.e. $\frac{D\rho}{Dt} = 0$. Thus we introduce a scalar stream function $\psi(x, y)$ with the velocity components of the flow defined as follows

$$u \triangleq \frac{\partial \psi}{\partial y} \quad \text{and} \quad v \triangleq -\frac{\partial \psi}{\partial x}$$

Simply by plugging in u and v in terms of ψ into the 2D incompressible flow continuity equation we get

$$\begin{aligned} \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} &= 0 \\ \frac{\partial}{\partial x}\left(\frac{\partial \psi}{\partial y}\right) + \frac{\partial}{\partial y}\left(-\frac{\partial \psi}{\partial x}\right) &= 0 \\ \frac{\partial^2 \psi}{\partial x \partial y} - \frac{\partial^2 \psi}{\partial y \partial x} &= 0 \end{aligned}$$

By Clairaut's Theorem, for any smooth continuous function $\psi(x, y)$ of differentiability class C^2, the order of differentiation does not matter so

$$\frac{\partial^2 \psi}{\partial x \partial y} = \frac{\partial^2 \psi}{\partial y \partial x} \Rightarrow \frac{\partial^2 \psi}{\partial x \partial y} - \frac{\partial^2 \psi}{\partial y \partial x} = 0 \Rightarrow 0 = 0$$

Therefore, we have showed that the replacement of u and v with the transform variable ψ satisfies the continuity equation. □

Now having proved that the continuity equation does hold using a scalar stream function, we extend this idea to the next part of the task: deriving a transformed variable form of the u-momentum equation.

Task II.i u-Momentum Equation: Next, we want to convert the steady forms of the u-momentum equation using the transform variables.

Proof. We expand the components of velocity in terms of the stream functions. For notational convenience, we let $\left(\frac{g\beta q_w}{5k\nu^2}\right)^{1/5} \triangleq \xi$

$$\begin{aligned} u \triangleq \frac{\partial \psi}{\partial y} &= \frac{\partial}{\partial y}\left(5\nu\left(\frac{g\beta q_w}{5k\nu^2}\right)^{1/5} x^{4/5} F(\eta)\right) \\ &= 5\nu\xi x^{4/5}\frac{\partial F(\eta)}{\partial y} \end{aligned}$$

By chain rule and defining $\frac{dF(\eta)}{d\eta} \triangleq F'(\eta)$ we now have

$$u = 5\nu\xi x^{4/5}\frac{dF(\eta)}{d\eta} \cdot \frac{\partial \eta}{\partial y}$$

$$
\begin{aligned}
&= 5\nu\xi x^{4/5} F'(\eta) \cdot \frac{\partial}{\partial y}\left(\frac{y}{x}\left(\frac{g\beta q_w}{5k\nu^2}\right)^{1/5} x^{4/5}\right) \\
&= 5\nu\xi x^{4/5} F'(\eta) \cdot \frac{\partial}{\partial y}(y\xi x^{-1/5}) \\
&= 5\nu\xi x^{4/5} F'(\eta)\xi x^{-1/5} \Rightarrow u = 5\nu\xi^2 x^{3/5} F'(\eta)
\end{aligned}
$$

Taking the partial derivatives of u w.r.t. x

$$
\begin{aligned}
\frac{\partial u}{\partial x} &= \frac{\partial}{\partial x}(5\nu\xi^2 x^{3/5} F'(\eta)) \\
&= 5\nu\xi^2 \frac{\partial}{\partial x}(x^{3/5} \cdot F'(\eta)) \\
&= 5\nu\xi^2 \left(\frac{3}{5} x^{-2/5} F'(\eta) + F''(\eta)\left(-\frac{1}{5}\xi y x^{-6/5}\right) x^{3/5}\right)
\end{aligned}
$$

Substituting in $y = \frac{1}{\xi} x^{1/5}\eta$, we get

$$
\frac{\partial u}{\partial x} = 5\nu\xi^2 \left(\frac{3}{5} x^{-2/5} F'(\eta) - \frac{1}{5}\eta x^{-2/5} F''(\eta)\right) \Rightarrow \frac{\partial u}{\partial x} = \nu\xi^2 x^{-2/5}\left[3F'(\eta) - \eta F''(\eta)\right]
$$

Taking the partial w.r.t. y we get

$$
\begin{aligned}
\frac{\partial u}{\partial y} &= \frac{\partial}{\partial y}(5\nu\xi^2 x^{3/5} F'(\eta)) \\
&= 5\nu\xi^2 x^{3/5} F''(\eta)\frac{\partial \eta}{\partial y}
\end{aligned}
$$

From above, we determined that $\frac{\partial \eta}{\partial y} = \xi x^{-1/5}$,

$$
\frac{\partial u}{\partial y} = 5\nu\xi^2 x^{3/5} F''(\eta) \cdot \xi x^{-1/5} \Rightarrow \frac{\partial u}{\partial y} = 5\nu\xi^3 x^{2/5} F''(\eta)
$$

Taking the second partial derivative w.r.t. y we get

$$
\frac{\partial^2 u}{\partial y^2} = 5\nu\xi^3 x^{2/5} F'''(\eta) \cdot \frac{\partial \eta}{\partial y} \Rightarrow \frac{\partial^2 u}{\partial y^2} = 5\nu\xi^4 x^{1/5} F'''(\eta)
$$

We now compute v in a similar fashion.

$$
\begin{aligned}
v \triangleq -\frac{\partial \psi}{\partial x} &= -\frac{\partial}{\partial x}\left(5\nu\left(\frac{g\beta q_w}{5k\nu^2}\right)^{1/5} x^{4/5} F(\eta)\right) \\
&= -5\nu\xi \frac{\partial}{\partial x}(x^{4/5} F(\eta))
\end{aligned}
$$

By the product rule and chain rule, we now have

$$
\begin{aligned}
v &= -5\nu\xi\left[\frac{4}{5} x^{-1/5} F(\eta) + x^{4/5}\frac{dF(\eta)}{d\eta} \cdot \frac{d\eta}{dx}\right] \\
&= -5\nu\xi\left[\frac{4}{5} x^{-1/5} F(\eta) + x^{4/5} F'(\eta)\frac{\partial}{\partial x}\left(\frac{y}{x}\left(\frac{g\beta q_w}{5k\nu^2}\right)^{1/5} x^{4/5}\right)\right] \\
&= -5\nu\xi\left[\frac{4}{5} x^{-1/5} F(\eta) + x^{4/5} F'(\eta) y\xi \frac{\partial}{\partial x}(x^{-1/5})\right] \\
&= -5\nu\xi\left[\frac{4}{5} x^{-1/5} F(\eta) - \frac{1}{5} x^{-2/5} F'(\eta) y\xi\right]
\end{aligned}
$$

Notice that we can rewrite y as $y = \frac{1}{\xi}x^{1/5}\eta$ by rearranging the similarity variable η. Substituting our expression for y into v yields

$$\begin{aligned} v &= -5\nu\left[\frac{4}{5}x^{-1/5}F(\eta) - \frac{1}{5}x^{-2/5}F'(\eta)\frac{1}{\xi}x^{1/5}\eta\right] \\ &= -5\nu\xi\left[\frac{4}{5}x^{-1/5}F(\eta) - \frac{1}{5}x^{-1/5}\eta F'(\eta)\right] \Rightarrow v = -5\nu\xi x^{-1/5}\left[\frac{4}{5}F(\eta) - \frac{\eta}{5}F'(\eta)\right] \end{aligned}$$

Let's rearrange $H(\eta)$ for $T - T_\infty$

$$\begin{aligned} H(\eta) &= \frac{T - T_\infty}{q_w x/k}\left(\frac{g\beta q_w}{5k\nu^2}\right)^{1/5} x^{4/5} \\ T - T_\infty &= \frac{q_w}{k}x^{1/5}\xi^{-1}H(\eta) \end{aligned}$$

We substitute our calculated derivatives into the steady u-momentum equation. Let's calculate the LHS first.

$$\begin{aligned} &\Rightarrow u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} \\ &= 5\nu\xi^2 x^{3/5}F'(\eta)(\nu\xi^2 x^{-2/5}\left[3F'(\eta) - \eta F''(\eta)\right]) - 5\nu\xi x^{-1/5}\left[\frac{4}{5}F(\eta) - \frac{\eta}{5}F'(\eta)\right]5\nu\xi^3 x^{2/5}F''(\eta) \\ &= 5\nu^2\xi^4 x^{1/5}[3F'(\eta)^2 - \eta F'(\eta)F''(\eta)] - [5\nu^2\xi^4 x^{1/5}(4F(\eta)F''(\eta) - \eta F'(\eta)F''(\eta))] \\ &= 5\nu^2\xi^4 x^{1/5}[3F'(\eta)^2 - 4F(\eta)F''(\eta)] \end{aligned}$$

The RHS of the u-momentum equation is

$$\begin{aligned} &\Rightarrow g\beta(T - T_\infty) + \nu\frac{\partial^2 u}{\partial y^2} \\ &= g\beta\left(\frac{q_w}{k}x^{1/5}\xi^{-1}H(\eta)\right) + \nu\left(5\nu\xi^4 x^{1/5}F'''(\eta)\right) \end{aligned}$$

But recall that $\xi^5 = \frac{g\beta q_w}{5k\nu^2} \Rightarrow \frac{g\beta q_w}{k} = 5\nu^2\xi^5$, then we can simplify the RHS to

$$\begin{aligned} &= \left(5\nu^2\xi^5 x^{1/5}\xi^{-1}H(\eta)\right) + \nu\left(5\nu\xi^4 x^{1/5}F'''(\eta)\right) \\ &= \left(5\nu^2 x^{1/5}\xi^4 H(\eta)\right) + \left(5\nu^2\xi^4 x^{1/5}F'''(\eta)\right) \\ &= 5\nu^2\xi^4 x^{1/5}(H(\eta) + F'''(\eta)) \end{aligned}$$

So our u-momentum balance is

$$\begin{aligned} u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} &= g\beta(T - T_\infty) + \nu\frac{\partial^2 u}{\partial y^2} \\ 5\nu^2\xi^4 x^{1/5}[3F'(\eta)^2 - 4F(\eta)F''(\eta)] &= 5\nu^2\xi^4 x^{1/5}(H(\eta) + F'''(\eta)) \\ 3F'^2 - 4FF'' &= H + F''' \\ F''' - 3F'^2 + 4FF'' + H &= 0 \end{aligned}$$

□

Notice now that the use of these similarity variables has now led us to the exact form of **Eq. 11**. Next, we deal with the energy equation.

Task II.i Energy Equation: we want to convert the steady forms of the energy equation equation using the transform variables.

Proof. Next, in order to substitute our values into the steady-state energy equation, we need to find the derivatives of temperature. Taking the first partial derivative of T w.r.t. x yields

$$\frac{\partial(T-T_\infty)}{\partial x} = \frac{\partial}{\partial x}\left(\frac{q_w}{k}\xi^{-1}x^{1/5}H(\eta)\right)$$

By the product rule and chain rule, we get the following expression

$$\begin{aligned}
&= \frac{q_w}{k}\xi^{-1}\left[\frac{1}{5}x^{-4/5}H(\eta) + x^{1/5}H'(\eta)\cdot\frac{\partial\eta}{\partial x}\right] \\
&= \frac{q_w}{k}\xi^{-1}\left[\frac{1}{5}x^{-4/5}H(\eta) + x^{1/5}H'(\eta)\left(-\frac{1}{5}\xi y x^{-6/5}\right)\right] \\
&= \frac{q_w}{k}\xi^{-1}\left[\frac{1}{5}x^{-4/5}H(\eta) + x^{1/5}H'(\eta)\left(-\frac{1}{5}\cancel{\xi}\cdot\frac{1}{\cancel{\xi}}x^{1/5}\eta x^{-6/5}\right)\right] \\
&= \frac{q_w}{k}\xi^{-1}\left[\frac{1}{5}x^{-4/5}H(\eta) + x^{1/5}H'(\eta)\left(-\frac{1}{5}x^{-1}\eta\right)\right] \\
\frac{\partial(T-T_\infty)}{\partial x} &= \frac{q_w}{k}\xi^{-1}\frac{x^{-4/5}}{5}[H(\eta)-\eta H'(\eta)]
\end{aligned}$$

Then taking the partial derivative w.r.t. y we get

$$\begin{aligned}
\frac{\partial(T-T_\infty)}{\partial y} &= \frac{\partial}{\partial y}\left(\frac{q_w}{k}\xi^{-1}x^{1/5}H(\eta)\right) \\
&= \frac{q_w}{k}\xi^{-1}x^{1/5}H'(\eta)\cdot\frac{\partial\eta}{\partial y} \\
&= \frac{q_w}{k}\cancel{\xi^{-1}x^{1/5}}H'(\eta)\cancel{\xi x^{-1/5}} \Rightarrow \frac{\partial(T-T_\infty)}{\partial y} = \frac{q_w}{k}H'(\eta)
\end{aligned}$$

Taking the 2nd partial derivative w.r.t. y leaves us with

$$\begin{aligned}
\frac{\partial^2(T-T_\infty)}{\partial y^2} &= \frac{\partial}{\partial y}\left(\frac{q_w}{k}H'(\eta)\right) \\
&= \frac{q_w}{k}H''(\eta)\cdot\frac{\partial\eta}{\partial y} \Rightarrow \frac{\partial^2(T-T_\infty)}{\partial y^2} = \frac{q_w}{k}\xi x^{-1/5}H''(\eta)
\end{aligned}$$

The energy equation is given as

$$\begin{aligned}
u\frac{\partial T}{\partial x} + v\frac{\partial T}{\partial y} &= \alpha\frac{\partial^2 T}{\partial y^2} \\
u\frac{\partial(T-T_\infty)}{\partial x} + v\frac{\partial(T-T_\infty)}{\partial y} &= \alpha\frac{\partial^2(T-T_\infty)}{\partial y^2}
\end{aligned}$$

Evaluating the first term on the LHS, we substitute in our values and simplify

$$\begin{aligned}
u\frac{\partial(T-T_\infty)}{\partial x} &= 5\nu\xi^2 x^{3/5}F'(\eta)\cdot\frac{q_w}{k}\xi^{-1}\frac{x^{-4/5}}{5}[H(\eta)-\eta H'(\eta)] \\
&= \frac{q_w}{k}\nu\xi x^{-1/5}F'(\eta)[H(\eta)-\eta H'(\eta)]
\end{aligned}$$

for the second term on the LHS, we get

$$\begin{aligned}
v\frac{\partial(T-T_\infty)}{\partial y} &= -\cancel{5}\nu\xi x^{-1/5}\left[\frac{4}{\cancel{5}}F(\eta) - \frac{1}{\cancel{5}}\eta F'(\eta)\right]\cdot\frac{q_w}{k}H'(\eta) \\
&= -\frac{q_w}{k}\nu\xi x^{-1/5}[4F(\eta)-\eta F'(\eta)]H'(\eta)
\end{aligned}$$

Combining both yields

$$\begin{aligned} u\frac{\partial(T-T_\infty)}{\partial x}+v\frac{\partial(T-T_\infty)}{\partial y} &= \frac{q_w}{k}\nu\xi x^{-1/5}\left(F'(\eta)[H(\eta)-\eta H'(\eta)]-[4F(\eta)-\eta F'(\eta)]H'(\eta)\right) \\ &= \frac{q_w}{k}\nu\xi x^{-1/5}(F'(\eta)H(\eta)-\cancel{F'(\eta)\eta H'(\eta)}-4F(\eta)H'(\eta)+\cancel{\eta F'(\eta)H'(\eta)}) \\ &= \frac{q_w}{k}\nu\xi x^{-1/5}(F'(\eta)H(\eta)-4F(\eta)H'(\eta)) \end{aligned}$$

The RHS is simplified by plugging in the second derivative as follows

$$\alpha\frac{\partial^2(T-T_\infty)}{\partial y^2}=\alpha\frac{q_w}{k}\xi x^{-1/5}H''(\eta)$$

Fully writing out the energy equation, notice both sides have a common factor so we simply cancel them out

$$\begin{aligned} u\frac{\partial(T-T_\infty)}{\partial x}+v\frac{\partial(T-T_\infty)}{\partial y} &= \alpha\frac{\partial^2(T-T_\infty)}{\partial y^2} \\ \frac{q_w}{k}\nu\xi x^{-1/5}(F'(\eta)H(\eta)-4F(\eta)H'(\eta)) &= \alpha\frac{q_w}{k}\xi x^{-1/5}H''(\eta) \\ v[F'(\eta)H(\eta)-4F(\eta)H'(\eta)] &= \alpha H''(\eta) \end{aligned}$$

Dividing both sides by α, and recalling that the Prandtl number is defined as $\Pr \triangleq \frac{\nu}{\alpha}$, we obtain

$$H''(\eta)+\Pr[4F(\eta)H'(\eta)-F'(\eta)H(\eta)]=0$$

□

Again, using the similarity variables, we show that we get the same energy equation as in **Eq. 12**. Next, we need to show that the boundary conditions hold (**Eq. 13a-b**).

Task II.i Boundary Conditions: Firstly, let's deal with the boundary conditions on $F(\eta)$

Proof. Given that $u=5\nu\xi^2x^{3/5}F'(\eta)$. If we are at the wall, that is, $y=0\Rightarrow\eta=0$ since $\eta=\xi yx^{1/5}$. So the only way that $u(0)=0$ is if $F'(0)=0$.

The other boundary condition is $v=0$ at $y=0$. Given that $v=-5\nu\xi x^{-1/5}[\frac{4}{5}F(\eta)-\frac{\eta}{5}F'(\eta)]$. Like above, at the wall $y=0\Rightarrow\eta=0$ so $v(0)=0$. The only way that this happens if if $F(0)=0$.

For the far-field conditions, as $y\to\infty,\eta\to\infty$ since $\eta=\xi yx^{-1/5}$. Also given $u=5\nu\xi^2x^{3/5}F'(\eta)$, u must vanish if $\eta\to\infty$ so we must have that $F'(\infty)=0$.

For the H boundary conditions, given that $T-T_\infty=\frac{q_w}{k}x^{1/5}\xi^{-1}H(\eta)$ and that at the wall, $y=0$, the uniform heat flux BC is

$$k\frac{\partial T}{\partial y}\Big|_{y=0}=-q_w\Rightarrow\frac{\partial T}{\partial y}\Big|_{y=0}=-\frac{q_w}{k}$$

Taking the derivative of $T-T_\infty$ w.r.t. y yields

$$\frac{\partial(T-T_\infty)}{\partial y}=\frac{q_w}{k}x^{1/5}\xi^{-1}H'(\eta)\cdot\frac{\partial\nu}{\partial y}=\frac{q_w}{k}H'(\eta)$$

so now, we evaluate it at $y=0$

$$\frac{\partial(T-T_\infty)}{\partial y}\Big|_{y=0}=\frac{q_w}{k}H'(0)$$

but since $\frac{\partial T}{\partial y}\Big|_{y=0} = -\frac{q_w}{k}$, since T_∞ is constant, then we equate both of them

$$\frac{q_w}{k}H'(0) = -\frac{q_w}{k} \Rightarrow H'(0) = -1$$

For the far-field condition for H, given that as $y \to \infty$, $T \to T_\infty \Rightarrow T - T_\infty \to 0$. Taking the partial of $T - T_\infty$. Given that $T - T_\infty = \frac{q_w}{k}x^{1/5}\xi^{-1}H(\eta)$, the only way for $T - T_\infty$ to approach 0 holding x fixed is for $H(\eta) \to 0$ as $\eta \to \infty$ so we must have that $H(\infty) = 0$. □

To determine the constants $C_{2.1}$ and $C_{2.2}$, we do the following

Task II.i (Part 2): Determination of numerical constants is done as follows

Proof. Simply by inspection, we take the u-momentum balance from above and we compare it against **Eq. 11**.

$$F''' - 3F'^2 + 4FF'' + H = 0 \Leftrightarrow F''' - C_{2.1}F'^2 + C_{2.2}FF'' + H = 0$$

From this it is clear that

$$C_{2.1} = 3 \quad \text{and} \quad C_{2.2} = 4$$

□

1.2.3 Integral Analysis

When we construct an x-direction force-momentum balance on a segment of the boundary layer, we obtained the relation

$$\mu\left[\frac{\partial u}{\partial y}\right]_{y=0}\Delta x = -\rho\frac{d}{dy}\int_0^\delta u^2 dy\Delta x + \rho\frac{d}{dy}\int_0^\delta uU_\infty dy\Delta x \tag{14}$$

where Δx is the length of the segment in the x direction, and the control volume has unit width in the direction normal to the paper. For the natural convection boundary layer considered here, this relation changes in two ways. We add the effect of the buoyancy force by integrating its effect per unit volume $\rho g\beta(T - T_\infty)$ over the volume of the control volume. In addition, there is zero free stream velocity in the natural convection flow considered here so we set U_∞ to zero. With these changes, the momentum balance equation becomes

$$\mu\left[\frac{\partial u}{\partial y}\right]_{y=0}\Delta x = -\rho\frac{d}{dy}\int_0^\delta u^2 dy\Delta x + \rho\frac{d}{dy}\int_0^\delta \rho g\beta(T - T_\infty) dy\Delta x \tag{15}$$

Dividing by Δx, Eq. (15) simplifies to:

$$\frac{d}{dx}\int_0^\delta u^2 dy = -\nu\left[\frac{\partial u}{\partial y}\right]_{y=0} + \int_0^\delta g\beta(T - T_\infty) dy \tag{16}$$

The integral form of the energy equation derived in lecture for forced convection also applies to the natural convection boundary layer flow here. Since the Prandtl number of interest is close to one, we assume here that $\delta_t = \delta$, and we therefore write the integral form of the energy equation derived in lecture as

$$\frac{d}{dx}\int_0^\delta u(T_\infty - T) dy = \alpha\left[\frac{\partial T}{\partial y}\right]_{y=0} \tag{17}$$

A solution can be generated using equation (16) and (17) if temperature and velocity profiles are postulated. Here we assume the following functional forms:

$$u = \hat{U}\frac{y}{\delta}\left(1 - \frac{y}{\delta}\right)^2, \quad \text{where } \hat{U} = Ax^m \tag{18a}$$

$$T - T_\infty = \Delta T \left(1 - \frac{y}{\delta}\right)^2 \tag{18b}$$

$$\delta = Bx^n \tag{18c}$$

Note that the heat flux at the wall must satisfy

$$q_w = -k \left[\frac{\partial T}{\partial y}\right]_{y=0} \tag{19a}$$

Differentiating the temperature relation (18b) and substituting into (19a) yields

$$q_w = \frac{2\Delta T k}{\delta} \tag{19b}$$

Equation (19b) implies that if q_w is constant, ΔT must have the same x dependence as δ. We therefore take

$$\Delta T = \frac{q_w B x^n}{2k} \tag{20}$$

which is the relation obtained when solving (19b) for ΔT and substituting the δ relation (18c).

