A Computational Model for TEACHING FREE CONVECTION

AARON S. GOLDSTEIN
Virginia Polytechnic Institute • Blacksburg, VA 24061-0211

Convective transport phenomena are fundamental principles of chemical engineering and are covered in both graduate- and undergraduate-level courses, but the transport equations are frequently coupled and cannot be solved analytically. Consequently, classroom teaching of this topic is usually fragmented; the fundamental equations are developed and the semi-empirical transport correlations are applied, but methods to solve the fundamental equations and arrive at the Nusselt and Sherwood numbers are usually neglected.

With the ubiquity of fast and inexpensive computers and easy-to-implement software packages (*e.g.,* Matlab, Visual Basic), it has become tractable to fill this gap. Implementation of computational methods in engineering education remains limited, however.[1] This deficit can be alleviated by the incorporation of numerical methods into the curriculum and increased availability of pertinent, ready-to-use code.

Free convection near a vertical wall is a classic example of convective transport and involves simultaneously solving the Navier-Stokes, heat, and continuity equations. Pohlhausen showed that this problem could be approximated by coupled second- and third-order ordinary differential equations (ODEs) with respect to a similarity variable. Importantly, this approximation can be readily solved using ODE solvers built into common software packages (*e.g.,* Matlab, Mathematica, Polymath).

This example has been used in two separate courses taught at Virginia Tech. In a graduate-level transport phenomena course, students employed Euler's method to solve the coupled ODEs and match the boundary conditions. The objective was for the students to reproduce Figure 12-5 and 12-6 from Deen.[2] In an undergraduate numerical methods course, students were provided with the complete and working code as part of an interactive laboratory exercise. The goal was to introduce the students to the use of an ODE solver and the shooting method to solve boundary value problems.

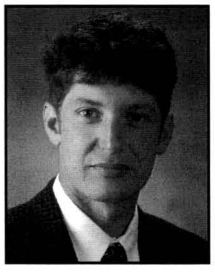
The example will be presented in four steps:

1. *Brief derivations of the coupled second- and third-order ODEs from the fundamental transport equations are presented.*
2. *These two equations are solved as an initial value problem consisting of five coupled first-order ODEs.*
3. *A shooting method algorithm is presented that employs the Newton's method to iteratively find initial conditions that satisfy conditions far from the wall.*
4. *The solution is used to predict the average Nusselt number and temperature and velocity distributions near the wall.*

The solution is demonstrated using Matlab (version 6), but other programming languages can be used. This step-wise approach is intended to aid student learning by breaking the full problem into discrete modules that implement different numerical methods and programming structures. In addition, the graphical display of predictions in dimensional form is intended to aid visualization of fluid mechanics. The learning objectives that are illustrated in this example can be subdivided into categories of transport phenomena, numerical methods, and Matlab implementation (see Table 1).

DESCRIPTION OF THE PROBLEM

A fluid (*e.g.,* air) of density ρ, viscosity μ, heat capacity C_p, thermal conductivity k, and coefficient of thermal expansivity β, is in contact with a vertical wall (see Figure 1). If the surface temperature of the wall, T_w, is greater than



Aaron S. Goldstein *is Assistant Professor in the Department of Chemical Engineering at Virginia Polytechnic Institute and State University and a faculty member of the Wake Forest/Virginia Tech School of Biomedical Engineering and Sciences. He received his doctorate in chemical engineering and bioengineering at Carnegie Mellon University in 1997. His research interests include biomaterials, interfacial phenomena, and transport phenomena as they relate to tissue engineering.*

the temperature of the fluid far from the wall, T_o, then thermal expansion of the fluid near the wall will lead to buoyancy-driven flow (*i.e.,* free convection) in response to gravity, g. In this two-dimensional geometry the x-axis (defined as parallel to the surface) is oriented vertically, and the y-axis is oriented horizontally. For this two-dimensional system, the relevant transport equations are continuity

$$\frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} = 0 \tag{1}$$

the Navier-Stokes equation for flow in the x-direction (parallel to the surface

$$\rho\left(v_x \frac{\partial v_x}{\partial x} + v_y \frac{\partial v_x}{\partial y}\right) = \frac{dP}{dx} - \beta(T - T_o)\rho g - \mu\left(\frac{\partial^2 v_x}{\partial x^2} + \frac{\partial^2 v_x}{\partial y^2}\right) \tag{2}$$

and the heat equation

TABLE 1
Learning Objectives
Relevant to the Free Convection Example

Learning Objectives Relevant to Transport Phenomena

- Understand the transport equations that describe free convection
- Understand the boundary conditions relevant to free convection near a vertical wall
- Visualize and evaluate the 2D temperature and velocity profiles predicted from theory