Task III.i Reformulated Balance Equations: If we substitute relations (18) and (20) into integral equations (16) and (17), we can show that the result is two equations that can be cast in the forms

$$\frac{1}{C_{3.1}}\frac{d}{dx}(\hat{U}^2\delta) = \frac{1}{3}g\beta\Delta T\delta - \frac{\nu\hat{U}}{\delta} \quad \text{and} \quad \frac{1}{C_{3.2}}\frac{d}{dx}(\hat{U}\Delta T\delta) = \frac{2\alpha\Delta T}{\delta}$$

where $C_{3.1}$ and $C_{3.2}$ are integer numerical constants. The proof is as follows.

Proof. Starting by substituting $u = \hat{U}\frac{y}{\delta}\left(1 - \frac{y}{\delta}\right)^2$, where $\hat{U} = Ax^m$, into the integral equation (**Eq. 16**) we get

$$\frac{d}{dx}\int_0^\delta u^2 dy = -\nu\left[\frac{\partial u}{\partial y}\right]_{y=0} + \int_0^\delta g\beta(T - T_\infty)dy$$

We start by evaluating the integral on the LHS and letting $\kappa \triangleq \frac{y}{\delta} \Rightarrow y = \delta\kappa$

$$\begin{aligned}\int_0^\delta \left[\hat{U}\frac{y}{\delta}\left(1 - \frac{y}{\delta}\right)^2\right]^2 dy &= \hat{U}^2\int_0^\delta \frac{y^2}{\delta^2}\left(1 - \frac{y}{\delta}\right)^4 dy \\ &= \hat{U}^2\int_0^1 \frac{(\delta\kappa)^2}{\delta^2}(1-\kappa)^4\delta d\kappa \\ &= \hat{U}^2\delta\int_0^1 \kappa^2(1-\kappa)^4 d\kappa\end{aligned}$$

We can evaluate this integral by expanding it but that would be tedious. Instead, notice that it is in the form of the Euler integral of the first kind or the Beta function

$$B(z_1, z_2) \triangleq \int_0^1 t^{z_1-1}(1-t)^{z_2-1}dt \equiv \frac{\Gamma(z_1)\Gamma(z_2)}{\Gamma(z_1+z_2)}$$

So, our integral can be rewritten as follows

$$\begin{aligned}\int_0^\delta \left[\hat{U}^2 \frac{y}{\delta}\left(1-\frac{y}{\delta}\right)^2\right]^2 dy &= \hat{U}^2\delta B(3,5)\\ &= \hat{U}^2\delta\frac{\Gamma(3)\Gamma(5)}{\Gamma(8)}\\ &= \hat{U}^2\delta\frac{(3-1)!(5-1)!}{(8-1)!}\\ &= \hat{U}^2\delta\cdot\frac{48}{5040} = \frac{1}{105}\hat{U}^2\delta\end{aligned}$$

Then, by differentiating w.r.t. x, we get

$$\frac{d}{dx}\int_0^\delta u^2 dy = \frac{1}{105}\frac{d}{dx}(\hat{U}^2\delta)$$

Then, moving to the RHS, we start with the last term and differentiate u w.r.t. y using the product rule

$$\begin{aligned}-\nu\left.\frac{du}{dy}\right|_{y=0} &= -\nu\frac{d}{dy}\left[\hat{U}\frac{y}{\delta}\left(1-\frac{y}{\delta}\right)^2\right]\Bigg|_{y=0}\\ &= \nu\hat{U}\frac{d}{dy}\left[\frac{y}{\delta}\left(1-\frac{y}{\delta}\right)^2\right]\Bigg|_{y=0}\\ &= \nu\hat{U}\left[\frac{1}{8}\left(1-\frac{y}{\delta}\right)^2 - \frac{2y}{\delta^2}\left(1-\frac{y}{\delta}\right)\right]\Bigg|_{y=0} \Rightarrow -\nu\left[\frac{\partial u}{\partial y}\right]_{y=0} = -\nu\frac{\hat{U}}{\delta}\end{aligned}$$

The second term on the RHS is the buoyancy term. We substitute in the temperature profile (**Eq. 18b**) into the buoyancy term to get

$$\int_0^\delta g\beta(T-T_\infty)dy = g\beta\Delta T\int_0^\delta\left(1-\frac{y}{\delta}\right)^2 dy$$

Again utilizing our non-dimensional variable $\kappa \triangleq \frac{y}{\delta}$

$$\begin{aligned}g\beta\Delta T\int_0^\delta\left(1-\frac{y}{\delta}\right)^2 dy &= g\beta\Delta T\int_0^1(1-\kappa)^2\delta d\kappa\\ &= g\beta\Delta T\delta\int_0^1(1-\kappa)^2 d\kappa\\ &= g\beta\Delta T\delta\left[\kappa-\kappa^2+\frac{1}{3}\kappa^3\right]_0^1 \Rightarrow \int_0^\delta g\beta(T-T_\infty)dy = \frac{1}{3}g\beta\Delta T\delta\end{aligned}$$

Substituting all these expressions into the integral equation yields

$$\begin{aligned}\frac{d}{dx}\int_0^\delta u^2 dy &= -\nu\left[\frac{du}{dy}\right]_{y=0} + \int_0^\delta g\beta(T-T_\infty)dy\\ \frac{1}{105}\frac{d}{dx}(\hat{U}^2\delta) &= -\nu\frac{\hat{U}}{\delta} + \frac{1}{3}g\beta\Delta T\delta\\ \frac{1}{105}\frac{d}{dx}(\hat{U}^2\delta) &= \frac{1}{3}g\beta\Delta T\delta - \frac{\nu\hat{U}}{\delta}\end{aligned}$$

which is exactly the form shown above with $C_{3.1} = 105$. For the second form, we start by substituting $T - T_\infty = \Delta T \left(1 - \frac{y}{\delta}\right)^2 \Rightarrow T_\infty - T = -\Delta T \left(1 - \frac{y}{\delta}\right)^2$ into the integral equation (**Eq. 17**)

$$\frac{d}{dx}\int_0^\delta u(T_\infty - T)dy = \alpha \left[\frac{\partial T}{\partial y}\right]_{y=0}$$

Evaluating the integral on the LHS and letting $\kappa \triangleq \frac{y}{\delta}$ again

$$\begin{aligned}\int_0^\delta u(T_\infty - T)dy &= -\Delta T\hat{U}\int_0^\delta \frac{y}{\delta}\left(1-\frac{y}{\delta}\right)^2\left(1-\frac{y}{\delta}\right)^2 dy \\ &= -\Delta T\hat{U}\int_0^\delta \frac{y}{\delta}\left(1-\frac{y}{\delta}\right)^4 dy \\ &= -\Delta T\hat{U}\int_0^1 \kappa(1-\kappa)^4\delta d\kappa\end{aligned}$$

Again, instead of expanding, it is also in the form of the Beta function so

$$\begin{aligned}\int_0^\delta u(T_\infty - T)dy &= -\Delta T\hat{U}\delta B(2,5) \\ &= -\Delta T\hat{U}\delta\frac{\Gamma(2)\Gamma(5)}{\Gamma(7)} \\ &= -\Delta T\hat{U}\delta\frac{(2-1)!(5-1)!}{(7-1)!} \\ &= -\Delta T\hat{U}\delta\cdot\frac{24}{720} = -\frac{1}{30}\Delta T\hat{U}\delta\end{aligned}$$

Then differentiating w.r.t. x, we get

$$\frac{d}{dx}\int_0^\delta u(T_\infty - T)dy = -\frac{1}{30}\frac{d}{dx}(\hat{U}\Delta T\delta)$$

Then, evaluating the RHS, we have

$$\begin{aligned}\alpha\left[\frac{\partial T}{\partial y}\right]_{y=0} &= \alpha\left[\frac{\partial}{\partial y}\left(T_\infty + \Delta T\left(1-\frac{y}{\delta}\right)^2\right)\right]_{y=0} \\ &= 2\alpha\Delta T\left(1-\frac{y}{\delta}\right)\left(-\frac{1}{\delta}\right)\Bigg|_{y=0} \\ &= -\frac{2\alpha\Delta T}{\delta}\left(1-\frac{y}{\delta}\right)\Bigg|_{y=0} \Rightarrow \alpha\left[\frac{\partial T}{\partial y}\right]_{y=0} = -\frac{-2\alpha\Delta T}{\delta}\end{aligned}$$

Finally, substituting all these expressions into the integral formula we end up with

$$\begin{aligned}\frac{d}{dx}\int_0^\delta u(T_\infty - T)dy &= \alpha\left[\frac{\partial T}{\partial y}\right]_{y=0} \\ \frac{1}{30}\frac{d}{dx}(\hat{U}\Delta T\delta) &= \frac{2\alpha\Delta T}{\delta}\end{aligned}$$

Which is exactly in the form shown above with $C_{3.2} = 30$. □

Now that we have shown that we can rewrite the momentum and energy balances in the reformulated form, we need to find the exponents m and n. The work is shown below.

Task III.i: Exponent Determination

Proof. To find the exponents m and n, we use the following relations: $\hat{U} = Ax^m$, $\Delta T = \frac{q_w B x^n}{2k}$, and $\delta = Bx^n$. Substituting them into into the reformulated integral relation derived above yields

$$\begin{aligned}
\frac{1}{105}\frac{d}{dx}(\hat{U}^2\delta) &= \frac{1}{3}g\beta\Delta T\delta - \nu\frac{\hat{U}}{\delta} \\
\frac{1}{105}\frac{d}{dx}((Ax^m)^2(Bx^n)) &= \frac{1}{3}g\beta\left(\frac{q_w B x^n}{2k}\right)(Bx^n) - \nu\frac{(Ax^m)}{(Bx^n)} \\
\frac{1}{105}\frac{d}{dx}(A^2Bx^{2m+n}) &= \frac{g\beta q_w B^2}{6k}x^{2n} - \frac{\nu A}{B}x^{m-n} \\
\frac{A^2B}{105}(2m+n)x^{2m+n-1} &= \frac{g\beta q_w B^2}{6k}x^{2n} - \frac{\nu A}{B}x^{m-n}
\end{aligned}$$

Next, we look at the energy relation

$$\begin{aligned}
\frac{1}{30}\frac{d}{dx}(\hat{U}\Delta T\delta) &= \frac{2\alpha\Delta T}{\delta} \\
\frac{1}{30}\frac{d}{dx}\left((Ax^m)\left(\frac{q_w B x^n}{2k}\right)(Bx^n)\right) &= \frac{2\alpha}{Bx^n}\left(\frac{q_w B x^n}{2k}\right) \\
\frac{1}{30}\frac{d}{dx}\left(AB^2\frac{q_w}{2k}x^{m+2n}\right) &= \frac{q_w\alpha}{k} \\
\frac{1}{30}AB^2\frac{q_w}{2k}(m+2n)x^{m+2n-1} &= \frac{q_w\alpha}{k}
\end{aligned}$$

Since there is no x term on the RHS, the only way this equation holds $\forall x$ is if $x^{m+2n-1} = 0 \Rightarrow m + 2n - 1 = 0 \Rightarrow m + 2n = 1$. Going back to the momentum equation, since the x dependence of every term in each of the equations are the same, then all the terms must share the same power of x, so we have the following

$$2m + n - 1 = 2n \Rightarrow n = 2m - 1 \quad \text{and} \quad 2m + n - 1 = m - n$$

For the first equality, if we manipulate it, we get

$$2m + n - 1 = 2m + (2m - 1) - 1 \Rightarrow 2m + n - 1 = 4m - 2$$

We also know that the exponent on the RHS are also equal yielding us

$$2n = m - n \Rightarrow m = 3n$$

Substituting that into our exponent relation from the energy equation we get

$$m + 2n = 1 \Rightarrow 5n = 1 \Rightarrow n = \frac{1}{5}$$

So substitute $n = \frac{1}{5}$ into $m = 3n$ to get

$$m = 3n \Rightarrow m = 3\left(\frac{1}{5}\right) \Rightarrow m = \frac{3}{5}$$

Therefore, the exponents are $m = 0.6$ and $n = 0.2$. □

As m and n are determined above, we can simply substitute their values into the reformulated integral equations and solve them for the constants A and B in terms of the other system parameters.

Task III.i: Determination of Unknown Constants

Proof. We take the exponents we got from above—$m = 0.6$ and $n = 0.2$—and substitute them into our reformulated energy balance yielding

$$\begin{aligned}
\frac{1}{30}AB^2\frac{q_w}{2k}(m+2n)x^{m+2n-1} &= \frac{q_w\alpha}{k} \\
\frac{1}{30}AB^2\frac{q_w}{2k}(0.6+2(0.2))x^{0.6+2(0.2)-1} &= \frac{q_w\alpha}{k} \\
\frac{1}{30}AB^2\frac{q_w}{2k} &= \frac{q_w\alpha}{k} \\
\frac{1}{60}AB^2 = \alpha \Rightarrow A &= \frac{60\alpha}{B^2}
\end{aligned}$$

Now we can substitute the exponents into the reformulated momentum balance

$$\begin{aligned}
\frac{A^2B}{105}(2m+n)x^{2m+n-1} &= \frac{g\beta q_w B^2}{6k}x^{2n} - \frac{\nu A}{B}x^{m-n} \\
\frac{A^2B}{105}(2(0.6)+0.2)x^{2(0.6)+0.2-1} &= \frac{g\beta q_w B^2}{6k}x^{2(0.2)} - \frac{\nu A}{B}x^{0.6-0.2} \\
\frac{A^2B}{105}(1.4)\cancel{x^{0.4}} &= \frac{g\beta q_w B^2}{6k}\cancel{x^{0.4}} - \frac{\nu A}{B}\cancel{x^{0.4}} \\
\frac{1}{75}A^2B &= \frac{g\beta q_w B^2}{6k} - \frac{\nu A}{B} \\
\frac{1}{75}A^2B + \frac{\nu A}{B} - \frac{g\beta q_w B^2}{6k} &= 0
\end{aligned}$$

Substituting in A from the energy balance, we get

$$\begin{aligned}
\frac{1}{75}\left(\frac{60\alpha}{B^2}\right)^2 B + \frac{\nu}{B}\left(\frac{60\alpha}{B^2}\right) - \frac{g\beta q_w B^2}{6k} &= 0 \\
\frac{1}{75}\left(\frac{3600\alpha^2}{B^4}\right) B + \frac{60\nu\alpha}{B^3} - \frac{g\beta q_w B^2}{6k} &= 0 \\
\frac{48\alpha^2}{B^3} + \frac{60\nu\alpha}{B^3} - \frac{g\beta q_w B^2}{6k} &= 0 \\
48\alpha^2 + 60\nu\alpha - \frac{g\beta q_w B^5}{6k} &= 0 \\
g\beta q_w B^5 &= 6k(48\alpha^2 + 60\nu\alpha) \\
B &= \left(\frac{72\alpha k(4\alpha+5\nu)}{g\beta q_w}\right)^{1/5}
\end{aligned}$$

Substituting B into our expression for A we get

$$A = \frac{60\alpha}{B^2} \Rightarrow A = 60\alpha\left(\frac{g\beta q_w}{72\alpha k(4\alpha+5\nu)}\right)^{2/5}$$

□

From this, we are now able to write out the complete relations for the $u(x, y/\delta)$ and $T(x, y/\delta)$ fields and $\delta(x)$.

Task III.i: Complete Solution for the Fields and δ

Proof. We start with writing out the complete relation for the boundary layer thickness $\delta(x)$.

$$\delta(x) \triangleq Bx^n = \left(\frac{72\alpha k(4\alpha+5\nu)}{g\beta q_w}\right)^{1/5} x^{0.2}$$

Next, we write the complete relation for the velocity field $u(x, y/\delta)$

$$\begin{aligned} u(x,y/\delta) &= \hat{U}\frac{y}{\delta}\left(1-\frac{y}{\delta}\right)^2, \quad \text{where } \hat{U} = Ax^m \\ &= Ax^m\frac{y}{\delta}\left(1-\frac{y}{\delta}\right)^2 \Rightarrow u(x,y/\delta) = 60\alpha\left(\frac{g\beta q_w}{72\alpha k(4\alpha+5\nu)}\right)^{2/5} x^{0.6}\cdot\frac{y}{\delta}\left(1-\frac{y}{\delta}\right)^2 \end{aligned}$$

Finally, we write the complete relation for the temperature field $T(x, y/\delta)$

$$\begin{aligned} T - T_\infty &= \Delta T\left(1-\frac{y}{\delta}\right)^2 \quad \text{where } \Delta T = \frac{q_w B x^n}{2k} \\ T - T_\infty &= \frac{q_w}{2k}Bx^{0.2}\left(1-\frac{y}{\delta}\right)^2 \\ T(x,y/\delta) &= T_\infty + \frac{q_w}{2k}Bx^{0.2}\left(1-\frac{y}{\delta}\right)^2 \\ T(x,y/\delta) &= T_\infty + \frac{q_w}{2k}\left(\frac{72\alpha k(4\alpha+5\nu)}{g\beta q_w}\right)^{1/5} x^{0.2}\left(1-\frac{y}{\delta}\right)^2 \end{aligned}$$

□

2 Implementation and Development of Code

The following section will be a summary of our analysis organization including a description of our code organization for each task as well as different cases within each task that were considered. For each major block of the script, we include the relevant code snippet followed by an explanation that explains how that section fits into the overall analysis. A complete copy of our code for each question, respectively, can be found in Section 5 at the end of the report. Also note that running the code for different configurations and material properties are not included since the change is minimal.

2.1 Task I

Our analysis is organized into several main tasks. In this task, we implement a modular code design where we specify the flow conditions and other relevant parameters, discretize the domain and solve the governing laminar boundary layer equations with a finite-difference FTCS algorithm, apply boundary conditions, and post-process our data to visualize our results. Firstly, we start with the flow conditions and defining the geometric parameters.

```
%% Flow Conditions and Geometric Parameters
g = 9.81;            % gravity
beta = 3.3e-3;       % thermal expansion coeff.
nu = 1.5e-5;         % kinematic vis.
alpha = 2.2e-5;      % thermal diff.
k_air = 0.0261;      % thermal cond. of air
Tinf = 30;           % ambient temp.
q_w = 220;           % surface heat flux
```

The purpose of the initialization section was that it helped set up our simulation by defining all the necessary geometric and material parameters so we could keep track of all of them and make changes if necessary. Some of the physical properties include the gravitational acceleration constant, coefficient of thermal expansion,

viscosity, diffusivity, and conductivity, along with the specified ambient temperature and the constant heat flux on the vertical plate. All of these values are initialized for use in subsequent sections.

```
% Mesh Grid Dimensions
Nx = 51;              % points in vertical (x) direction
Ny = 51;              % points in horizontal (y) direction

% Grid spacings, simulation time, and tolerance
dx = 2.0e-3;          % vertical grid spacing
dy = 0.4e-3;          % horizontal grid spacing
dt = 5.0e-4;          % time step
maxTime = 100;        % time limit
Tol = 1.0e-6;         % convergence tolerance

% Initializing coordinate vectors
x = linspace(0, (Nx-1)*dx, Nx);
y = linspace(0, (Ny-1)*dy, Ny);
numSteps = ceil(maxTime/dt);
```

We focus on discretization and geometry in this section. `Nx` and `Ny` are the number of grid points in both the vertical x and horizontal y directions; the grid spacings are defined as `dx` and `dy`, respectively. We also define simulation parameters such as the desired time step `dt`, as well as the max time in which we run the method for; we change this in future parts to show the transient nature of the flow as it starts up. The convergence tolerance for steady-state determination is also listed here. Initially, we calculate the `x` and `y` coordinate vectors and those are initialized, computed, and assigned to provide the locations where we will compute the solution.

```
%% Array Initialization
u_old = zeros(Nx, Ny);          % u: vertical velocity component
v_old = zeros(Nx, Ny);          % v: horizontal velocity component
T_old = Tinf * ones(Nx, Ny);    % temperature field
u_new = u_old;
v_new = v_old;
T_new = T_old;

% Preallocate arrays
Tsurf_vals = zeros(1, numSteps);
time_vals  = zeros(1, numSteps);
u_inlet_vals = zeros(1, numSteps);
usurf_vals = zeros(1, numSteps);
```

Pre-allocating memory is important for any iterative scheme in order to reduce computation time in case we choose to run it for a long time. Thus, we reserve space for the solution arrays for the components of the velocity field as well as temperature field. We initialize the temperature field to be uniform at the ambient temperature and create arrays to store the old (unprimed) and new (primed) field variables at each node.

Additionally, we have two arrays `Tsurf_vals` and `u_inlet_vals` that will store the surface temperature and inlet velocity as time progresses.

```
%% Time Loop for FTCS
for n = 1:numSteps

    % Ensure local scope
    u = u_old;
    v = v_old;
    T = T_old;

    % Update temperature in interior
    for i = (Nx-1):-1:2
        for j = 2:(Ny-1)

            % Upwind differencing in x
            if u(i,j) >= 0
                dTdx = (T(i,j) - T(i-1,j)) / dx;
            else
```

```
                dTdx = (T(i+1,j) - T(i,j)) / dx;
            end

            % Upwind differencing in y
            if v(i,j) >= 0
                dTdy = (T(i,j) - T(i,j-1)) / dy;
            else
                dTdy = (T(i,j+1) - T(i,j)) / dy;
            end
            convT = - ( u(i,j)*dTdx + v(i,j)*dTdy );

            % Diffusion term
            diffT_x = alpha * ( T(i+1,j) - 2*T(i,j) + T(i-1,j) ) / (dx^2);
            diffT_y = alpha * ( T(i,j+1) - 2*T(i,j) + T(i,j-1) ) / (dy^2);

            % New temperature
            T_new(i,j) = T(i,j) + dt*( convT + diffT_x + diffT_y );
        end
    end

    % Update vertical velocity in interior
    for i = (Nx-1):-1:2
        for j = 2:(Ny-1)

            % Upwind differencing for u in x
            if u(i,j) >= 0
                dudx = (u(i,j) - u(i-1,j)) / dx;
            else
                dudx = (u(i+1,j) - u(i,j)) / dx;
            end

            % Upwind differencing in y
            if v(i,j) >= 0
                dudy = (u(i,j) - u(i,j-1)) / dy;
            else
                dudy = (u(i,j+1) - u(i,j)) / dy;
            end

            convU = - ( u(i,j)*dudx + v(i,j)*dudy );

            % Diffusion term
            diffU_x = nu * ( u(i+1,j) - 2*u(i,j) + u(i-1,j) ) / (dx^2);
            diffU_y = nu * ( u(i,j+1) - 2*u(i,j) + u(i,j-1) ) / (dy^2);

            % Bouyancy term
            buoy = g * beta * ( T(i,j) - Tinf );

            % New x-direction velocity
            u_new(i,j) = u(i,j) + dt*( convU + diffU_x + diffU_y + buoy );
        end
    end

    % Update horizontal velocity v from continuity
    for i = (Nx-1):-1:2
        for j = 2:(Ny-1)
            du_dx = (u(i,j) - u(i-1,j)) / dx;
            dv_dy = (v(i,j) - v(i,j-1)) / dy;
            v_new(i,j) = v(i,j) - dt*( du_dx + dv_dy );
        end
    end
```

This block of code above is our explicit Forward-Time-Central Space (FTCS) algorithm which is used to advance the temperature and velocity fields in time. We loop over all the time steps within `maxTime` For each iteration, we first assign the previous iterations' values to current local variables just for clarity. We then compute the fields at the next time step at all non-boundary points using a top-down approach hence why our step in the for loop is -1. Following from the introduction, we use upwind differencing for the convective terms, and second-order central differences for the diffusion term. The velocity undergoes the exact same

process but we introduce the buoyancy term invoking the Boussinesq approximation for the x velocity and we enforce continuity for the y velocity.

Next, we focus on defining and applying boundary conditions. They come into effect on the edge of the domain. The left boundary is the constant heat flux on the vertical plate and is enforced using a FDM given by

$$T'_{i,1} = T'_{i,2} + \Delta y \frac{q_w}{k}$$

```
%% BOUNDARY CONDITIONS

% Left wall (constant heat flux)
for i = 1:Nx
    u_new(i,1) = 0;
    v_new(i,1) = 0;
    % Apply constant heat flux in the y-direction:
    T_new(i,1) = T_new(i,2) + (q_w * dy / k_air);
end

% Right boundary = zero-gradient and far field
for i = 1:Nx
    T_new(i,Ny) = T_new(i,Ny-1);
    u_new(i,Ny) = 0;
    v_new(i,Ny) = v_new(i,Ny-1);
end

% Bottom boundary = zero-gradient and T=Tinf
for j = 1:Ny
    T_new(1,j) = Tinf;
    u_new(1,j) = u_new(2,j);
    v_new(1,j) = v_new(2,j);
end

% Top boundary = zero-gradient
for j = 1:Ny
    T_new(Nx,j) = T_new(Nx-1,j);
    u_new(Nx,j) = u_new(Nx-1,j);
    v_new(Nx,j) = v_new(Nx-1,j);
end
```

We apply Neumann type boundary conditions to almost all other edges since there is no gradient. However, on the right most boundary we have our u velocity boundary condition set to 0. We use a no-slip condition in the u and v directions since it there can be no velocity on the wall. In the same section, we also extract all the necessary data and check the convergence of the solution.

```
% Top edge temperature
Tsurf_vals(n) = T_new(Nx,1);

% Bottom edge velocity
u_inlet_vals(n) = u_new(Nx,2);

% Store the current time
time_vals(n)  = (n-1)*dt;

% Check for convergence
diffU = max(abs(u_new(:) - u_old(:)));
diffV = max(abs(v_new(:) - v_old(:)));
diffT = max(abs(T_new(:) - T_old(:)));
maxChange = max([diffU, diffV, diffT]);

% Update fields for the next iteration
u_old = u_new;
v_old = v_new;
T_old = T_new;

% Check if steady-state is reached
```

```
    if maxChange < Tol
        fprintf('Steady-state reached at iteration %d, time ~ %.4f s\n', n, time_vals(n));
        break;
    end
end
```

We store the top edge temperature, inlet velocity at the bottom edge and keep track of their evolution over time. We also take the max ℓ_1 norm of u, v and T so we can determine whether or not steady state has been reached if the maximum absolute differences fall below a pre-defined tolerance threshold.

Lastly, we post-process our data and generate visualizations for our results. We have a plot showing how the surface temperature of the plate evolves over time, contour plots of the temperature and velocity fields, the v component of the velocity as well as several different values of x for which velocities are calculated. Other miscellaneous pieces of information are also calculated but only displayed in the console such as when the leading-edge effect is reached and steady-state flow information. Since this part is mostly self-explanatory, we include the code in the Appendix.

2.2 Task II

Our original set of partial differential equations **Eq. 1-3** were reduced to two ordinary differential equations where the u-momentum equation is third order in F, the energy equation is second order in H, and this system of ODEs combined yields us a fifth order system of 1st order ODEs. As always, we first set up the physical parameters and flow conditions as follows and boundary conditions.

```
%% Flow Conditions and Geometric Parameters
Pr = 0.733;          % Prandtl number
g = 9.81;            % gravity
beta = 0.0033;       % thermal expansion coeff,
nu = 1.5e-5;         % kinematic vis.
k_air = 0.026;       % thermal cond. of air
T_inf = 30;          % ambient temp.

% Boundary Conditions
F0 = 0; Fp0 = 0; Hp0 = -1; etaMax = 10;
```

The `F0` and `Fp0` are the stream functions and its respective first derivative. `Hp0` is the temperature gradient at the wall and `etaMax` is the upper limit for integration. We then utilize `fsolve` to determine the unknown initial conditions by using the literature as our initial guess.

```
% Guess and find the correct F''(0) and H(0)
guess = [0.80893; 1.47981];
options = optimset('TolFun',1e-7,'TolX',1e-7);
sol = fsolve(@(soln) res(soln,Pr,Hp0,etaMax), guess, options);
Fpp0_sol = sol(1); H0_sol = sol(2);
```

Next, we integrate the ODE via the RK4 scheme. We set up the grid and pre-allocate arrays with our integration results. We also use a function handle to define the system of ODEs containing our state variables and apply the RK4 algorithm in a loop where we calculate the intermediate slopes k_1 to k_4 within each step.

```
%% 4th Order Runge-Kutta Method
Fpp0 = Fpp0_sol; H0 = H0_sol;

% Calculated constants C2.1 and C2.2
C21 = 3; C22 = 4;

N = 1000; dEta = etaMax/N;

% Arrays for storing solutions
etaVals = zeros(N+1,1);
yF = zeros(N+1,1);
yFp = zeros(N+1,1);
yFpp = zeros(N+1,1);
```

```
yH = zeros(N+1,1);
yHp = zeros(N+1,1);

% Initial values at eta=0
yFpp(1) = Fpp0; yH(1) = H0; yHp(1) = Hp0;

% Define ODE system used in final integration
% F' (dF/deta); F''(dF'/deta); F''' (dF''/deta); H'(dH/deta); H''(dH'/deta)
sys = @(eta, Y) [
    Y(2);
    Y(3);
    C21*(Y(2)^2)-C22*Y(1)*Y(3) - Y(4);
    Y(5);
    Pr*(Y(2)*Y(4) - 4*Y(1)*Y(5))
];

% Runge-Kutta Scheme
for i = 1:N
    eta_i = etaVals(i);
    Yi    = [yF(i); yFp(i); yFpp(i); yH(i); yHp(i)];

    % Advance solution one step from eta to deta
    k1 = sys(eta_i, Yi);
    k2 = sys(eta_i + 0.5*dEta, Yi + 0.5*dEta*k1);
    k3 = sys(eta_i + 0.5*dEta, Yi + 0.5*dEta*k2);
    k4 = sys(eta_i + dEta, Yi + dEta*k3);

    % For each new i=1 to 5, yi at the new eta location is:
    Y_new = Yi + (dEta/6)*(k1 + 2*k2 + 2*k3 + k4);

    etaVals(i+1) = eta_i + dEta;
    yF(i+1) = Y_new(1);
    yFp(i+1) = Y_new(2);
    yFpp(i+1) = Y_new(3);
    yH(i+1) = Y_new(4);
    yHp(i+1) = Y_new(5);
end
```

Next, we plot out the similarity solutions $F'(\eta)$ and $H(\eta)$. We use our **Task I** files and we load the temperature and velocity results for $q_w = 220\text{W/m}^2$ and $q_w = 150\text{W/m}^2$ and convert the spatial coordinates to the η-space and overlay them on top of the similarity plots. We transform them using the similarity variables defined in Section 1.2.2.

```
    % Loop over the chosen x-coordinates
    for selected_x = [30, 60, 100]
        % Find index in x_array that is closest to selected x
        [~, iX] = min(abs(x_array_mm - selected_x));
        xVal_m = x_array(iX);
        xi = ((g * beta * qW) / (5 * k_air * nu^2))^(1/5);
        eta_fd = zeros(size(y_array));
        Fp_fd = zeros(size(y_array));
        H_fd = zeros(size(y_array));

        for j = 1:length(y_array)
            yVal_m = y_array(j);
            eta_fd(j) = (yVal_m / xVal_m^(0.2)) * xi;
            uVal = Ufield(iX, j);
            Fp_fd(j) = (uVal / (5 * nu * xi^2 * xVal_m^(3/5)));
            TVal = Tfield(iX, j);
            H_fd(j) = (TVal - T_inf)/((qW/k_air)*xVal_m^(1/5)*(xi^(-1)));
        end

        subplot(1,2,1);
        plot(eta_fd, Fp_fd, 'x', 'DisplayName', sprintf('$q=$%d, $x=$%.0f$mm$', qW,
            selected_x));

        subplot(1,2,2);
```

```
        plot(eta_fd, H_fd, 'x', 'DisplayName', sprintf('$q=$%d, $x=$%.0f$mm$', qW,
            selected_x));
    end
end
```

For each heat fluxes $q_w = 150\text{W/m}^2$ and $q_w = 220\text{W/m}^2$, we extracts the corresponding values in the field variables. For the x-locations, η is calculated; the field profiles are transformed to non-dimensional forms and are overlayed with the RK4 results. We also plot the log-log scaled version of the temperature difference. Some helper functions include one to reformulate similarity equations into a state-space form along with a residual function that uses 10000 points to run a very fine RK4 scheme to compute the residuals at the end. All of this code can be seen in the Appendix.

2.3 Task III

This code was NOT NEEDED FOR THIS PROJECT, however we include it for completeness. Similar to all the other scripts, we start by defining the physical constants and flow conditions (same as **Task I**). In addition, the ambient temperature and surface heat flux are provided. The streamwise locations $x_1 = 0.03\text{m}$ and $x_2 = 0.06\text{m}$ are where we will be evaluating both the integral relations as well as the similarity solutions.

```
%% Flow Conditions from Task I and Parameters
g = 9.8;                        % gravity
beta = 0.0033;                  % thermal expansion coeff.
nu = 1.613e-5;                  % kinematic vis.
alpha = 2.2e-5;                 % thermal diff.
k = 0.0261;                     % thermal cond. of air
T_inf = 30;                     % ambient temp.
q_w = 220;                      % surface heat flux

% Stream wise locations
x1 = 0.03; x2 = 0.06;

%% Integral Solutions
% Exponents and constants for reformulated balance equations
n = 0.2; m = 0.6;
B = ((72*alpha*k*(4*alpha+5*nu))/(g*beta*q_w))^0.2;
A = 60*alpha*(((g*beta*q_w)/(72*alpha*k*(4*alpha+5*nu)))^0.4);

% Boundary Layer Thickness at specified x
delta1 = B*x1^n; delta2 = B*x2^n;

% Similarity variables and other constants
xi = ((g*beta*q_w)/(5*k*nu^2))^(1/5);
eta = @(y, x) xi * y ./ (x^(1/5));
```

We then calculate all of the parameters needed to calculate the integral solution for the boundary layer. The expressions can be found in Section 1.2.3. Integral solutions are then transcribed into MATLAB as function handles and calculated as follows

```
% Integral solution expressions for velocity and temperature at x's
u_int_03 = @(y) A*x1^m .* (y/delta1) .* (1 - (y/delta1)).^2;
u_int_06 = @(y) A*x2^m .* (y/delta2) .* (1 - (y/delta2)).^2;
T_int_03 = @(y) T_inf + (q_w/(2*k)) * B*x1^n .* (1 - (y/delta1)).^2;
T_int_06 = @(y) T_inf + (q_w/(2*k)) * B*x2^n .* (1 - (y/delta2)).^2;
```

Also recall that we defined our similarity variable as

$$\eta \equiv \xi \frac{y}{x^{1/5}}$$

This allows us to still write the profiles with the physical y but we just scale it externally afterwards. Then, we plot the integral solutions for both fields. As for the similarity variables, we have the following

```
%% Similarity Variables and Transformations

% Similarity transformation for velocity:  F'(eta)
```

```
F_transform = @(u, x) u ./ (5*nu*xi^2*(x^(0.6)));

% Similarity transformation for temperature: H(eta)
H_transform = @(T, x) (T - T_inf) ./ ((q_w/k) .* (x^(0.2)/xi));

% For x = 0.03 m
u_vals_03 = u_int_03(y_vals_03);
T_vals_03 = T_int_03(y_vals_03);
Fprime_int_03 = F_transform(u_vals_03, x1);
H_int_03 = H_transform(T_vals_03, x1);

% For x = 0.06 m
u_vals_06 = u_int_06(y_vals_06);
T_vals_06 = T_int_06(y_vals_06);
Fprime_int_06 = F_transform(u_vals_06, x2);
H_int_06 = H_transform(T_vals_06, x2);
```

We needed to derive an explicit x-dependence by turning velocity and temperature into their similarity forms defined in Section 1.2.2. We use the `F_transform` and `H_transform` to do this conversion. Again, we just plot $F'(\eta)$ and $H(\eta)$ at two points downstream. Lastly, we plot the wall temperature as we vary x. For the integral solution, we just used one of the expressions we got from previous parts and rearranged it to find T_{wall}. Note that since we are evaluating it at $y = 0 \Rightarrow \eta = 0$.

2.4 Task IV

2.4.1 Bisection Method for Free Convection

This code was also NOT NEEDED FOR THIS PROJECT, but we include it for completeness. The first part of code used in **Task IV. i** was written to simplify the trial and error technique to estimate the temperature difference between the surface of the electronic device and the ambient temperature (ΔT) illustrated in Incropera and Dewitt's *Principles of Heat and Mass Transfer*. The core of the model is based on a standard empirical correlation for laminar free convection along a vertical plate and is modeled after the Bisection method—a divide-and-conquer algorithm with a relatively optimized time complexity of $\mathcal{O}(\log_2(1/\text{TOL}))$.

$$\overline{\text{Nu}}_L = 0.59(\text{Gr}_L\text{Pr})^{1/4}$$

We start by defining all relevant flow condition, fluid, and heat properties, as well as the target heat flux and our acceptable stop tolerance. A function `q_pred(DeltaT)` is then defined to compute the predicted heat flux for any guessed temperature difference using the correlation. The bisection loop begins with an initial lower and upper guess for ΔT and repeatedly halves the interval until the predicted flux is within 0.01W/m^2 of the target difference. A fully implemented code can be found in the Appendix with a tolerance of 0.01.

At each iteration, we find midpoint, recalculate the heat flux, and updates the lower and upper bounds accordingly. Once the condition is met, the script outputs the converged ΔT. This approach ensures a self-consistent, and accurate prediction of temperature difference. A pseudo algorithm is found below

2.4.2 FTCS FD Approach with Radiative Heat Exchange

We will only explain the relevant new parts added to the code, namely the incorporation of a new mode of heat transfer into the boundary condition at the left plate. At the boundary, the finite difference solution must obey the following

$$-\frac{q_e}{k} = \frac{T_{i,2} - T_{i,1}}{\Delta y} - \frac{\sigma\epsilon_s}{k}(T_{i,1}^4 - T_\infty^4)$$

This section is added to account for radiative heat loss in addition to conduction. For the flow conditions and geometric parameters, the only ones we added were the Stefan-Boltzmann constant as well as the emissivity of the surface. We also include the maximum allowable surface temperature and we define query points that we want to test for different heat fluxes. We then loop through different surface generation q_w and use our **Task I** code. Below is the sections of code with changes from previous codes:

Algorithm 2 Bisection Method for ΔT in Laminar Free Convection

1. **Initialize:** Choose ΔT_{min} and ΔT_{max} such that the actual solution lies within $[\Delta T_{\text{min}}, \Delta T_{\text{max}}]$.
2. **Compute midpoint:**
$$\Delta T_{\text{mid}} \leftarrow \frac{\Delta T_{\text{min}} + \Delta T_{\text{max}}}{2}.$$
3. **Compute and assign predicted heat flux**:
$$q_{\text{pred}} \leftarrow \frac{k}{L} 0.59 \left(\text{Pr} \frac{g \beta (\Delta T_{\text{mid}}) L^3}{\nu^2} \right)^{\frac{1}{4}} \times \Delta T_{\text{mid}}.$$
4. **Compare with target:**
   - If $\left| q_{\text{pred}} - q_{\text{target}} \right| \leq$ TOL, then **stop**.
   - Otherwise, if $q_{\text{pred}} > q_{\text{target}}$, set $\Delta T_{\text{max}} \leftarrow \Delta T_{\text{mid}}$.
   - Else set $\Delta T_{\text{min}} \leftarrow \Delta T_{\text{mid}}$.
5. **Repeat:** Go back to step 2 until convergence or until the iteration limit is reached.

Output: The final midpoint, ΔT_{mid}, is your solution. Then $T_{\text{s}} = T_{\infty} + \Delta T_{\text{mid}}$.

```
% Heated left wall
        for i = 1:Nx
            u_new(i,1) = 0;
            v_new(i,1) = 0;
            % Combine conduction and radiation
            radiation = (sigma*emis/k_air)*((T_new(i,2)+273.15)^4 - (Tinf+273.15)^4);
            conduction = (q_w / k_air);
            T_new(i,1) = T_new(i,2) + dy*(conduction + radiation);
        end


    %radiation q calc
    q_rad_vals = zeros(Nx,1);
    for i = 1:Nx
        T_adj_K = T_new(i,2) + 273.15;
        % Change temp to Kelvin
        Tinf_K  = Tinf + 273.15;
        q_rad_vals(i) = emis * sigma * (T_adj_K^4 - Tinf_K^4);
        q_rad_avg = mean(q_rad_vals);
    end
```

We calculate the local radiative rejection along the wall using the adjacent node and the ambient temperature. For the lunar portion of our analysis, we simply reduce the gravitational acceleration by 1/6 and re-run the script. The total heat generation rate is then calculated by multiplying the conduction and radiation combined heat flux by the wall area to get the units of W.