Learning Objectives Relevant to Numerical Methods

- Set up coupled ordinary differential equations and solve them as an initial value problem
- Employ secant method to iteratively solve multiple nonlinear equations
- Employ secant method to solve coupled ordinary differential equations as a boundary value problem

Learning Objectives Relevant to the Use of Matlab

- Write and use an m-file as a function
- Use ODE45 to solve a set of coupled first-order ordinary differential equations
- Implement matrix operations: inverse, transpose, multiplication
- Graph data using "plot" and "contour" commands

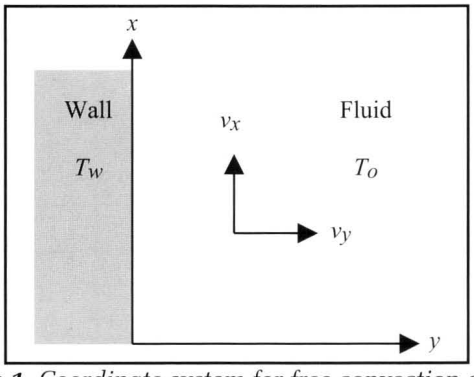


Figure 1. *Coordinate system for free convection of a fluid at T_o near a heated wall at T_w.*

$$\rho C_p\left(v_x \frac{\partial T}{\partial x} + v_y \frac{\partial T}{\partial y}\right) = k\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right) \tag{3}$$

These equations have four dependent variables (T, P, v_x, and v_y) and two independent variables (x and y). From visual inspection, we see that ten boundary conditions are required to completely solve them. Although they can be solved using finite element or finite difference numerical approaches, an approximation of the solution can be obtained by first simplifying these equations to a pair of coupled ordinary differential equations.

The simplification scheme (described in detail by Deen, pp. 493-501[2]) is given here briefly. First, these mathematical expressions may be simplified by neglecting pressure drop, dP/dx, and by recognizing that diffusive transport in the x direction, $\partial^2 v_x / \partial x^2$ and $\partial^2 T / \partial x^2$, is negligible relative to convective transport. This eliminates the dependent variable P and the need for three boundary conditions. Second, a stream function, which automatically satisfies Eq. (1)

$$v_x = \frac{\partial \psi}{\partial y} \qquad v_y = -\frac{\partial \psi}{\partial x} \tag{4}$$

is used to eliminate one equation and one dependent variable. Third, a similarity variable originally proposed by Pohlhausen, $\eta \propto y / x^{0.25}$, is used to convert the partial differential equations into ordinary differential equations. Fourth, the remaining dependent and independent variables are nondimensionalized and scaled to achieve the follow two equations:

$$\frac{d^3F}{d\eta^3} + 3F\frac{d^2F}{d\eta^2} - 2\left(\frac{dF}{d\eta}\right)^2 + \Theta = 0 \tag{5a}$$

$$\frac{d^2\Theta}{d\eta^2} + 3\,\mathrm{Pr}\,F\frac{d\Theta}{d\eta} = 0 \tag{5b}$$

Here the variables are

$$F = \frac{\psi}{\nu\left(4^3 \mathrm{Gr}_x\right)^{0.25}} \tag{6a}$$

$$\Theta = \frac{T - T_o}{T_w - T_o} \tag{6b}$$

$$\eta = \left(\mathrm{Gr}_x / 4\right)^{0.25}(y / x) \tag{6c}$$

the dimensionless groups are

$$\text{Grashoff number:} \quad \mathrm{Gr}_x = g\nu^2 x^3 \beta\left(T_w - T_o\right) \tag{6d}$$

$$\text{Prandtl number:} \quad \mathrm{Pr} = \mu C_p / k \tag{6e}$$

and kinematic viscosity is $\nu = \mu/\rho$. Equations (5a) and (5b) must be solved subject to five boundary conditions

$$F(\eta = 0) = \left.\frac{dF}{d\eta}\right|_{\eta=0} = \left.\frac{dF}{d\eta}\right|_{\eta\to\infty} = 0$$

$$\Theta(\eta \to \infty) = 0 \quad \text{and} \quad \Theta(\eta = 0) = 1 \tag{7}$$

Note that because Θ and $dF/d\eta$ must asymptotically approach zero as η goes to infinity, their derivatives, $d^2F/d\eta^2$ and $d\Theta/d\eta$, also must go to zero.