3 Results and Discussion

3.1 Task I Results and Discussion

We were required to use 3 different heat fluxes $q_w = 220\text{W/m}^2$, $q_w = 300\text{W/m}^2$, and $q_w = 150\text{W/m}^2$ to study what would happen to a variety of things including temperature and velocity profiles. We monitored the time at which the leading edge effect was reached, the time at which the flow reached 99% of steady state, as well as the final 99% steady state wall temperature along with the time required to get there. Below, we

summarize our findings in **Table 1**.

q_w (W/m^2)	t_{LE} (s)	$t_{99\%}$ (s)	T_{peak} (°C)	$t_{T,\text{peak}}$ (s)	u_{peak} (m/s)	T_{SS} (°C)	u_{SS} (m/s)
220	0.1615	0.7860	73.50	0.9300	0.0614	70.99	0.0584
300	0.1215	0.6905	85.84	0.8185	0.0735	82.55	0.0735
150	0.2305	0.9225	61.96	1.088	0.0491	60.17	0.0468

Table 1: Transient Behavior Characteristics for Various Heat Fluxes q_w

For a heat flux $q_w = 220\text{W/m}^2$, we determine that the time for the leading-edge effect to reach the top edge of the uppermost component is 0.1615 seconds. The time for the flow to reach 99% of its steady state surface temperature is 0.7860 seconds. If the heat flux was changed to $q_w = 300\text{W/m}^2$, the leading edge time and 99 percent of steady state values are 0.1215 and 0.6675 seconds respectively. These times being quicker makes sense because the greater heat flux will heat the gas to a higher temperature, causing it to expand more, and thus increasing the velocity of the leading edge and overall speed of the system.

Going back to the original heat flux, if $q_w = 220\text{W/m}^2$, we can plot the variation of the temperature of the top edge as a function of time as seen below in **Figure 3**.

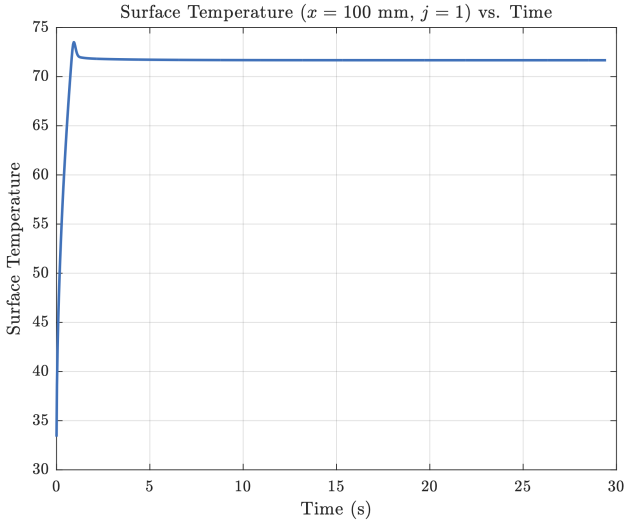

Figure 3: Variation of Surface Temperature at Top Edge for $q_w = 220\text{W/m}^2$

The temperature does not rise monotonically. Initially, the temperature rises sharply and overshoots, before eventually settling/converging on a value a few degrees below the peak temperature. It might be natural to wonder if an overshoot can physically happen or is it an artifact of the numerical model. The FTCS method initially overshoots because its way of approximating derivatives introduces small errors, causing temporary oscillations. Over time, numerical diffusion smooths out these oscillations allowing convergence.

The peak and steady state values of u-velocity and surface temperature at the top edge of the surface during transient. The peak and steady state values for velocity at the top boundary at 0.0614 m/s and 0.0584 m/s respectively. The corresponding temperatures are 73.50 °C and 70.99 °C.

Below in **Figures 4-9**, are a series of surface plots for $q_w = 220\text{W/m}^2$ that illustrate the fluid behavior throughout the start-up transient.

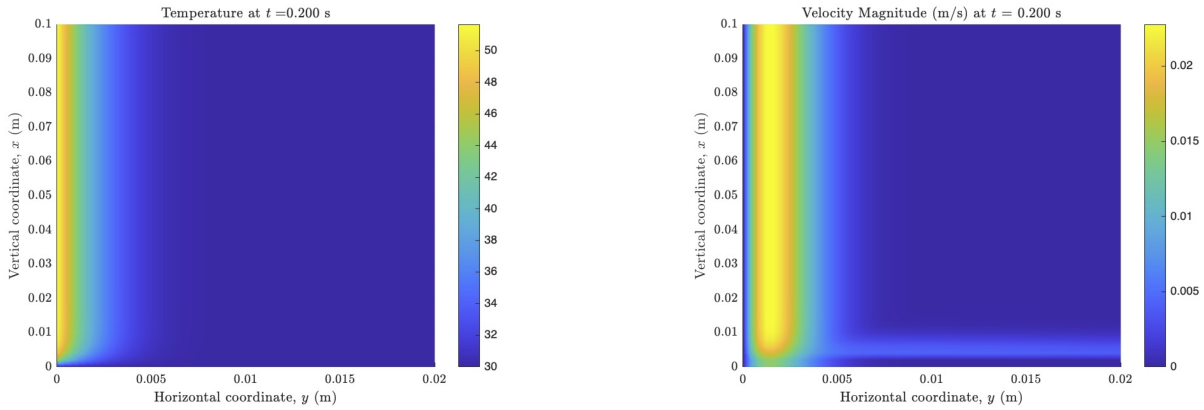

Figure 4: Temperature and Velocity Profiles at $t = 0.2$ (s) for $q_w = 220\text{W/m}^2$

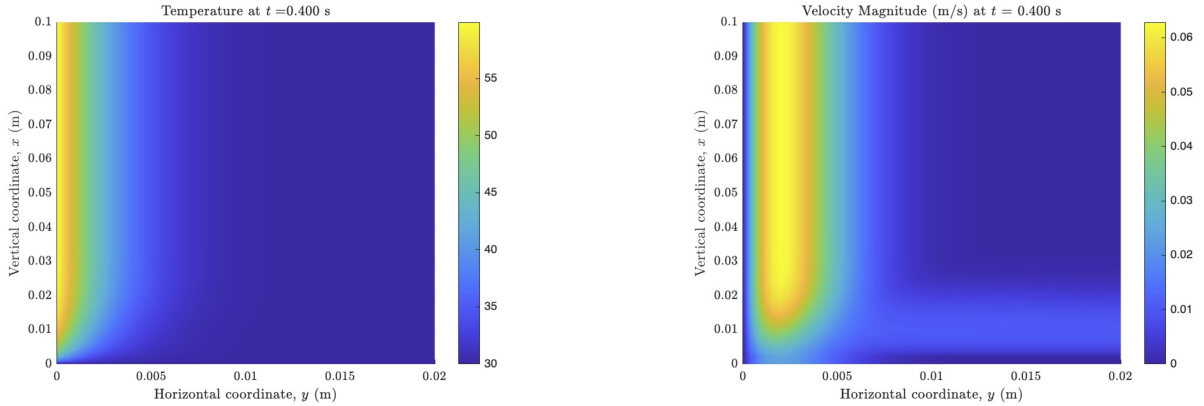

Figure 5: Temperature and Velocity Profiles at $t = 0.4$ (s) for $q_w = 220\text{W/m}^2$

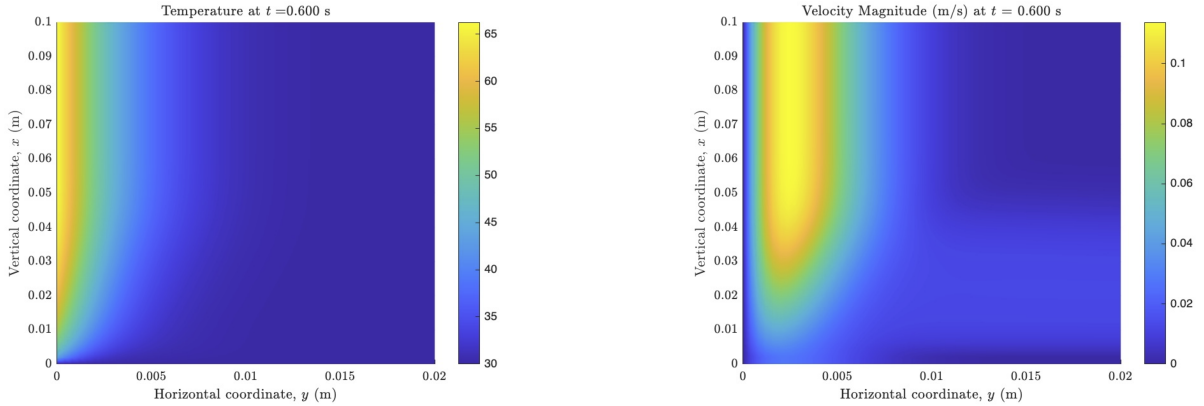

Figure 6: Temperature and Velocity Profiles at $t = 0.6$ (s) for $q_w = 220\text{W/m}^2$

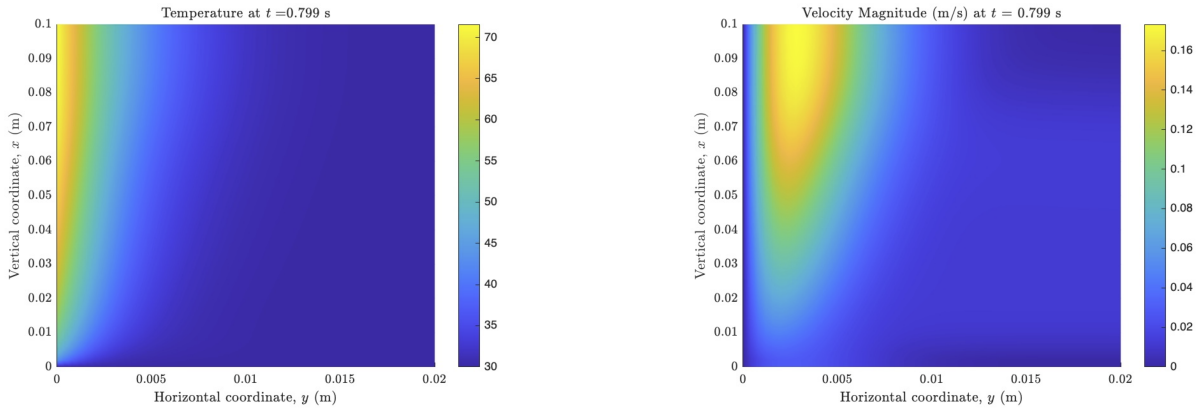

Figure 7: Temperature and Velocity Profiles at $t = 0.8$ (s) for $q_w = 220\text{W/m}^2$

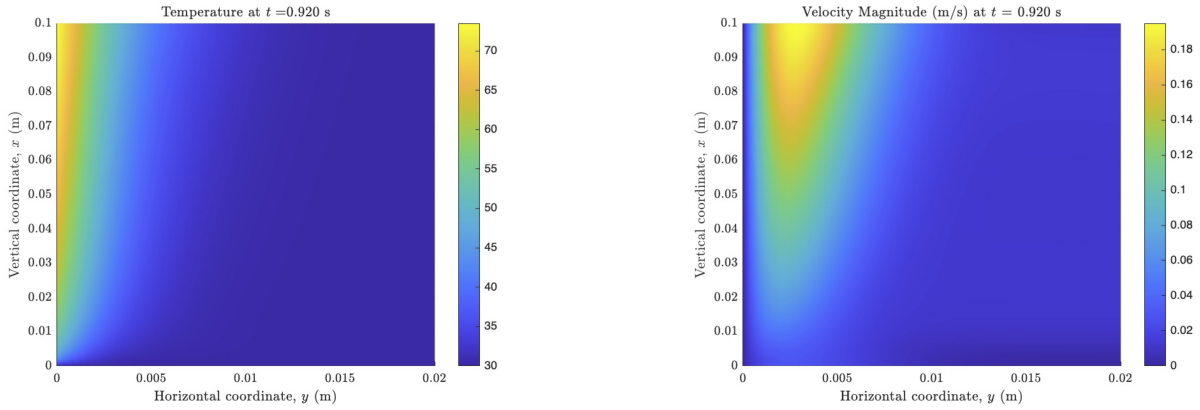

Figure 8: Temperature and Velocity Profiles at (peak) $t = 0.9205$ (s) for $q_w = 220\text{W/m}^2$

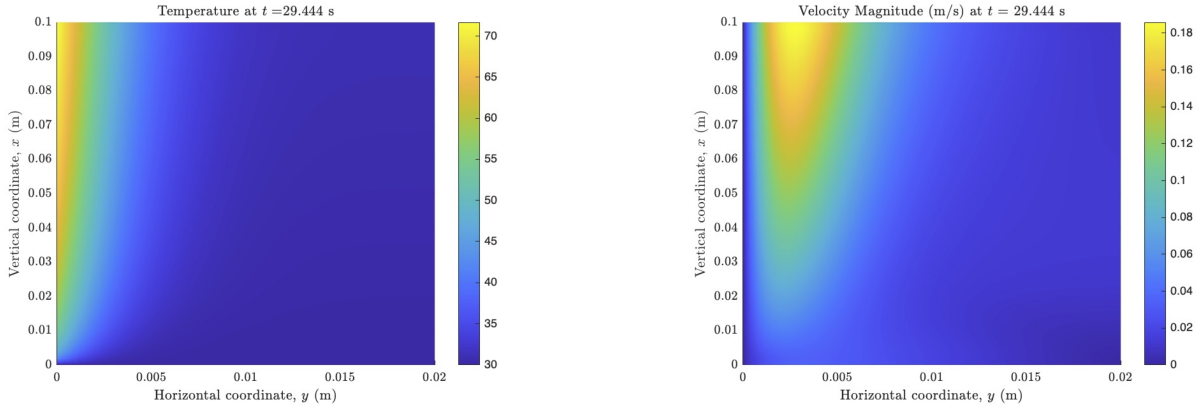

Figure 9: Temperature and Velocity Profiles at steady-state for $q_w = 220\text{W/m}^2$

The series of contours above show a very detailed and illustrative picture of how the field variables change during the transient. For the temperature field, it can be seen that there is a relatively constant temperature along the heated wall at t = 0.2 seconds. Following the left column of the screen through Figure 19, the temperature field can be seen to develop into its steady state solution. The same can be said with the velocity

field on the right, as it can be seen to start to develop through the transient. Additionally, the same plots above are shown in a 3D isometric orientation that can better illustrate some boundary layer development and especially velocity physical effects and their development in the transient.

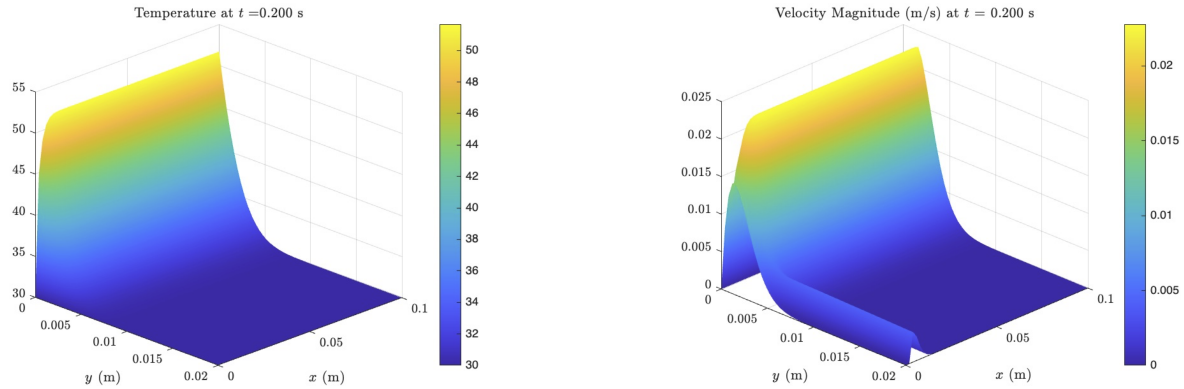

Figure 10: 3D Temperature and Velocity Profiles at $t = 0.2$ (s) for $q_w = 220\text{W/m}^2$

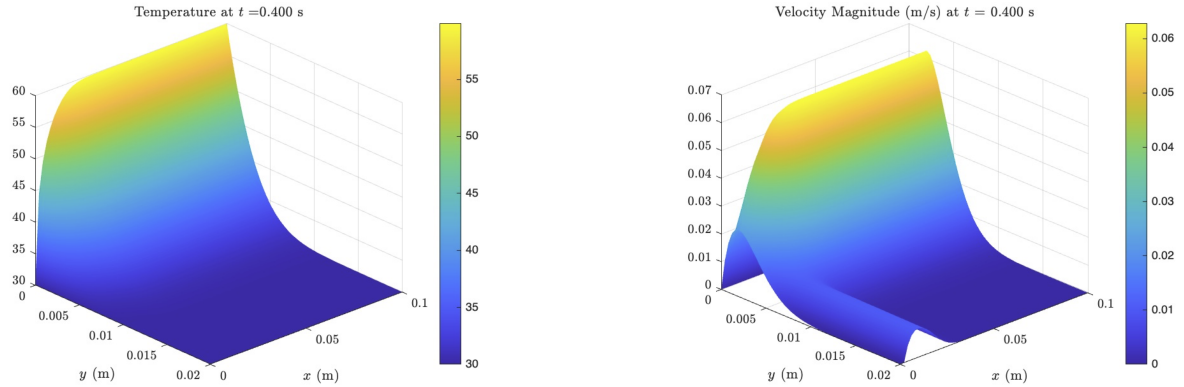

Figure 11: 3D Temperature and Velocity Profiles at $t = 0.4$ (s) for $q_w = 220\text{W/m}^2$

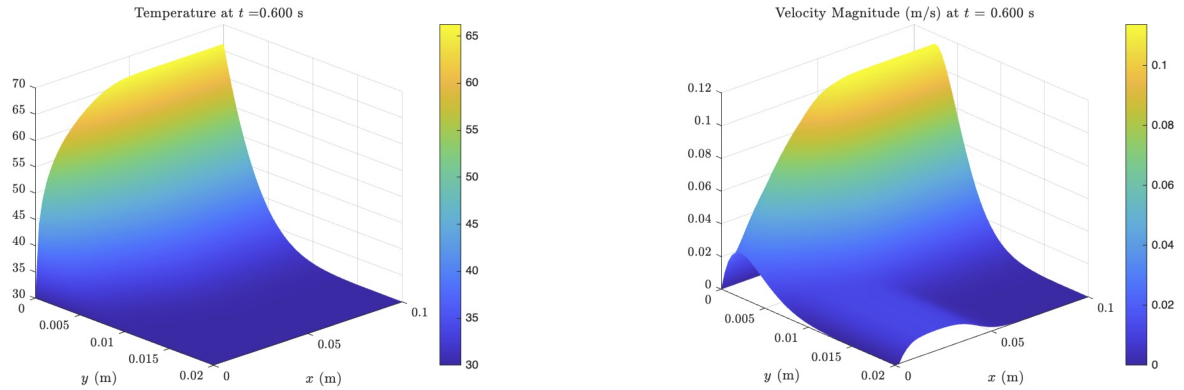

Figure 12: 3D Temperature and Velocity Profiles at $t = 0.6$ (s) for $q_w = 220\text{W/m}^2$

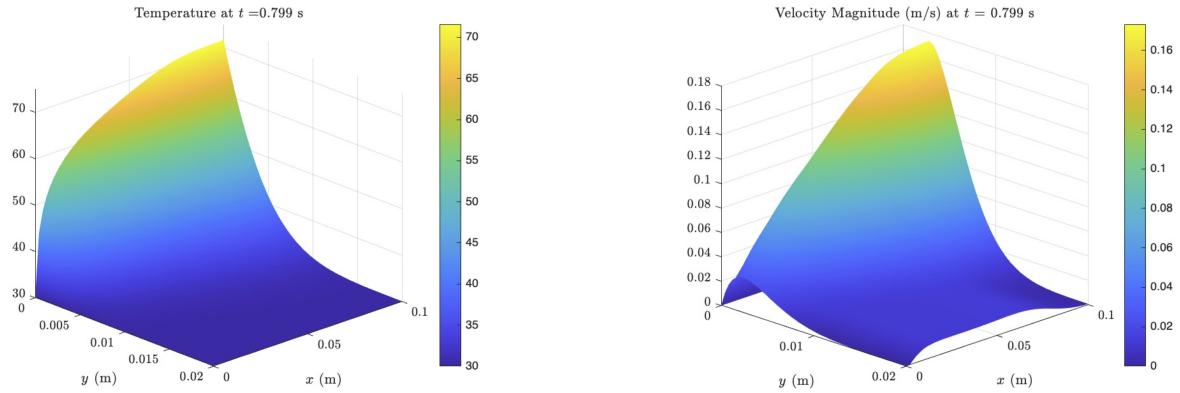

Figure 13: 3D Temperature and Velocity Profiles at $t = 0.8$ (s) for $q_w = 220\text{W/m}^2$

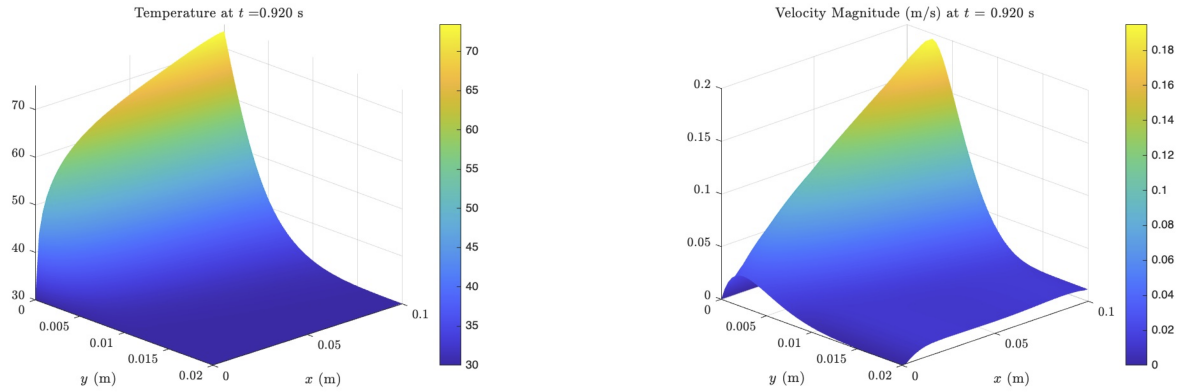

Figure 14: 3D Temperature and Velocity Profiles at (peak) $t = 0.9205$ (s) for $q_w = 220\text{W/m}^2$

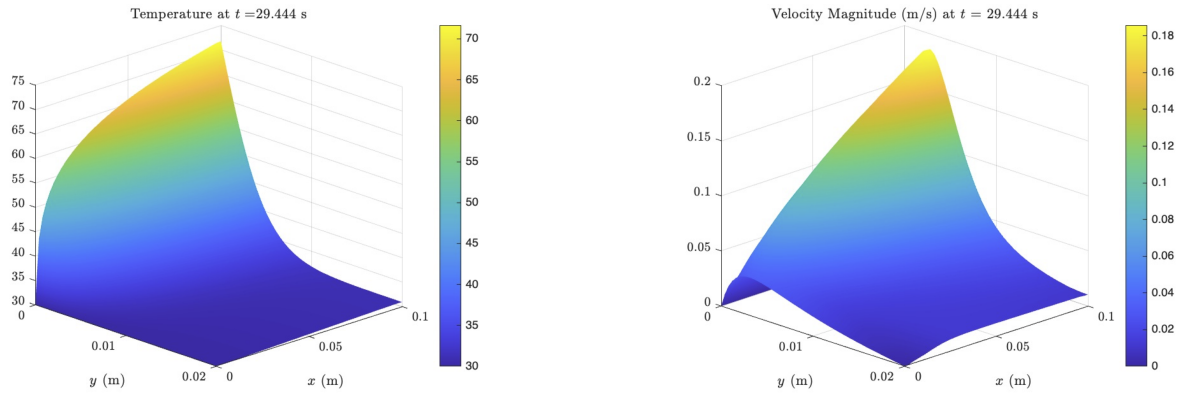

Figure 15: 3D Temperature and Velocity Profiles at steady-state for $q_w = 220\text{W/m}^2$

Also for $q_w = 220\text{W/m}^2$, as we plot the v velocity at the outer edge of the boundary layer as a function of x below.

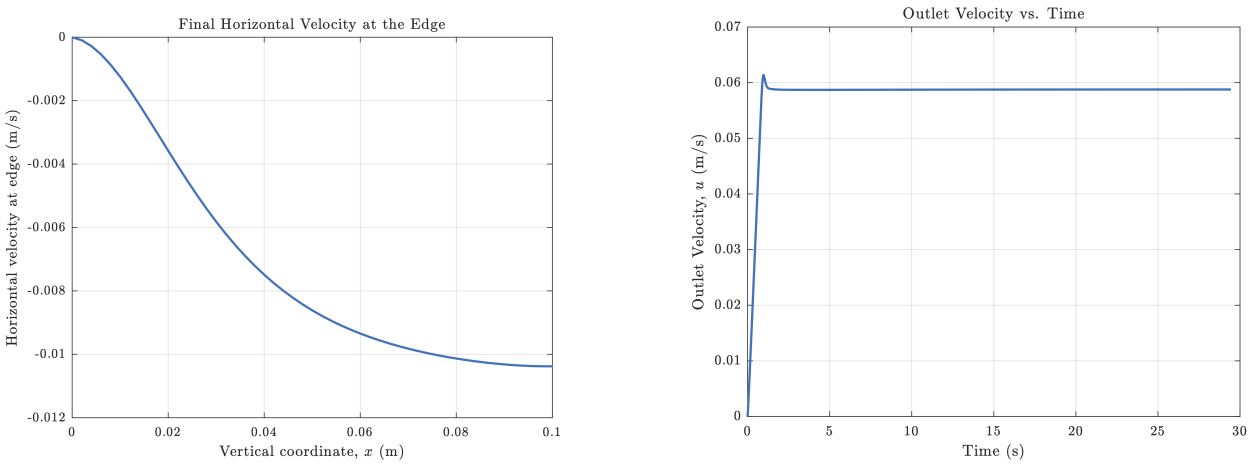

Figure 16: Horizontal Velocity at Edge and Outlet Velocity for $q_w = 220\text{W/m}^2$

Notice that at the edge of the boundary layer as $x \to 0$, v also tend to 0. This makes sense due to the no slip boundary condition implemented at the heated wall, because this boundary condition means that in order for there to be any horizontal velocity at the free edge, there would have to be no u velocity, which is impossible in the free convective problem we have.

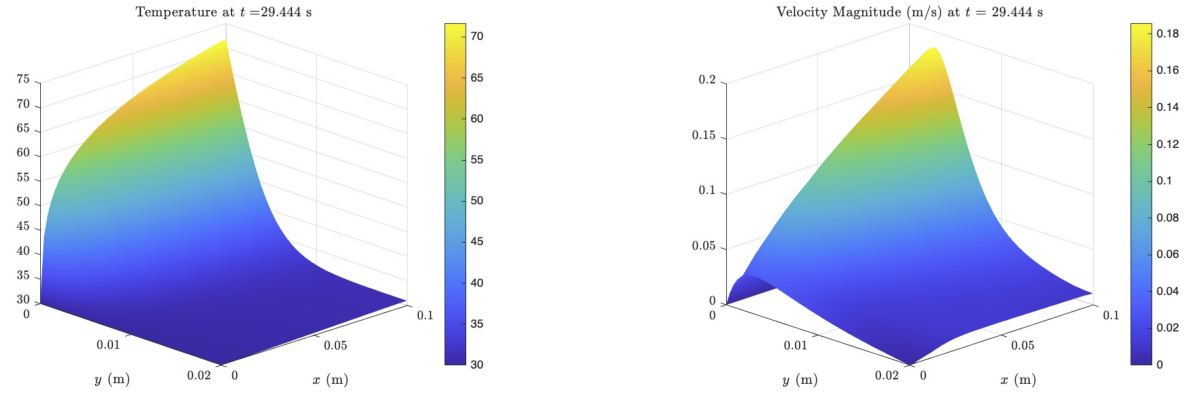

Figure 17: 3D Temperature and Velocity Profiles at steady-state for $q_w = 220\text{W/m}^2$

The transient behavior results indicate that the flow reaches a steady state in a smooth and gradual manner, with no signs of turbulent behavior, despite minor numerical artifacts. The temperature overshoot observed using the FTCS method quickly dampens out, and the velocity profile adheres to the no-slip boundary condition at the heated wall.

We now generate a variety of temperature and velocity steady-state solutions for $q_w = 220\text{W/m}^2$ and $q_w = 150\text{W/m}^2$ at the surface. These are shown in **Figures 18-20** and our interpretation is below.

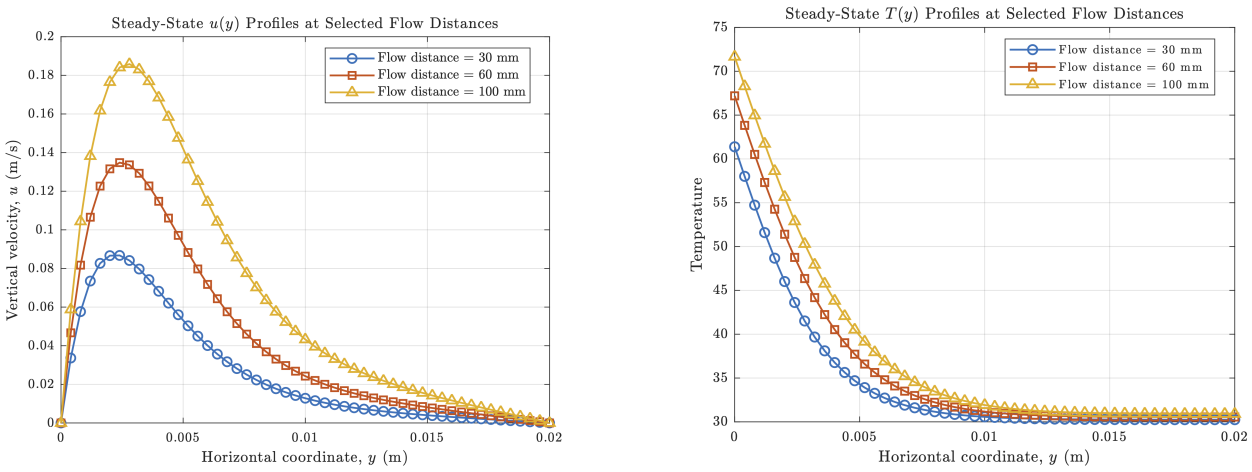

Figure 18: Steady-state Temperature and Velocity Profiles at Specified Flow Distances for $q_w = 220\text{W/m}^2$

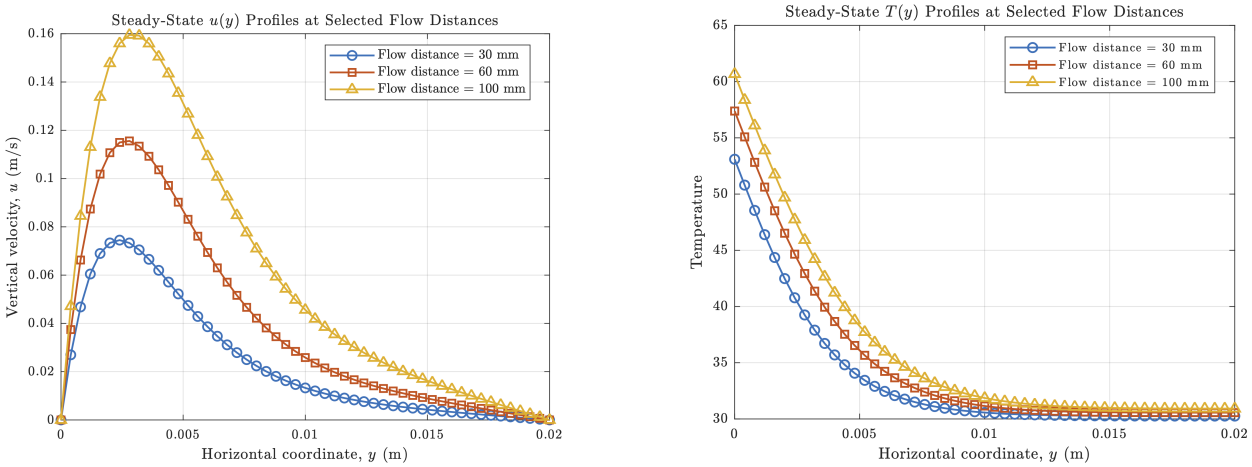

Figure 19: Steady-state Temperature and Velocity Profiles at Specified Flow Distances for $q_w = 150\text{W/m}^2$

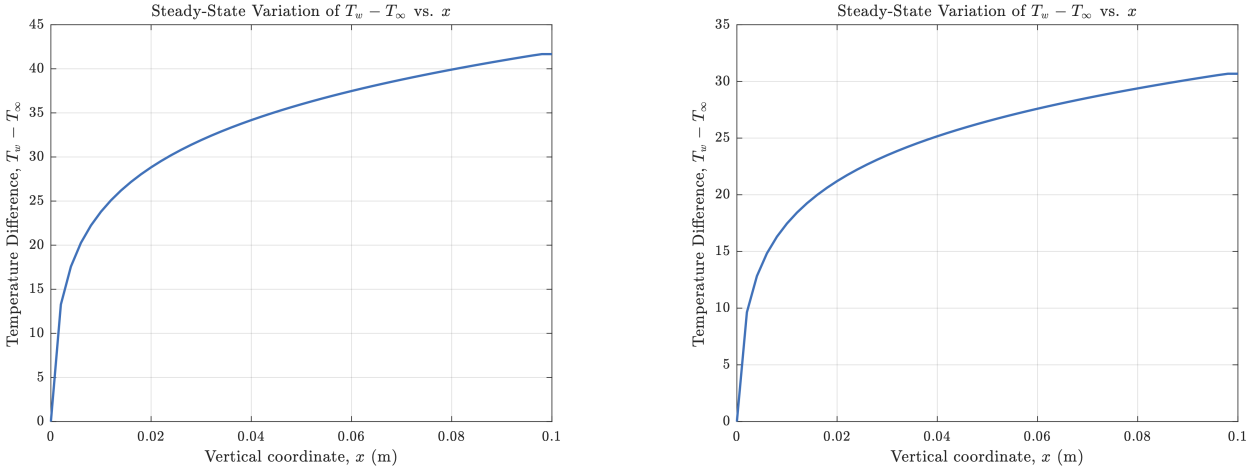

Figure 20: Steady-state Variation of $T_w - T_\infty$ vs. x for $q_w = 220\text{W/m}^2$(left), $q_w = 150\text{W/m}^2$(right)

Immediately, we notice that for $q_w = 220\text{W/m}^2$ all the field variables are higher than those where the heat flux is $q_w = 150\text{W/m}^2$ which intuitively makes sense as we generate a larger temperature difference implying stronger buoyancy forces so our boundary layer-velocity is higher and thicker. We also notice that as we increase the downstream distance, our velocities and temperatures increase. As for the temperature variations, we see that there is the strongest heating near the bottom of the plate as the fluid is just starting to move. Both plots are steadily decreasing in $T_w - T_\infty$ with an increasing x implying the top of the plate is higher temperature.

Transition in a free convection boundary layer depends on the relative magnitude of the buoyancy and viscous forces in the fluid. Thus, for the conditions considered here, we firstly calculate the Rayleigh number and see if it is below the critical number

$$\text{Ra}_x = \text{Gr}_x\text{Pr} = \frac{g\beta(T_s - T_\infty)x^3}{\nu\alpha}$$

From **Figure 20**, we see that if $q_w = 220\text{W/m}^2$, at the end of the plate, we have a maximum temperature difference of $T_w - T_\infty \approx 41°\text{C}$ and for $q_w = 150\text{W/m}^2$, at the end of the plate, we have a maximum temperature difference of $T_w - T_\infty \approx 30°\text{C}$. Then, substituting in all the flow parameters, we get

$$\begin{aligned}
\frac{g\beta(T_s - T_\infty)x^3}{\nu\alpha} &= \frac{(9.81)(0.0033)(41)(0.1)^3}{(1.613\times10^{-5})(2.2\times10^{-5})} \\
&= 3.74\times10^6 < 10^9 \quad \text{for } q_w = 220\text{W/m}^2 \\
\frac{g\beta(T_s - T_\infty)x^3}{\nu\alpha} &= \frac{(9.81)(0.0033)(30)(0.1)^3}{(1.613\times10^{-5})(2.2\times10^{-5})} \\
&= 2.74\times10^6 < 10^9 \quad \text{for } q_w = 150\text{W/m}^2
\end{aligned}$$

Since both Ra are below the critical Rayleigh number, we are in the laminar region. This is also verified by looking at our graphs. Our velocity profiles are smooth and show gradual variations across the boundary layer without abrupt fluctuation or erratic differences. Similarly, the temperature profile is smooth and continuous, and no oscillations which would be indicative of turbulent flow. Quantitatively, the peak velocities of 0.0614 m/s and a steady-state value of 0.0584 m/s are relatively slow indicating the inertial effects are much smaller than the viscous effects.

3.2 Task II Results and Discussion

In Section 1.2.2, we determined the constants $C_{2.1} = 3$ and $C_{2.2} = 4$ to end up with a system of ODEs with boundary conditions. To solve the system, normally one must guess $F''(0)$ and $H(0)$, compute variations of F, F', and H with η, and then check to see if the boundary conditions $F(\infty) = 0$ and $H(\infty) = 0$ are satisfied, new guesses for $F''(0)$ and $H(0)$ are generated and the process is iterated until the far-field boundary conditions are satisfied. Determination of the variations of F, F' and H with η from boundary conditions at $\eta = 0$ is accomplished using the 4th Order Runge-Kutta Scheme for numerically solving ODE's.

Numerical solutions have been obtained and reported in the literature. In particular

$$\text{for Pr} = 0.733: \quad F''(0) = 0.80893, \quad H(0) = 1.47981$$

We used these and the RK4 scheme to compute variations of F, F', and H with η for $\text{Pr} = 0.733$ using the above boundary conditions for $F''(0)$ and $H(0)$ together with $F(0) = 0, F'(0) = 0, H'(0) = -1$. We then plot the resulting variations of F' and H with η below

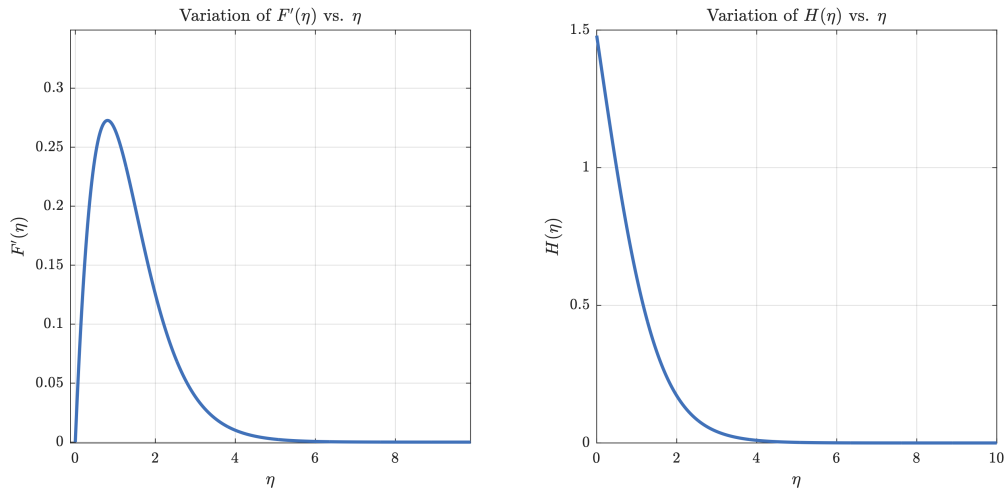

Figure 21: Variations of F' and H with η

These graphs show our similarity solutions to the boundary layer equations. We see that the $F'(\eta)$ rises fast and drops off just as fast and physically $\eta = 0$ and as expected, if $\eta \to \infty$ then the velocity gradient vanishes. From our steady-state u velocity and temperature field results for $x = 30, 60$, and 100 mm at $q'' = 220\text{W/m}^2$ and $q'' = 150\text{W/m}^2$ from **Task I** are expressed in terms of $F'(\eta), H$, and η.

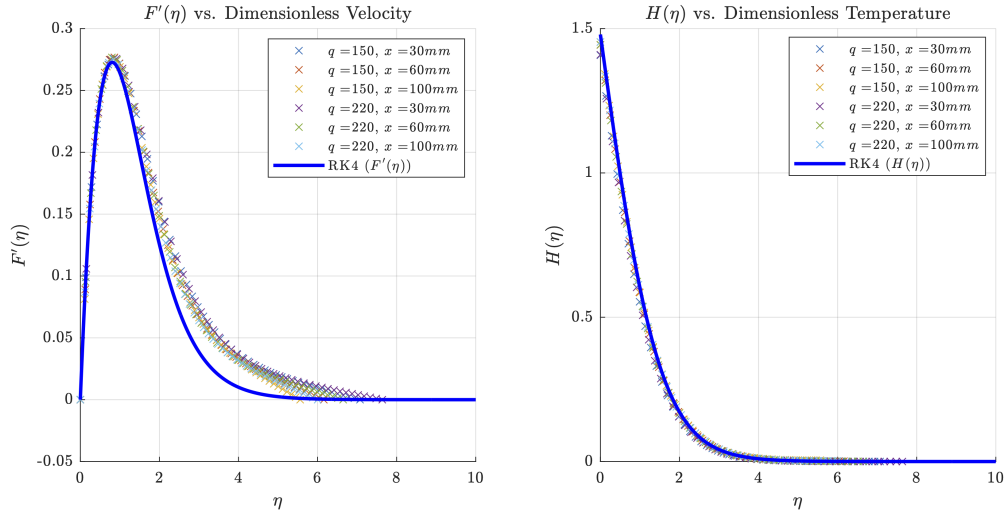

Figure 22: Similarity Functions vs. Dimensionless Field Variables

We also plot the variation of $T_w - T_\infty$ with x predicted by the finite difference and similarity solutions at $q'' = 220\text{W/m}^2$ on a $\log - \log$ plot. The RK4 solution actually matches pretty well with our FD solution at various downstream locations; they tend to collapse down onto the similarity solution especially temperature. Velocity we notice a little discrepency when $\eta > 2$.

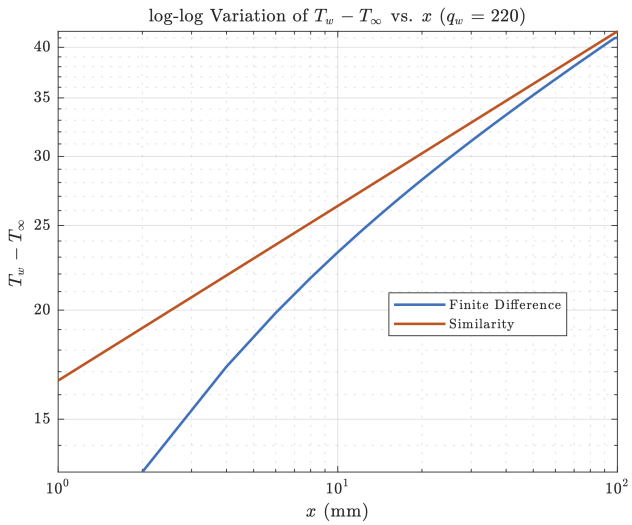

Figure 23: log-log plot of $T_w - T_\infty$ vs. x for $q_w = 220\text{W/m}^2$

There is a significant difference between the two approaches as the similarity and FD solutions have an offset in temperature differences especially near the leading edge. The FD boundary conditions are using the raw boundary layer equations rather than making idealized assumptions as the similarity variables do. However, it is still showing similar scaling behaviour for the wall surface temperature as we vary x.