SOLVING THE INITIAL VALUE PROBLEM

To solve Eqs. (5a) and (5b) numerically, they are first reduced to sets of first-order equations by defining new dependent variables F^0, F^1, F^2, Θ^0, and Θ^1 [3, p.671]

$$\begin{aligned}
\frac{dF^0}{d\eta} &= F^1 \\
\frac{dF^1}{d\eta} &= F^2 \\
\frac{dF^2}{d\eta} &= -3F^0F^2 + 2\left(F^1\right)^2 - \Theta^0 \\
\frac{d\Theta^0}{d\eta} &= \Theta^1 \\
\frac{d\Theta^1}{d\eta} &= -3\,\mathrm{Pr}\,F^0\Theta^1
\end{aligned} \tag{8}$$

Here F^0 and Θ^0 are equivalent to F and Θ in Eq. (5), and F^1, F^2, and Θ^1 are first and second derivatives of F and Θ with respect to η. Note that Eq. (5a) is third order and is replaced with three first-order equations, whereas Eq. (5b) is second order and is replaced with two equations.

The five coupled first-order ODEs can be readily solved in Matlab using the built-in ODE45 solver. The ODEs are defined within a Matlab m-file "freeconvect.m" (see Table 2) where the function "Y=freeconvect(eta,X,Pr)" calculates the set of first derivatives

$$Y = \begin{pmatrix} dF^0/d\eta \\ dF^1/d\eta \\ dF^2/d\eta \\ d\Theta^0/d\eta \\ d\Theta^1/d\eta \end{pmatrix} \tag{9}$$

subject to three arguments: 1) the independent variable, eta, 2) the set of dependent variables, X (at eta)

$$X = \begin{pmatrix} F^0(\eta) \\ F^1(\eta) \\ F^2(\eta) \\ \Theta^0(\eta) \\ \Theta^1(\eta) \end{pmatrix} \tag{10}$$

and 3) the Prandtl number, Pr. For this example, Pr = 0.72 (for air) is used. (Details regarding built-in ODE solvers for Matlab version 6 can be found in Higham and Higham[4], pp. 148-163.)

A driver program (see Table 3) is used to solve the five coupled ODEs subject to a set of five initial conditions, "Xinit," over a finite range of η, "etaspan." In this case, the finite range chosen is $0 \le \eta \le 10$. From Eq. (7), only three initial conditions (at $\eta = 0$) are known: F^0=0, F^1=0. and Θ^0=1. The remaining initial conditions must be guessed. Because F^2 and Θ^1 are proportional to the velocity gradient, dv_x/dy, and temperature gradient, dT/dy, respectively, F^2 should be positive and Θ^1 negative at $\eta = 0$. For the case of Pr = 0.72, the initial values are $F^2 = 0.6761$ and $\Theta^1 = -0.5047$ and produce the solid curves in Figure 2. In principle, a trial-and-error method can be employed to determine these initial values, but it is tedious. Alternatively, an algorithm to solve two coupled nonlinear equations, such as Newton's method, can be devised to iteratively solve the boundary value problem.

SOLVING TWO COUPLED NONLINEAR EQUATIONS

Newton's method for solving nonlinear equations involves an iterative process of iteratively refining x, by a correction, h

TABLE 2
Subroutine for ODE45 Solver

```
function Y=freeconvect(eta,X,Pr)
% X=(F0; F1; F2; Theta0; Theta1)
dF0deta=X(2);
df1deta=X(3);
dF2deta=-3*X(1)*X(3)+2*(X(2))^2-X(4);
dTheta0deta=X(5);
dTheta1deta=-3*Pr*X(1)*X(5);
Y=[dF0deta; dF1deta; dF2deta; dTheta0deta;
    dTheta1deta];
```

TABLE 3
Solver Program

```
% ODE45 Solver for freeconvect.m
Pr=0.72; % Prandtl number for air
etaspan=[0 10] ;
Xinit=[0;0;0.6761;1;-0.5047]
options=odeset('AbsTol',1e-7,'RelTol',1e-4);
%
% solver will call function 'freeconvect'
[Eta,X]=ode45(@freeconvect,etaspan,Xinit,options,Pr);
%
% plot results
figure(1), plot(Eta,X(:2),'k-')
xlabel('\it{\eta}','Fontsize',16)
ylabel('{\it{F}}^1','Fontsize',16)
figure(2), plot(Eta,X(:,4),'k-')
xlabel ('\it{\eta}','Fontsize',16)
ylabel ('{\Theta}^0','Fontsize',16)
```