3.3 Task III Results and Discussion

Using the integral solutions for u and T determined in Section 1.2.3, we predict the u and T profiles at $x = 30$ mm and 60 mm for the flow conditions considered in **Task I**. The flow conditions are given below

$$
\begin{aligned}
g &= 9.8\,\text{m/s}^2 && \text{(gravitational acceleration)},\\
\beta &= 0.0033\,\text{K}^{-1} && \text{(air thermal expansion coefficient)},\\
\nu &= 1.613\times10^{-5}\,\text{m}^2/\text{s} && \text{(air kinematic viscosity)},\\
\alpha &= 2.200\times10^{-5}\,\text{m}^2/\text{s} && \text{(air thermal diffusivity)},\\
k &= 0.0261\,\text{W/(m K)} && \text{(air thermal conductivity)},\\
T_\infty &= 30^\circ\text{C} && \text{(ambient air temperature)},\\
q_w &= 220\,\text{W/m}^2 && \text{(surface heat flux)},
\end{aligned}
$$

$$
\begin{aligned}
\Delta x &= 2\,\text{mm},\\
\Delta y &= 0.4\,\text{mm},\\
\Delta t &= 0.0005\,\text{s}.
\end{aligned}
$$

The rest of the work is shown below. All the expressions we are substituting in the values are found in Section 1.2.3: Task III.i Complete Solution for the Fields and δ.

Proof. Firstly, we need to determine the boundary layer thickness as both the velocity and temperature

fields require it.

$$\begin{aligned}\delta(x) &= \left(\frac{72\alpha k(4\alpha+5\nu)}{g\beta q_w}\right)^{1/5} x^{0.2} \\ &= \left(\frac{72(2.2\times10^{-5}\text{m}^2/\text{s})(0.0261\text{W}/(\text{mK}))(4(2.2\times10^{-5}\text{m}^2/\text{s})+5(1.613\times10^{-5}\text{m}^2/\text{s}))}{9.8\text{m}/\text{s}^2\cdot 0.0033\text{K}^{-1}\cdot 220\text{W}/\text{m}^2}\right)^{1/5} x^{0.2} \\ &= 0.015785\cdot x^{0.2}\end{aligned}$$

So now evaluating the boundary layer thickness at both $x = 30\text{mm} = 0.03\text{m}$

$$\delta(0.03\text{m}) = 0.015785(0.03\text{m})^{0.2} = 0.00783\text{m}$$
$$\delta(0.06\text{m}) = 0.015785(0.06\text{m})^{0.2} = 0.00899\text{m}$$

Let's start by evaluating the integral solution for u at $x = 30$ mm or $x = 0.03$m.

$$\begin{aligned}u(x,y/\delta) &= 60\alpha\left(\frac{g\beta q_w}{72\alpha k(4\alpha+5\nu)}\right)^{2/5} x^{0.6}\cdot\frac{y}{\delta}\left(1-\frac{y}{\delta}\right)^2 \\ u(0.03,y/\delta) &= 60\alpha\left(\frac{g\beta q_w}{72\alpha k(4\alpha+5\nu)}\right)^{2/5} (0.03)^{0.6}\cdot\frac{y}{\delta(0.03)}\left(1-\frac{y}{\delta(0.03)}\right)^2 \\ &= 0.646188\frac{y}{\delta(0.03)}\left(1-\frac{y}{\delta(0.03)}\right)^2 \\ &= 0.646188\frac{y}{0.00783}\left(1-\frac{y}{0.00783}\right)^2 \Rightarrow u(0.03,y/\delta) = 82.527y\left(1-\frac{y}{0.00783}\right)^2\end{aligned}$$

Similarly, the integral solution for u at $x = 60$ mm or $x = 0.06$ m is

$$\begin{aligned}u(x,y/\delta) &= 60\alpha\left(\frac{g\beta q_w}{72\alpha k(4\alpha+5\nu)}\right)^{2/5} x^{0.6}\cdot\frac{y}{\delta}\left(1-\frac{y}{\delta}\right)^2 \\ u(0.06,y/\delta) &= 60\alpha\left(\frac{g\beta q_w}{72\alpha k(4\alpha+5\nu)}\right)^{2/5} (0.06)^{0.6}\cdot\frac{y}{\delta(0.06)}\left(1-\frac{y}{\delta(0.06)}\right)^2 \\ &= 0.97944\frac{y}{\delta(0.06)}\left(1-\frac{y}{\delta(0.06)}\right)^2 \\ &= 0.97944\frac{y}{0.00899}\left(1-\frac{y}{0.00899}\right)^2 \Rightarrow u(0.06,y/\delta) = 108.92y\left(1-\frac{y}{0.00899}\right)^2\end{aligned}$$

Now, for the temperature fields T, we start with $x = 30$ mm or $x = 0.03$ m.

$$\begin{aligned}T(x,y/\delta) &= T_\infty + \frac{q_w}{2k}\left(\frac{72\alpha k(4\alpha+5\nu)}{g\beta q_w}\right)^{1/5} x^{0.2}\left(1-\frac{y}{\delta}\right)^2 \\ T(0.03,y/\delta) &= T_\infty + \frac{q_w}{2k}\left(\frac{72\alpha k(4\alpha+5\nu)}{g\beta q_w}\right)^{1/5} (0.03)^{0.2}\left(1-\frac{y}{\delta(0.03)}\right)^2 \\ &= 30^\circ\text{C} + 32.99\left(1-\frac{y}{\delta(0.03)}\right)^2 \Rightarrow T(0.03,y/\delta) = 30^\circ\text{C} + 32.99\left(1-\frac{y}{0.00783}\right)^2\end{aligned}$$

Finally, for the temperature fields T with $x = 60$ mm or $x = 0.06$ m, we have

$$\begin{aligned}T(x,y/\delta) &= T_\infty + \frac{q_w}{2k}\left(\frac{72\alpha k(4\alpha+5\nu)}{g\beta q_w}\right)^{1/5} x^{0.2}\left(1-\frac{y}{\delta}\right)^2 \\ T(0.06,y/\delta) &= T_\infty + \frac{q_w}{2k}\left(\frac{72\alpha k(4\alpha+5\nu)}{g\beta q_w}\right)^{1/5} (0.06)^{0.2}\left(1-\frac{y}{\delta(0.06)}\right)^2 \\ &= 30^\circ\text{C} + 37.8989\left(1-\frac{y}{\delta(0.06)}\right)^2 \Rightarrow T(0.06,y/\delta) = 30^\circ\text{C} + 37.8989\left(1-\frac{y}{0.00899}\right)^2\end{aligned}$$

□

Furthermore, we then convert the results from above to the similarity variables used in **Task II**. The math is shown below. Although we explicitly show the calculation of the functions here, we wrote a MATLAB script to calculate it for us in order to keep all the precision lost in rounding by hand.

Proof. We know that in terms of similarity variables, we can write

$$u = 5\nu\xi^2 x^{3/5} F'(\eta) \quad \text{and} \quad T = T_\infty + \frac{q_w}{k} x^{1/5} \xi^{-1} H(\eta)$$

Then rearranging for both $F'(\eta)$ and $H(\eta)$ we get

$$F'(\eta) = \frac{u(x, y/\delta)}{5\nu\xi^2 x^{3/5}} \quad \text{and} \quad H(\eta) = \frac{T - T_\infty}{\frac{q_w}{k} x^{1/5} \xi^{-1}}$$

We now solve for $F'(\eta)$ and $H(\eta)$ at $x = 0.03$ m, where $\xi \triangleq \left(\frac{g\beta q_w}{5k\nu^2}\right)^{1/5} = 183.76$ using the flow conditions described in **Task I** and the integral solutions.

$$\begin{aligned} F'(\eta) &= \frac{u(0.03, y/\delta)}{5\nu\xi^2 (0.03)^{3/5}} \\ &= \frac{82.527y\left(1 - \frac{y}{0.00783}\right)^2}{5\nu\xi^2(0.03)^{3/5}} \Rightarrow F'(\eta) = 248.431y\left(1 - \frac{y}{0.00783}\right)^2 \end{aligned}$$

$$\begin{aligned} H(\eta) &= \frac{T(0.03, y/\delta) - T_\infty}{\frac{q_w}{k} x^{1/5} \xi^{-1}} \\ &= \frac{\cancel{30°C} + 32.99\left(1 - \frac{y}{0.00783}\right)^2 - \cancel{30°C}}{\frac{q_w}{k}(0.03)^{1/5}\xi^{-1}} \Rightarrow H(\eta) = 1.45\left(1 - \frac{y}{0.00783}\right)^2 \end{aligned}$$

But in order to express $F'(\eta)$ and $H(\eta)$ in terms of η, we need to change y to η. Recall that we defined $\eta \triangleq \xi \frac{y}{x^{0.2}} \Rightarrow y = \frac{\eta x^{0.2}}{\xi}$ which yields

$$F'(\eta) = 248.431\left(\frac{\eta(0.03)^{0.2}}{183.76}\right)\left(1 - \frac{\eta\frac{(0.03)^{0.2}}{183.76}}{0.00783}\right)^2 \Rightarrow F'(\eta) = 0.67047\eta(1 - 0.3447\eta)^2$$

$$H(\eta) = 1.45\left(1 - \frac{\eta\left(\frac{(0.03)^{0.2}}{183.76}\right)}{0.0783}\right) \Rightarrow H(\eta) = 1.45(1 - 0.3447\eta)^2$$

Similarity, we can also find the results using similarity variables for $x = 0.06$ m.

$$\begin{aligned} F'(\eta) &= \frac{u(0.06, y/\delta)}{5\nu\xi^2(0.03)^{3/5}} \\ &= \frac{108.92y\left(1 - \frac{y}{0.00899}\right)^2}{5\nu\xi^2(0.06)^{3/5}} \Rightarrow F'(\eta) = 216.321y\left(1 - \frac{y}{0.00899}\right)^2 \end{aligned}$$

$$\begin{aligned} H(\eta) &= \frac{T(0.06, y/\delta) - T_\infty}{\frac{q_w}{k} x^{1/5}\xi^{-1}} \\ &= \frac{\cancel{30°C} + 37.8989\left(1 - \frac{y}{0.00899}\right)^2 - \cancel{30°C}}{\frac{q_w}{k}(0.06)^{1/5}\xi^{-1}} \Rightarrow H(\eta) = 1.45034\left(1 - \frac{y}{0.00899}\right)^2 \end{aligned}$$

But in order to express $F'(\eta)$ and $H(\eta)$ in terms of η, we need to change y to η. Recall that we defined $\eta \triangleq \xi \frac{y}{x^{0.2}} \Rightarrow y = \frac{\eta x^{0.2}}{\xi}$ which yields

$$F'(\eta) = 216.321 \left(\frac{\eta (0.06)^{0.2}}{183.76} \right) \left(1 - \frac{\eta \frac{(0.06)^{0.2}}{183.76}}{0.00899} \right)^2 \Rightarrow F'(\eta) = 0.6706\eta(1 - 0.3448\eta)^2$$

$$H(\eta) = 1.45034 \left(1 - \frac{\eta \left(\frac{(0.06)^{0.2}}{183.76} \right)}{0.00899} \right) \Rightarrow H(\eta) = 1.45034(1 - 0.3448\eta)^2$$

□

Firstly, we plot the integral solution as a function of η in **Figure. 24**. The figure shows the integral solution at two downstream locations $x = 30$ mm and $x = 60$ mm. Notice that both the velocity plots follow a similar shape, as we increase η—moving right from the wall—we see the velocity profile rises to a peak and then decays to zero. At a streamwise location fo 60 mm, we see the velocity is overall higher than that of 30 mm indicating a stronger flow downstream which is consistent with the math as $u(x, y/\delta) \sim x^{0.6} \Rightarrow$ vertical velocity increases as we move further downstream.

The temperature plots for the integral solution decay from a maximum at the wall gradually to the ambient temperature of 30°C. At $x = 60$ mm, we have a shallower slope indicating a thicker boundary layer which would make sense as temperature increases at the wall upstream. It is also worth mentioning that the integral solutions for both plots are near a point of collapse, that is when they overlap each other, which indicates the self-similarity of the solution where the residual between the two streamwise locations is due to our polynomial approximation as there remains a small x-dependence.

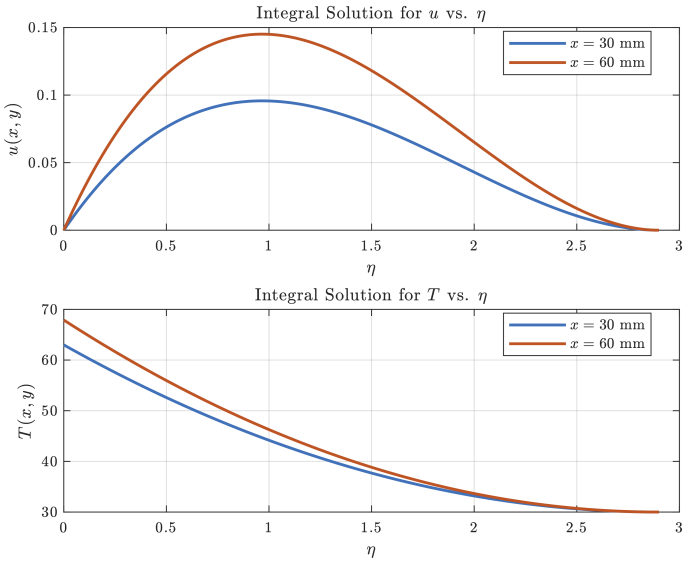

Figure 24: Integral Solutions of u and T at $x = 30, 60$ mm as a Function of η

The predictions of the similarity solution of F' and H as functions of η are displayed in **Figure. 25**.

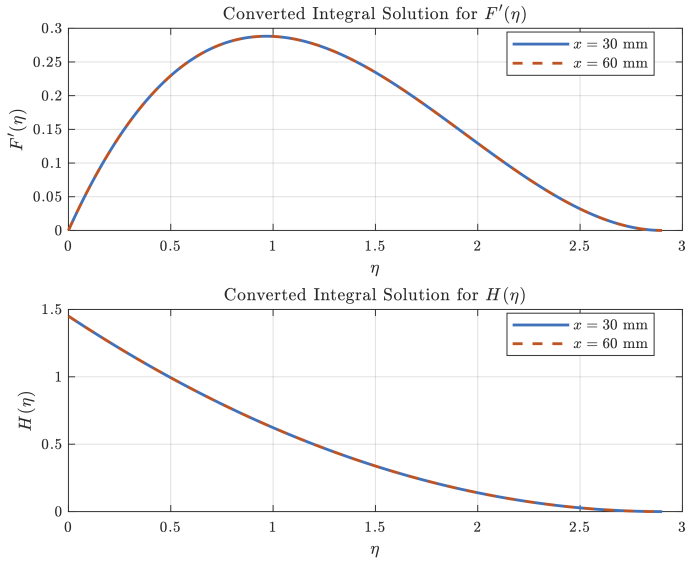

Figure 25: Similarity Solutions of u and T at $x = 30, 60$ mm as a Function of η

In both the dimensionless velocity profile, $F'(\eta)$, and the dimensionless temperature profile, $H(\eta)$, we evaluated the solution at two streamwise locations $x = 30$ mm and $x = 60$ mm. We see that if the appropriate self-similarity transformation is applied, the curves for these two different streamwise locations nearly overlap perfectly. Although there are slight numerical differences as shown above—which is expected due to our approximation of the polynomial in the integral method—the fact that the transformed profiles effectively collapsed onto one another means that the integral solution is self-similar.

In other words, once we introduce the similarity variable η and convert the physical velocity and temperature fields into $F'(\eta)$ and $H(\eta)$, any explicit x-dependence is removed, and our functional polynomial forms will accurately capture the self-similar boundary-layer structure as expected. In addition, we plot the variation of wall temperature $H(0)$ with x predicted by the integral and similarity solutions below in **Figure. 26**.

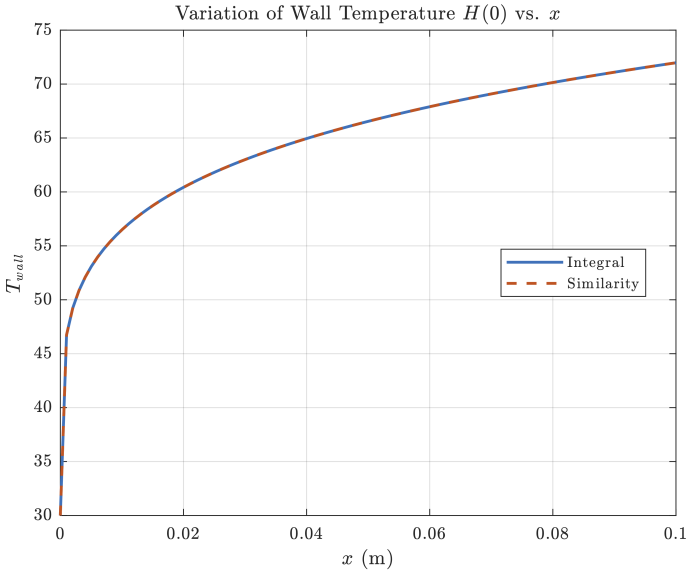

Figure 26: Variation of Wall Temperature $H(0)$ with Integral and Similarity solutions of x

It is clear from above that T_{wall}, changes with the streamwise location x with both the integral and similarity

solutions. Again, like above, the solutions collapse into one graph indicating that both methods are equivalently the same when predicting the growth of the vertical surface plate temperature; the integral solutions using a polynomial functional approximation captures the self-similarity in the boundary layer. A point of interest with the graph is the sudden steep rise in T_{wall} when x is near 0 which reflects the local effect of heating at the leading edge, the temperature profile is much more tapered as x increases which seems to suggest that as we go further downstream, the boundary layer grows thicker and the wall temperature levels off. This again confirms that our observation that the integral solution results are consistent with the similarity solutions results.

3.4 Task IV Results and Discussion

Since the majority of the discussion and results in this project have mostly been theoretical, we would like to apply what we have learned to a real world application regarding electronic device cooling. Consider a thin electronic device mounted on a vertical circuit board that is 3 cm tall and 5 cm wide. The board essentially insulates the back of the device so that all the heat generated by the device is transferred to the adjacent air on the opposite side by natural convection within a large space inside an electronics cabinet. The heat is generated uniformly over the surface of the device. In answering the questions below, use the properties for air indicated in **Task I**, and assume the ambient air is at 30°C.

If the 3cm $\times$ 5cm device dissipates 0.3 W uniformily over its surface, we want to check if the use of boundary layer theory is appropriate for this natural convection process. We quantitatively justify our answer below.

Task IV.i: To quantitatively determine if boundary layer theory is appropriate we do the following

Proof. We show that the boundary layer theory is applicable by checking the following condition

$$\frac{\delta}{x} \sim \frac{1}{\text{Gr}_x^{1/4}} \Rightarrow 10^4 < \text{Gr}_x < 10^9 \Rightarrow 10 < \frac{\delta}{x}$$

where Gr_x is the Grashof number and is defined as

$$\text{Gr}_x \triangleq \frac{gx^3\beta\Delta T}{\nu^2}$$

If the Grashof number is below this value (close to the leading edge), the boundary layer thickness is too large relative to the characteristic length x to ensure the validity of the approximations. Above, 10^9, we have a turbulent region. However, based on the governing equations provided, we are assuming laminar flow so

$$10^4 \lesssim \text{Ra}_L \lesssim 10^9$$

In which case, we invoke the following correlation

$$\overline{\text{Nu}}_L = \frac{\overline{h}L}{k} = C\text{Ra}_L^n = C(\text{Gr}_L\text{Pr})^n = 0.59(\text{Gr}_L\text{Pr})^{1/4}$$

where $\overline{\text{Nu}}_L$ is the average Nusselt Number, Pr is the Prandtl number, and Gr_L is the Grashof number based on the characteristic length L of the geometry. Then we use Newton's law of cooling and the dimensionless numbers above to get

$$q_w = h(T_s - T_\infty) = \frac{k\overline{\text{Nu}}_L}{L}\Delta T = \left(\frac{k}{L}0.59(\text{Gr}_L\text{Pr})^{1/4}\right)\Delta T \Rightarrow q_w = \frac{k}{L}0.59\left(\text{Pr}\frac{g\beta(\Delta T)L^3}{\nu^2}\right)^{1/4}\Delta T$$

The way we solve for ΔT is iteratively. We have written a code in MATLAB to perform a divide and conquer bisection algorithm which is explained in Section 2.4. From this, we achieved a $\Delta T = 26.76$K.

Then calculating the Grashof number we get

$$\mathrm{Gr}_L = \frac{g\beta\Delta T L^3}{\nu^2} = \frac{(9.8\mathrm{m/s^2})(0.0033\mathrm{K^{-1}})(26.76\mathrm{K})(0.03\mathrm{m})^3}{(1.613\times10^{-5}\mathrm{m^2/s})^2} = 8.98\times10^4$$

Which means that we are in fact in the laminar free convection regime where boundary layer flow does occur and it is appropriate to use boundary layer theory for this natural convection process. □

Now, assuming boundary layer analysis is applicable, if the maximum allowable surface temperature at any location on the device is 70°C, we determine the maximum rate of heat generation that the device can reject by natural convection to air at an ambient temperature of 30°C as follows

Task IV.ii: We start with the boundary layer analysis expressions, namely the temperature field.

Proof. Since we require that the maximum surface temperature to be no more than 70°C and the ambient air temperature is 30°C, we have $T - T_\infty = 40°\mathrm{C}$. Then solving for q_w, we get

$$\frac{q_w B x^n}{2k} = T - T_\infty$$
$$\frac{q_w B x^n}{2k} = 40 \Rightarrow q_w = \frac{80k}{Bx^n} \quad \text{where } B \triangleq \left(\frac{72\alpha k(4\alpha+5\nu)}{g\beta q_w}\right)^{1/5}$$

We use $x = 0.03$ m since the boundary layer develops along the vertical direction. Also, note that n was determined to be 0.2 and B can be calculated using the flow conditions from **Task I** to be $0.015785\ \mathrm{m}^{0.8}$.

$$\max q_w = \frac{80(0.0261)}{(0.015785)(0.03)^{0.2}} = 266.72\mathrm{W/m^2}$$

So, the maximum rate of heat generation that the device can reject by natural convection to air is

$$\max \dot{Q} = \max q_w \cdot A = 266.72 \cdot (0.03\times0.05) = 0.4\mathrm{W}$$

□

We modified the boundary conditions in the finite difference formulation used in **Task I** at the heated component surface to include radiation exchange with the surroundings at temperature T_∞. We assume that the components are gray bodies with surface total hemispherical emissivity ε_s, and that the surface is surrounded by a blackbody enclosure at T_∞. At the boundary, the finite difference solution must satisfies

$$\frac{-q_e}{k} = \frac{T_{i,2} - T_{i,1}}{\Delta y} - \frac{\sigma\varepsilon_s}{k}(T_{i,1}^4 - T_\infty^4) \tag{21}$$

where$\sigma = 5.67\times10^{-8}\mathrm{W/m^2K^4}$ is the Stefan-Boltzmann constant. We developed a way to implement this boundary condition in the finite difference algorithm as explained above in Section 2.3. This code was then used to determine that the maximum rate of heat generation that the device can reject by natural convection and radiation to air and surrounding surfaces at 30°C assuming that the surface of the device has an emissivity of 0.85 to be 0.32 W. This value was tabulated by iterating over the wall heat flux values, solving for radiative heat flux given those temperatures, and running until the solution converges. Since it was given in the problem statement that the chip has a maximum allowable temperature of 70∘C, once the solution converged on a temperature above that threshold, the previous conductive and radiative heat flux values were chosen and the total heat flux was found by summing the two numbers. Then the maximum allowable rate of heat generation was found by multiplying by the total area of the chip.

If the circuit board with the heat dissipating device is located at a research base on the surface of the moon, which has gravitational acceleration of $1.62\mathrm{m/s^2}$ ($\sim 1/6$ of Earth's), the Grashof number would just be above

the lower threshold since $1/6 \cdot 8.98 \times 10^4 \approx 1.49 \times 10^4 > 10^4$. However, it is worth noting that the bounds are merely approximate and we should definitely consider another approach since we are approaching the border of what is acceptable. As for (IV.iv), we ran the same code following the same steps as outlined in the previous paragraph, and found that the maximum allowable heat generation decreased to 0.21 W.

4 Conclusion

This study investigated laminar and buoyancy driven boundary layer flow near a vertical surface while heated at a uniform flux. Three different analytical and numerical approaches: a finite-difference simulation, a similarity solution, and an integral boundary layer model, were employed to understand both the transient and steady-state behavior. Despite relying on distinct assumptions and levels of complexity, all methods converged on consistent predictions of velocity fields, temperature distributions, boundary layer thickness, and overall flow structure.

The explicit finite-difference scheme revealed how the flow starts up following a sudden heat input, tracking how leading-edge disturbances progress in time. The similarity solution condensed the original partial differential equations into ordinary differential equations, confirming that at steady conditions, the boundary layer exhibits self-similarity with minimal dependence on the distance from the leading edge. Meanwhile, the integral boundary layer approach, though simpler, still produced results that agreed closely with both the similarity and finite-difference analyses, showcasing its practicality for rapid engineering estimates. The finite difference method simulation showed results similar to that of both analytical methods, but instead also demonstrated the start up transient thermal and velocity effects.

Finally the framework was applied to cooling an electronic device under both terrestrial and reduced (lunar) gravity, illustrating how modest changes in gravitational acceleration or inclusion of radiative heat transfer can significantly alter thermal performance limits. These results underscore the robustness and flexibility of boundary layer approximations for buoyancy-driven convection and highlight how various solution strategies from quick integral estimates to detailed time-dependent simulations can be used effectively to design and assess natural-convection flows.

5 Code Appendix

This appendix will include all the code relevant for **Task I**, **Task II**, **Task IV**. The code is also commented to ease your understanding and to display our thought process while writing.

5.1 Task I Code

```
clear; clc; close all;

%% Flow Conditions and Geometric Parameters
g = 9.81;             % gravity
beta = 3.3e-3;        % thermal expansion coeff.
nu = 1.5e-5;          % kinematic vis.
alpha = 2.2e-5;       % thermal diff.
k_air = 0.0261;       % thermal cond. of air
Tinf = 30;            % ambient temp.
q_w = 220;            % surface heat flux

% Mesh Grid Dimensions
Nx = 51;              % points in vertical (x) direction
Ny = 51;              % points in horizontal (y) direction

% Grid spacings, simulation time, and tolerance
dx = 2.0e-3;          % vertical grid spacing
dy = 0.4e-3;          % horizontal grid spacing
dt = 5.0e-4;          % time step
maxTime = 0.9205;        % time limit
Tol = 1.0e-6;         % convergence tolerance CAN BE CHANGED

% Initializing coordinate vectors
x = linspace(0, (Nx-1)*dx, Nx);
y = linspace(0, (Ny-1)*dy, Ny);
numSteps = ceil(maxTime/dt);

%% Array Initialization
u_old = zeros(Nx, Ny);          % u: vertical velocity component
v_old = zeros(Nx, Ny);          % v: horizontal velocity component
T_old = Tinf * ones(Nx, Ny);    % temperature field
u_new = u_old;
v_new = v_old;
T_new = T_old;

% Preallocate arrays
Tsurf_vals = zeros(1, numSteps);
time_vals  = zeros(1, numSteps);
u_inlet_vals = zeros(1, numSteps);
usurf_vals = zeros(1, numSteps);

%% Time Loop for FTCS
for n = 1:numSteps

    % Ensure local scope
    u = u_old;
    v = v_old;
    T = T_old;

    % Update temperature in interior
    for i = (Nx-1):-1:2
        for j = 2:(Ny-1)

            % Upwind differencing in x
            if u(i,j) >= 0
                dTdx = (T(i,j) - T(i-1,j)) / dx;
            else
                dTdx = (T(i+1,j) - T(i,j)) / dx;
            end
```

```
60
61              % Upwind differencing in y
62              if v(i,j) >= 0
63                  dTdy = (T(i,j) - T(i,j-1)) / dy;
64              else
65                  dTdy = (T(i,j+1) - T(i,j)) / dy;
66              end
67              convT = - ( u(i,j)*dTdx + v(i,j)*dTdy );
68
69              % Diffusion term
70              diffT_x = alpha * ( T(i+1,j) - 2*T(i,j) + T(i-1,j) ) / (dx^2);
71              diffT_y = alpha * ( T(i,j+1) - 2*T(i,j) + T(i,j-1) ) / (dy^2);
72
73              % New temperature
74              T_new(i,j) = T(i,j) + dt*( convT + diffT_x + diffT_y );
75          end
76      end
77
78      % Update vertical velocity in interior
79      for i = (Nx-1):-1:2
80          for j = 2:(Ny-1)
81
82              % Upwind differencing for u in x
83              if u(i,j) >= 0
84                  dudx = (u(i,j) - u(i-1,j)) / dx;
85              else
86                  dudx = (u(i+1,j) - u(i,j)) / dx;
87              end
88
89              % Upwind differencing in y
90              if v(i,j) >= 0
91                  dudy = (u(i,j) - u(i,j-1)) / dy;
92              else
93                  dudy = (u(i,j+1) - u(i,j)) / dy;
94              end
95
96              convU = - ( u(i,j)*dudx + v(i,j)*dudy );
97
98              % Diffusion term
99              diffU_x = nu * ( u(i+1,j) - 2*u(i,j) + u(i-1,j) ) / (dx^2);
100             diffU_y = nu * ( u(i,j+1) - 2*u(i,j) + u(i,j-1) ) / (dy^2);
101
102             % Bouyancy term
103             buoy = g * beta * ( T(i,j) - Tinf );
104
105             % New x-direction velocity
106             u_new(i,j) = u(i,j) + dt*( convU + diffU_x + diffU_y + buoy );
107         end
108     end
109
110     % Update horizontal velocity v from continuity
111     for i = (Nx-1):-1:2
112         for j = 2:(Ny-1)
113             du_dx = (u(i,j) - u(i-1,j)) / dx;
114             dv_dy = (v(i,j) - v(i,j-1)) / dy;
115             v_new(i,j) = v(i,j) - dt*( du_dx + dv_dy );
116         end
117     end
118
119     %% BOUNDARY CONDITIONS
120
121     % Left wall (constant heat flux)
122     for i = 1:Nx
123         u_new(i,1) = 0;
124         v_new(i,1) = 0;
125         % Apply constant heat flux in the y-direction:
126         T_new(i,1) = T_new(i,2) + (q_w * dy / k_air);
127     end
```

```
128
129     % Right boundary = zero-gradient
130     for i = 1:Nx
131         T_new(i,Ny) = T_new(i,Ny-1);
132         u_new(i,Ny) = 0;
133         v_new(i,Ny) = v_new(i,Ny-1);
134     end
135
136     % Bottom boundary = zero-gradient and T=Tinf
137     for j = 1:Ny
138         T_new(1,j) = Tinf;
139         u_new(1,j) = u_new(2,j);
140         v_new(1,j) = v_new(2,j);
141     end
142
143     % Top boundary = zero-gradient
144     for j = 1:Ny
145         T_new(Nx,j) = T_new(Nx-1,j);
146         u_new(Nx,j) = u_new(Nx-1,j);
147         v_new(Nx,j) = v_new(Nx-1,j);
148     end
149
150     % Top edge temperature
151     Tsurf_vals(n) = T_new(Nx,1);
152
153     % Bottom edge velocity
154     u_inlet_vals(n) = u_new(Nx,2);
155
156     % Store the current time
157     time_vals(n)  = (n-1)*dt;
158
159     % Check for convergence
160     diffU = max(abs(u_new(:) - u_old(:)));
161     diffV = max(abs(v_new(:) - v_old(:)));
162     diffT = max(abs(T_new(:) - T_old(:)));
163     maxChange = max([diffU, diffV, diffT]);
164
165     % Update fields for the next iteration
166     u_old = u_new;
167     v_old = v_new;
168     T_old = T_new;
169
170     % Check if steady-state is reached
171     if maxChange < Tol
172         fprintf('Steady-state reached at iteration %d, time ~ %.4f s\n', n, time_vals(n));
173         break;
174     end
175 end
176
177 final_time = time_vals(n);
178
179 %% Post-Processing
180
181 % Set all the graphs to LaTex
182 set(groot,'defaulttextinterpreter','latex');
183 set(groot, 'defaultAxesTickLabelInterpreter','latex');
184 set(groot, 'defaultLegendInterpreter','latex');
185
186 % Surface temp. vs time
187 figure;
188 plot(time_vals(1:n), Tsurf_vals(1:n), 'LineWidth', 1.5);
189 xlabel('Time (s)', Interpreter='latex');
190 ylabel('Surface Temperature', Interpreter='latex');
191 title('Surface Temperature ($x = 100$ mm, $j = 1$) vs. Time', Interpreter='latex');
192 grid on;
193
194 % Temperature field surface plot
195 figure;
```

```
surf(y, x, T_new);
shading interp;
view(2);
title(['Temperature at $t = $', num2str(final_time, '%.3f'), ' s']);
xlabel('Horizontal coordinate, $y$ (m)', Interpreter='latex');
ylabel('Vertical coordinate, $x$ (m)', Interpreter='latex');
colorbar;

% Velocity field surface plot
Umag = sqrt(u_new.^2 + v_new.^2);
figure;
surf(y, x, Umag);
shading interp;
view(2);
title(['Velocity Magnitude (m/s) at $t$ = ', num2str(final_time, '%.3f'), ' s'], Interpreter
    ='latex');
xlabel('Horizontal coordinate, $y$ (m)', Interpreter='latex');
ylabel('Vertical coordinate, $x$ (m)', Interpreter='latex');
colorbar;

% Horizontal velocity
figure;
plot(x, v_new(Nx,:), 'LineWidth', 1.5);
xlabel('Vertical coordinate, $x$ (m)', Interpreter='latex');
ylabel('Horizontal velocity at edge (m/s)', Interpreter='latex');
title('Final Horizontal Velocity at the Edge', 'Interpreter','latex');
grid on;

% Vertical velocities at different x's as a function of y
X_targets = [0.03, 0.06, 0.10];
i_profiles = round(X_targets/dx );
figure;
plot(y, u_new(i_profiles(1),:), '-o', ...
     y, u_new(i_profiles(2),:), '-s', ...
     y, u_new(i_profiles(3),:), '-^', 'LineWidth',1.2);
legend(['Flow distance = ', num2str(X_targets(1)*1e3), ' mm'], ...
       ['Flow distance = ', num2str(X_targets(2)*1e3), ' mm'], ...
       ['Flow distance = ', num2str(X_targets(3)*1e3), ' mm'], ...
       'Location', 'Best', Interpreter='latex');
xlabel('Horizontal coordinate, $y$ (m)');
ylabel('Vertical velocity, $u$ (m/s)');
title('Steady-State $u(y)$ Profiles at Selected Flow Distances', Interpreter='latex');
grid on;

figure;
plot(y, T_new(i_profiles(1),:), '-o', ...
     y, T_new(i_profiles(2),:), '-s', ...
     y, T_new(i_profiles(3),:), '-^', 'LineWidth',1.2);
legend(['Flow distance = ', num2str(X_targets(1)*1e3), ' mm'], ...
       ['Flow distance = ', num2str(X_targets(2)*1e3), ' mm'], ...
       ['Flow distance = ', num2str(X_targets(3)*1e3), ' mm'], ...
       'Location', 'Best');
xlabel('Horizontal coordinate, $y$ (m)');
ylabel('Temperature');
title('Steady-State $T(y)$ Profiles at Selected Flow Distances', Interpreter='latex');
grid on;

% Leading edge effect and 99% SS time
tol_UE = 1e-2;
idx_UE = find( abs(u_inlet_vals) > tol_UE, 1, 'first' );
if ~isempty(idx_UE)
    t_UE = time_vals(idx_UE);
    fprintf('Leading-edge effect (velocity variation at x = 100 mm, j = 2) reached at t =
        %.4f s\n', t_UE);
else
    fprintf('Leading-edge effect (velocity variation) did not reach x = 100 mm within the
        simulation time.\n');
end
```

```

Tsteady = Tsurf_vals(n);
T_target = Tinf + 0.99 * (Tsteady - Tinf);
idx_99 = find( Tsurf_vals >= T_target, 1, 'first' );
if ~isempty(idx_99)
    t_99 = time_vals(idx_99);
    fprintf('Flow reaches 99%% of the steady-state surface temperature at t = %.4f s\n',
        t_99);
else
    fprintf('Flow did not reach 99%% of the steady-state surface temperature within the
        simulation time.\n');
end

% Plot u_inlet_vals as a function of time
figure;
plot(time_vals(1:n), u_inlet_vals(1:n), 'LineWidth', 1.5);
xlabel('Time (s)');
ylabel('Outlet Velocity, $u$ (m/s)', Interpreter='latex');
title('Outlet Velocity vs. Time', Interpreter='latex');
grid on;

% Plot the steady-state variation of temp. diff.
figure;
plot(x, T_new(:,1) - Tinf, 'LineWidth', 1.5);
xlabel('Vertical coordinate, $x$ (m)', Interpreter='latex');
ylabel('Temperature Difference, $T_w - T_{\infty}$', Interpreter='latex');
title('Steady-State Variation of $T_w - T_{\infty}$ vs. $x$', Interpreter='latex');
grid on;
```

5.2 Task II Code

```
clear; clc; close all;

%% Flow Conditions and Geometric Parameters
Pr = 0.733;          % Prandtl number
g = 9.81;            % gravity
beta = 0.0033;       % thermal expansion coeff,
nu = 1.5e-5;         % kinematic vis.
k_air = 0.026;       % thermal cond. of air
T_inf = 30;          % ambient temp.

% Boundary Conditions
F0 = 0; Fp0 = 0; Hp0 = -1; etaMax = 10;

% Guess and find the correct F''(0) and H(0)
guess = [0.80893; 1.47981];
options = optimset('TolFun',1e-5,'TolX',1e-5);
sol = fsolve(@(soln) res(soln,Pr,Hp0,etaMax), guess, options);
Fpp0_sol = sol(1); H0_sol = sol(2);

%% 4th Order Runge-Kutta Method
Fpp0 = Fpp0_sol; H0 = H0_sol;

% Calculated constants C2.1 and C2.2
C21 = 3; C22 = 4;

N = 1000; dEta = etaMax/N;

% Arrays for storing solutions
etaVals = zeros(N+1,1);
yF = zeros(N+1,1);
yFp = zeros(N+1,1);
yFpp = zeros(N+1,1);
yH = zeros(N+1,1);
yHp = zeros(N+1,1);