$$x_{i+1} = x_i + h \tag{11}$$

where h is calculated by linear extrapolation of the function, f(x), to zero[3, pp139-145]

$$0 = f(x_i) + (df/dx)_{x_i} h \tag{12}$$

This approach can be scaled up readily to solve the roots of coupled equations. For this particular example, the equations in matrix form are

$$\begin{pmatrix} F^2(\eta=0)_{i+1} \\ \Theta^1(\eta=0)_{i+1} \end{pmatrix} = \begin{pmatrix} F^2(\eta=0)_i \\ \Theta^1(\eta=0)_i \end{pmatrix} + \begin{pmatrix} h_1 \\ h_2 \end{pmatrix} \tag{13}$$

where $(h_1 h_2)^t$ is the solution to the equation

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} = \begin{pmatrix} F^1(\eta=10)_i \\ \Theta^0(\eta=10)_i \end{pmatrix} + \begin{pmatrix} \left.\frac{dF^1(\eta=10)}{dF^2(\eta=0)}\right|_{\Theta^1} & \left.\frac{dF^1(\eta=10)}{d\Theta^1(\eta=0)}\right|_{F^2} \\ \left.\frac{d\Theta^0(\eta=10)}{dF^2(\eta=0)}\right|_{\Theta^1} & \left.\frac{d\Theta^0(\eta=10)}{d\Theta^1(\eta=0)}\right|_{F^2} \end{pmatrix} \begin{pmatrix} h_1 \\ h_2 \end{pmatrix} \tag{14}$$

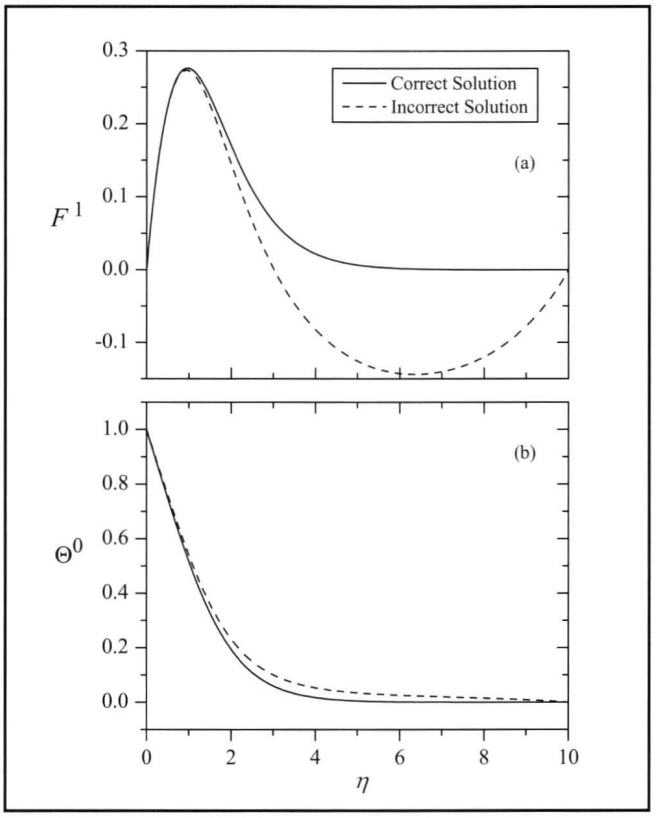


Figure 2. *Solution to dimensionless boundary value problem for Pr = 0.72. Solid line corresponds to the correct solution, which satisfies $F^1 = dF^1/d\eta = \Theta^0 = d\Theta^0/d\eta = 0$. Dashed line is incorrect solution, which only satisfies $F^1 = \Theta^0 = 0$.*

Note that these equations are designed to find initial values of F^2 and Θ^1 (at $\eta = 0$) that satisfy final values of $F^1 = 0$ and $\Theta^0 = 0$ (at $\eta = 10$). Equations (13) and (14) can be expressed using matrix and vector variables as

$$\underline{X}_{i+1} = \underline{X}_i + \underline{H} \tag{15}$$

and

$$0 = \underline{F} + \underline{\underline{K}}\,\underline{H} \tag{16}$$

respectively, and combined to achieve an iterative strategy

$$\underline{X}_{i+1} = \underline{X}_i - \underline{\underline{K}}^{-1}\underline{F} \tag{17}$$

where $\underline{\underline{K}}$ is the Jacobian matrix.