```

```
% Initial values at eta=0
yFpp(1) = Fpp0; yH(1) = H0; yHp(1) = Hp0;

% Define ODE system used in final integration
% F' (dF/deta); F''(dF'/deta); F''' (dF''/deta); H'(dH/deta); H''(dH'/deta)
sys = @(eta, Y) [
    Y(2);
    Y(3);
    C21*(Y(2)^2)-C22*Y(1)*Y(3) - Y(4);
    Y(5);
    Pr*(Y(2)*Y(4) - 4*Y(1)*Y(5))
];

% Runge-Kutta Scheme
for i = 1:N
    eta_i = etaVals(i);
    Yi    = [yF(i); yFp(i); yFpp(i); yH(i); yHp(i)];

    % Advance solution one step from eta to deta
    k1 = sys(eta_i, Yi);
    k2 = sys(eta_i + 0.5*dEta, Yi + 0.5*dEta*k1);
    k3 = sys(eta_i + 0.5*dEta, Yi + 0.5*dEta*k2);
    k4 = sys(eta_i + dEta, Yi + dEta*k3);

    % For each new i=1 to 5, yi at the new eta location is:
    Y_new = Yi + (dEta/6)*(k1 + 2*k2 + 2*k3 + k4);

    etaVals(i+1) = eta_i + dEta;
    yF(i+1) = Y_new(1);
    yFp(i+1) = Y_new(2);
    yFpp(i+1) = Y_new(3);
    yH(i+1) = Y_new(4);
    yHp(i+1) = Y_new(5);
end

% Plot the similarity solution for F'(eta) and H(eta)
figure;
subplot(1,2,1);
plot(etaVals, yFp,'LineWidth',2);
xlabel('$\eta$', Interpreter='latex');
ylabel('$F''(\eta)$', Interpreter='latex');
title('Variation of $F''(\eta)$ vs. $\eta$', Interpreter='latex');
grid on;

subplot(1,2,2);
plot(etaVals, yH,'LineWidth',2);
xlabel('$\eta$', Interpreter='latex');
ylabel('$H(\eta)$', Interpreter='latex');
title('Variation of $H(\eta)$ vs. $\eta$', Interpreter='latex');
grid on;

%% Converting FD to Similarity

% Load files from task 1
q_220 = 220; q_150 = 150;
u_new_220 = load('u_new_220sad.mat');
T_new_220 = load('Tnewfixed.mat');
u_new_150 = load('u_new_150_sad.mat');
T_new_150 = load('Tnew150.mat');

% Extract values from the loaded files
fn = fieldnames(u_new_220);
u_new_220 = u_new_220.(fn{1});
fn = fieldnames(T_new_220);
T_new_220 = T_new_220.(fn{1});

fn = fieldnames(u_new_150);
u_new_150 = u_new_150.(fn{1});
```

```
fn = fieldnames(T_new_150);
T_new_150 = T_new_150.(fn{1});

% FCTS FD Grid over the plate
x_array_mm = linspace(0,100, size(u_new_220,1));
y_array_mm = linspace(0,20,  size(u_new_220,2));

x_array = x_array_mm / 1000;
y_array = y_array_mm / 1000;

% Plot FD data on top of the similarity solution
figure;
subplot(1,2,1); hold on;  % F'(eta)
subplot(1,2,2); hold on;  % H(eta)

for qW = [150, 220]
    switch qW
        case 150
            Ufield = u_new_150;
            Tfield = T_new_150;
        otherwise
            Ufield = u_new_220;
            Tfield = T_new_220;
    end

    % Loop over the chosen x-coordinates
    for selected_x = [30, 60, 100]
        % Find index in x_array that is closest to selected x
        [~, iX] = min(abs(x_array_mm - selected_x));
        xVal_m = x_array(iX);
        xi = ((g * beta * qW) / (5 * k_air * nu^2))^(1/5);
        eta_fd = zeros(size(y_array));
        Fp_fd = zeros(size(y_array));
        H_fd = zeros(size(y_array));

        for j = 1:length(y_array)
            yVal_m = y_array(j);
            eta_fd(j) = (yVal_m / xVal_m^(0.2)) * xi;
            uVal = Ufield(iX, j);
            Fp_fd(j) = (uVal / (5 * nu * xi^2 * xVal_m^(3/5)));
            TVal = Tfield(iX, j);
            H_fd(j) = (TVal - T_inf)/((qW/k_air)*xVal_m^(1/5)*(xi^(-1)));
        end

        subplot(1,2,1);
        plot(eta_fd, Fp_fd, 'x', 'DisplayName', sprintf('$q=$%d, $x=$%.0f$mm$', qW,
            selected_x));

        subplot(1,2,2);
        plot(eta_fd, H_fd, 'x', 'DisplayName', sprintf('$q=$%d, $x=$%.0f$mm$', qW,
            selected_x));
    end
end

subplot(1,2,1);
plot(etaVals, yFp, 'LineWidth', 2, ...
     'DisplayName', 'RK4 $(F''(\eta))$', 'Color', 'b');
xlabel('$\eta$', Interpreter='latex');
ylabel('$F''(\eta)$', Interpreter='latex');
title('$F''(\eta)$ vs. Dimensionless Velocity', Interpreter='latex');
legend('Location','best', Interpreter='latex');
grid on;

subplot(1,2,2);
plot(etaVals, yH, 'LineWidth', 2, ...
     'DisplayName', 'RK4 $(H(\eta))$', 'Color', 'b');
xlabel('$\eta$', Interpreter='latex');
ylabel('$H(\eta)$', Interpreter='latex');
```

```
title('$H(\eta)$ vs. Dimensionless Temperature', Interpreter='latex');
legend('Location','best', Interpreter='latex');
grid on;

%% Plot Temperature Difference

qW_220 = 220;
Twall_FD = T_new_220(:, 1);
Tw_minus_FD = Twall_FD - T_inf;

figure;
set(gca, "YScale", "log", "XScale", "log");
loglog(x_array_mm, Tw_minus_FD, 'LineWidth',1.5, 'DisplayName','Finite Difference');
hold on;

H0_val = H0_sol;
n = 100;
xSimulated = linspace(0.001, 0.1, n);
Tw_sim = zeros(n,1);

for i=1:n
    x_ = xSimulated(i);
    xi = ((g * beta * qW) / (5 * k_air * nu^2))^(1/5);
    Tw_sim(i) = ((qW_220 * x_^(1/5) * xi^(-1))/k_air) * H0_val;
end

loglog(xSimulated*1000, Tw_sim, 'LineWidth',1.5, 'DisplayName','Similarity');
hold off; grid on;
xlabel('$x$ (mm)', Interpreter='latex');
ylabel('$T_w - T_\infty$', Interpreter='latex');
title('log-log Variation of $T_w-T_{\infty}$ vs. $x$ ($q_w=220$)', Interpreter='latex');
legend('Location','best');

function dY = Reformulate(Y, Pr)
    % State variables
    F = Y(1); Fp = Y(2); Fpp = Y(3);
    H = Y(4); Hp = Y(5);

    % Calculated constants C2.1 and C2.2
    C21 = 3; C22 = 4;

    % Recast state variables for state-space formulation
    dFdx = Fp;
    dFpdx = Fpp;
    dFppdx = C21 * Fp^2 - C22 * F * Fpp - H;
    dHdx = Hp;
    dHpdx = Pr * (Fp * H - 4 * F * Hp);

    % Return a column vector of derivatives
    dY = [dFdx; dFpdx; dFppdx; dHdx; dHpdx];
end

function re = res(sol, Pr, Hp0, etaMax)
    dEtaS = etaMax / 10000;
    Y = [0; 0; sol(1); sol(2); Hp0];

    for i=1:10000
        k1 = Reformulate(Y, Pr);
        k2 = Reformulate(Y + 0.5 * dEtaS * k1, Pr);
        k3 = Reformulate(Y + 0.5 * dEtaS * k2, Pr);
        k4 = Reformulate(Y + dEtaS * k3, Pr);
        Y  = Y + (dEtaS/6)*(k1 + 2 * k2 + 2 * k3 + k4);
    end
    re = [Y(2); Y(4)];
end
```

5.3 Task III Code

```
%% Flow Conditions from Task I and Parameters
g = 9.8;                        % gravity
beta = 0.0033;                  % thermal expansion coeff.
nu = 1.613e-5;                  % kinematic vis.
alpha = 2.2e-5;                 % thermal diff.
k = 0.0261;                     % thermal cond. of air
T_inf = 30;                     % ambient temp.
q_w = 220;                      % surface heat flux

% Stream wise locations
x1 = 0.03; x2 = 0.06;

%% Integral Solutions
% Exponents and constants for reformulated balance equations
n = 0.2; m = 0.6;
B = ((72*alpha*k*(4*alpha+5*nu))/(g*beta*q_w))^0.2;
A = 60*alpha*(((g*beta*q_w)/(72*alpha*k*(4*alpha+5*nu)))^0.4);

% Boundary Layer Thickness at specified x
delta1 = B*x1^n; delta2 = B*x2^n;

% Similarity variables and other constants
xi = ((g*beta*q_w)/(5*k*nu^2))^(1/5);
eta = @(y, x) xi * y ./ (x^(1/5));

% Integral solution expressions for velocity and temperature at x's
u_int_03 = @(y) A*x1^m .* (y/delta1) .* (1 - (y/delta1)).^2;
u_int_06 = @(y) A*x2^m .* (y/delta2) .* (1 - (y/delta2)).^2;
T_int_03 = @(y) T_inf + (q_w/(2*k)) * B*x1^n .* (1 - (y/delta1)).^2;
T_int_06 = @(y) T_inf + (q_w/(2*k)) * B*x2^n .* (1 - (y/delta2)).^2;

%% Plot the Integral Solutions versus eta

% Set all the graphs to LaTex
set(groot,'defaulttextinterpreter','latex');
set(groot, 'defaultAxesTickLabelInterpreter','latex');
set(groot, 'defaultLegendInterpreter','latex');

% For x = 0.03 m
y_vals_03 = linspace(0, delta1, 200);
eta_vals_03 = eta(y_vals_03, x1);

% For x = 0.06 m
y_vals_06 = linspace(0, delta2, 200);
eta_vals_06 = eta(y_vals_06, x2);

figure;
subplot(2,1,1);
plot(eta_vals_03, u_int_03(y_vals_03), 'LineWidth',1.5);
hold on;
plot(eta_vals_06, u_int_06(y_vals_06), 'LineWidth',1.5);
xlabel('$\eta$', 'Interpreter','latex');
ylabel('$u(x,y)$', 'Interpreter','latex');
title('Integral Solution for $u$ vs. $\eta$', 'Interpreter','latex');
legend('$x = 30$ mm','$x = 60$ mm', 'Location','best');
grid on;

subplot(2,1,2);
plot(eta_vals_03, T_int_03(y_vals_03), 'LineWidth',1.5);
hold on;
plot(eta_vals_06, T_int_06(y_vals_06), 'LineWidth',1.5);
xlabel('$\eta$', 'Interpreter','latex');
ylabel('$T(x,y)$', 'Interpreter','latex');
title('Integral Solution for $T$ vs. $\eta$', 'Interpreter','latex');
legend('$x = 30$ mm','$x = 60$ mm', 'Location','best');
grid on;
```

```
67
68  %% Similarity Variables and Transformations
69
70  % Similarity transformation for velocity:  F'(eta)
71  F_transform = @(u, x) u ./ (5*nu*xi^2*(x^(0.6)));
72
73  % Similarity transformation for temperature: H(eta)
74  H_transform = @(T, x) (T - T_inf) ./ ((q_w/k) .* (x^(0.2)/xi));
75
76  % For x = 0.03 m
77  u_vals_03 = u_int_03(y_vals_03);
78  T_vals_03 = T_int_03(y_vals_03);
79  Fprime_int_03 = F_transform(u_vals_03, x1);
80  H_int_03 = H_transform(T_vals_03, x1);
81
82  % For x = 0.06 m
83  u_vals_06 = u_int_06(y_vals_06);
84  T_vals_06 = T_int_06(y_vals_06);
85  Fprime_int_06 = F_transform(u_vals_06, x2);
86  H_int_06 = H_transform(T_vals_06, x2);
87
88  %% Plot the Converted Integral Similarity Profiles for F' and H
89  figure;
90  subplot(2,1,1);
91  plot(eta_vals_03, Fprime_int_03, 'LineWidth', 1.5, LineStyle='-');
92  hold on;
93  plot(eta_vals_06, Fprime_int_06, 'LineWidth', 1.5, LineStyle='--');
94  xlabel('$\eta$','Interpreter','latex');
95  ylabel('$F''(\eta)$','Interpreter','latex');
96  title('Converted Integral Solution for $F''(\eta)$','Interpreter','latex');
97  legend('$x = 30$ mm','$x = 60$ mm','Location','best');
98  grid on;
99
100 subplot(2,1,2);
101 plot(eta_vals_03, H_int_03, 'LineWidth', 1.5, LineStyle='-');
102 hold on;
103 plot(eta_vals_06, H_int_06, 'LineWidth', 1.5, LineStyle='--');
104 xlabel('$\eta$','Interpreter','latex');
105 ylabel('$H(\eta)$','Interpreter','latex');
106 title('Converted Integral Solution for $H(\eta)$','Interpreter','latex');
107 legend('$x = 30$ mm','$x = 60$ mm','Location','best');
108 grid on;
109
110 %% Plot Variation of Wall Temperature vs. x
111 % Integral solution at y = 0:
112 T_wall_int = @(x) T_inf + (q_w*B*x.^n)/(2*k);
113
114 % For the similarity solution, at y = 0 (   = 0), we have:
115 H_wall = H_int_03(1);  % extract first value of H_int_03
116 T_wall_sim = @(x) T_inf + (q_w/k) .* (x.^(0.2)/xi)*H_wall;
117
118 x_vals = linspace(0, 0.1, 100);
119 T_wall_int_vals = T_wall_int(x_vals);
120 T_wall_sim_vals = T_wall_sim(x_vals);
121
122 figure;
123 plot(x_vals, T_wall_int_vals, 'LineWidth', 1.5, LineStyle='-');
124 hold on;
125 plot(x_vals, T_wall_sim_vals, 'LineWidth', 1.5, LineStyle='--');
126 xlabel('$x$ (m)','Interpreter','latex'); ylabel('$T_{wall}$','Interpreter','latex');
127 title('Variation of Wall Temperature $H(0)$ vs. $x$','Interpreter','latex');
128 legend('Integral','Similarity','Location','best');
129 grid on;
```

5.4 Task IV Code

5.4.1 Bisection Method for Free Convection

```
g = 9.8;                % gravity
beta = 0.0033;          % thermal expansion coeff.
nu = 1.613e-5;          % kinematic vis.
alpha = 2.200e-5;       % thermal diff.
k = 0.0261;             % thermal cond. of air
Pr = nu / alpha;        % Prandtl number
L = 0.03;               % plate height
q_target = 220;         % target heat flux
T_infinity = 30;        % ambient temp.

% Predicted heat flux
q_pred = @(DeltaT) (k/L)*0.59 * (Pr * g * beta * DeltaT * L^3 / (nu^2))^(1/4) * DeltaT;

% Tolerance value
tol = 0.01;

low  = 0;        % lower bound guess for Delta T
high = 100;      % upper bound guess for Delta T
numIter = 100;   % just a safety max number of iterations

for i = 1:numIter

    % Calculate midpoint and use to evalute heat flux
    midDT = 0.5*(low + high);
    q_mid = q_pred(midDT);

    % Calculate the l1 norm
    if abs(q_mid - q_target) <= tol
        fprintf('Converged after %d iterations\n', i);
        fprintf('DeltaT = %.2f K,  q_pred = %.2f W/m^2\n', midDT, q_mid);
        break
    end

    q_low = q_pred(low);

    % If overshot, lower deltaT
    if (q_mid > q_target)
        high = midDT;
    else
        low = midDT;
    end
end
```

5.4.2 FTCS FD Approach with Radiative Heat Exchange

```
clear; clc; close all;

%% Flow Conditions and Geometric Parameters
g = 9.81;            % gravity
beta = 3.3e-3;       % thermal expansion coeff.
nu = 1.5e-5;         % kinematic vis.
alpha = 2.2e-5;      % thermal diff.
k_air = 0.0261;      % thermal cond. of air
Tinf = 30;           % ambient temp.
sigma = 5.67e-8;     % Stefan-Boltzmann
emis  = 0.85;        % emissivity of surface

% Mesh Grid Dimensions
Nx = 51;             % points in vertical (x) direction
Ny = 51;             % points in horizontal (y) direction

% Grid spacings, simulation time, and tolerance
```

```
dx = 2.0e-3;          % vertical grid spacing
dy = 0.4e-3;          % horizontal grid spacing
dt = 5.0e-4;          % time step
maxTime = 500;        % time limit
Tol = 1.0e-4;         % convergence tolerance

% Maximum allowable surface temp.
T_max_allowed = 70;
% List query heat flux points (adjust as needed)
q_test = 25:1:30;

% Preallocate storage for results
nTest = length(q_test);
Tsteady_vals = nan(nTest, 1);
finalTime_vals = nan(nTest, 1);
converged = false(nTest, 1);

% Loop through different surface generation q's
for idx = 1:nTest
    q_w = q_test(idx);
    fprintf('For q_w = %g W we get a ', q_w);
    % Run the simulation with the given q_w
    [convFlag, Tsteady, finalTime, T_new, q_rad_avg] = task1(q_w, g, beta, nu, alpha, k_air,
        Tinf, sigma, emis, Nx, Ny, dx, dy, dt, maxTime, Tol);

    converged(idx) = convFlag;
    Tsteady_vals(idx) = Tsteady;
    finalTime_vals(idx) = finalTime;

    fprintf('steady state surface temp T_s = %.2f  C \n', Tsteady);

    % Stop if the surface temperature exceeds the allowable maximum
    if Tsteady > T_max_allowed
        fprintf('Surface temperature exceeds allowable maximum (%.2f  C).\n', T_max_allowed)
            ;
        break;
    end
end
% Determine the maximum q_w that produced Tsteady below the allowable limit
validIdx = find(Tsteady_vals <= T_max_allowed & ~isnan(Tsteady_vals));
if isempty(validIdx)
    fprintf(['No q_w value yielded a steady ' ...
        'state below T_max_allowed.\n']);
else
    q_max = q_test(validIdx(end));
    T_max_steady = Tsteady_vals(validIdx(end));
end

%max heat rejection rate
q_rad_vals = zeros(Nx,1);
q_total_vals = zeros(Nx,1);

for i = 1:Nx
    % Use the temperature at the adjacent node (j=2)
    T_adj_K = T_new(i,2) + 273.15;   % convert to Kelvin
    Tinf_K  = Tinf + 273.15;
    % Radiative heat flux at the wall [W/m^2]
    q_rad_vals(i) = emis * sigma * (T_adj_K^4 - Tinf_K^4);
    % Total heat rejection = imposed conduction flux + radiative flux.
    q_total_vals(i) = q_w + q_rad_vals(i);
    q = mean(q_total_vals);
end
    fprintf(['\nMaximum allowable heat generation that ' ...
        'can be rejected = %.2f W\n'], q*Ny*Nx*dy*dx);
    fprintf(['Steady-state surface temperature at ' ...
        'this flux = %.2f  C \n'], T_max_steady);

%% TASK I CODE
```

```
function [converged, Tsteady, final_time, T_new, q_rad_avg] = task1(q_w, g, beta, nu, alpha,
     k_air, Tinf, sigma, emis, Nx, Ny, dx, dy, dt, maxTime, steadyTol)
    % Determine number of time steps.

    numSteps = ceil(maxTime/dt);

    %% Array Initialization
    u_old = zeros(Nx, Ny);            % u: vertical velocity component
    v_old = zeros(Nx, Ny);            % v: horizontal velocity component
    T_old = Tinf * ones(Nx, Ny);      % temperature field

    % Fields for next time step
    u_new = u_old;
    v_new = v_old;
    T_new = T_old;

    % Preallocate arrays to record some representative values (if desired)
    Tsurf_vals = zeros(1, numSteps);
    converged = false;
    final_time = 0;

    %% Time Loop for FTCS
    for n = 1:numSteps
        % Ensure local scope
        u = u_old;
        v = v_old;
        T = T_old;

        % Update temperature in interior
        for i = (Nx-1):-1:2
            for j = 2:(Ny-1)

                % Upwind differencing in x
                if u(i,j) >= 0
                    dTdx = (T(i,j) - T(i-1,j)) / dx;
                else
                    dTdx = (T(i+1,j) - T(i,j)) / dx;
                end

                % Upwind differencing in y
                if v(i,j) >= 0
                    dTdy = (T(i,j) - T(i,j-1)) / dy;
                else
                    dTdy = (T(i,j+1) - T(i,j)) / dy;
                end

                convT = - ( u(i,j)*dTdx + v(i,j)*dTdy );

                % Diffusion term
                diffT_x = alpha * ( T(i+1,j) - 2*T(i,j) + T(i-1,j) ) / (dx^2);
                diffT_y = alpha * ( T(i,j+1) - 2*T(i,j) + T(i,j-1) ) / (dy^2);

                % New temperature
                T_new(i,j) = T(i,j) + dt*( convT + diffT_x + diffT_y );
            end
        end

        % Update vertical velocity in interior
        for i = (Nx-1):-1:2
            for j = 2:(Ny-1)
                % Upwind differencing for u in x
                if u(i,j) >= 0
                    dudx = (u(i,j) - u(i-1,j)) / dx;
                else
                    dudx = (u(i+1,j) - u(i,j)) / dx;
                end

                % Upwind differencing in y
```

```
                if v(i,j) >= 0
                    dudy = (u(i,j) - u(i,j-1)) / dy;
                else
                    dudy = (u(i,j+1) - u(i,j)) / dy;
                end

                convU = - ( u(i,j)*dudx + v(i,j)*dudy );

                % Diffusion term
                diffU_x = nu * ( u(i+1,j) - 2*u(i,j) + u(i-1,j) ) / (dx^2);
                diffU_y = nu * ( u(i,j+1) - 2*u(i,j) + u(i,j-1) ) / (dy^2);

                % Bouyancy term
                buoy  = g * beta * ( T(i,j) - Tinf );

                % New x-direction velocity
                u_new(i,j) = u(i,j) + dt*( convU + diffU_x + diffU_y + buoy );
            end
        end

        % Update horizontal velocity v from continuity
        for i = (Nx-1):-1:2
            for j = 2:(Ny-1)
                du_dx = (u(i,j) - u(i-1,j)) / dx;
                dv_dy = (v(i,j) - v(i,j-1)) / dy;
                v_new(i,j) = v(i,j) - dt*( du_dx + dv_dy );
            end
        end

        %% BOUNDARY CONDITIONS
        % Heated left wall
        for i = 1:Nx
            u_new(i,1) = 0;
            v_new(i,1) = 0;
            % Combine conduction and radiation
            radiation = (sigma*emis/k_air)*((T_new(i,2)+273.15)^4 - (Tinf+273.15)^4);
            conduction = (q_w / k_air);
            T_new(i,1) = T_new(i,2) + dy*(conduction + radiation);
        end

        % Right boundary = zero-gradient
        for i = 1:Nx
            T_new(i,Ny) = T_new(i,Ny-1);
            u_new(i,Ny) = 0;
            v_new(i,Ny) = v_new(i,Ny-1);
        end

        % Bottom boundary = zero-gradient and T=Tinf
        for j = 1:Ny
            T_new(1,j) = Tinf;
            u_new(1,j) = u_new(2,j);
            v_new(1,j) = v_new(2,j);
        end

        % Top boundary = zero-gradient
        for j = 1:Ny
            T_new(Nx,j) = T_new(Nx-1,j);
            u_new(Nx,j) = u_new(Nx-1,j);
            v_new(Nx,j) = v_new(Nx-1,j);
        end

        % Record surface temperature at heated wall
        Tsurf_vals(n) = T_new(Nx,1);
        final_time = (n-1)*dt;

        % Check convergence
        diffU = max(abs(u_new(:) - u_old(:)));
        diffV = max(abs(v_new(:) - v_old(:)));
```

```
        diffT = max(abs(T_new(:) - T_old(:)));
        maxChange = max([diffU, diffV, diffT]);

        % Check if steady-state is reached
        if maxChange < steadyTol
            converged = true;
            break;
        end

        % Update fields for the next iteration
        u_old = u_new;
        v_old = v_new;
        T_old = T_new;
    end

    % Use the steady-state surface temperature at the heated wall
    Tsteady = Tsurf_vals(n);

    % Calculate the local radiative rejection along the wall (using adjacent node j=2)
    q_rad_vals = zeros(Nx,1);
    for i = 1:Nx
        T_adj_K = T_new(i,2) + 273.15;
        % Change temp to Kelvin
        Tinf_K  = Tinf + 273.15;
        q_rad_vals(i) = emis * sigma * (T_adj_K^4 - Tinf_K^4);
        q_rad_avg = mean(q_rad_vals);
    end
end
```