Although the four derivatives that comprise $\underline{\underline{K}}$ cannot be expressed analytically, they can be estimated using two-term forward finite divided differences. The first of the four is approximated by the equation

$$\left.\frac{dF^1(\eta=10)}{dF^2(\eta=0)}\right|_{\Theta^1} \approx \frac{F^1(\eta=10)\big|_{F^2+\delta_F,\Theta^1} - F^1(\eta=10)\big|_{F^2,\Theta^1}}{\delta_F} \tag{18}$$

To estimate the four derivatives, the ODEs in Eq. (8) have to be solved for three pairs of initial values $\{F_i^2, \Theta_i^1\}$, $\{F_i^2 + \delta_F, \Theta_i^1\}$, $\{F_i^2, \Theta_i^1, + \delta_\Theta\}$, where δ_F and δ_Θ are small numbers.

A Matlab code for solving this boundary-value problem is given in Table 4 (next page). Within the iterative loop, three steps are taken. First, the ODEs are solved over the interval $0 \le \eta \le 10$ for three pairs of initial values, $\{F_i^2, \Theta_i^1\}$, $\{F_i^2 + \delta_F, \Theta_I^1\}$, and $\{F_i^2, \Theta_i^1 + \delta_\Theta\}$, where "$\delta_F$=dF2=0.001" and "$\delta_\Theta$=dT1=0.001". The resultant values of F^1 and Θ^0 at η=10 for each pair of initial conditions is placed into an array (X1, X2, and X3). Second, the four derivatives that comprise $\underline{\underline{K}}$ are calculated. Third, new estimates of initial values are calculated using Eq. (17).

These three steps are repeated within a "while" loop that continues until the magnitudes of both corrections fall below 10^{-6}. Once the convergence tolerance is met, the algorithm exits the iterative loop and plots the results. Because the ODE solver evaluates $\{F_i^2, \Theta_i^1\}$ last, upon exiting the loop the matrix X holds the solution for these initial values. The accuracy of the solution can be improved by allowing δ_F and δ_Θ to decrease as the solution converges. In particular, a strategy of setting "dF2=H(1)" and "dT1=H(2)" is analogous to the Secant Method.[3, pp145-150]

When an initial guess of "initF2=1" and "initT1=-0.5" is used, a solution is found in 7 iterations that matches the boundary conditions (solid curves in Figure 2) and agrees closely with the tabulated values reported by Ostrach.[5] Interestingly, when the code is run with initial conditions "initF2=0.5" and

"initT1=-0.5", it converges in 15 iterations to yield the dashed curves in Figure 2. Although this solution also matches the specified boundary conditions at η=10, the derivatives are not asymptotically approaching zero.

INTERPRETING THE RESULTS

Once the unknown initial condition $\Theta^1(\eta=0)$ is determined, it can be used to calculate local and average Nusselt numbers, Nu_x and $\overline{Nu}$, respectively. Here, using the fundamental equality between the bulk heat flux into the fluid and conduction near the wall

$$h(T_w - T_o) = -k\left.\frac{\partial T}{\partial y}\right|_{y=0} \tag{19}$$

the local Nusselt number can be derived in terms of dimensionless groups[5]

$$Nu_x = \frac{hx}{k} = \frac{-x}{(T_w - T_o)}\left.\frac{\partial T}{\partial y}\right|_{y=0} = -x\left.\frac{\partial \Theta}{\partial \eta}\right|_{\eta=0}\frac{\partial \eta}{\partial y} = \left(\frac{Gr_x}{4}\right)^{0.25}\left(-\frac{\partial \Theta}{\partial \eta}\right)_{\eta=0} \tag{20}$$

Likewise, the average Nusselt number can be derived

$$\overline{Nu} = \frac{1}{L}\int_0^L \frac{hL}{k}dx = -\int_0^L \frac{1}{x}\left(\frac{Gr_x}{4}\right)^{0.25}\left.\frac{\partial \Theta}{\partial \eta}\right|_{\eta=0} dx = \frac{4}{3}\left(\frac{Gr_L}{4}\right)^{0.25}\left(-\frac{\partial \Theta}{\partial \eta}\right)_{\eta=0} \tag{21}$$

where Gr_L corresponds to Gr_x at x = L. Using values of $\Theta^1(\eta=0)$ determined for a range of Pr values and Eq. (21), the circles in Figure 3 can be obtained. These predictions show very good agreement with the semi-empirical formula that Le Fevre[6] developed from numerical solutions to Pohlhausen's approximation (solid curve).

$$\overline{Nu} = \left(\frac{Gr_L\, Pr^2}{2.435 + 4.884\, Pr^{0.5} + 4.953\, Pr}\right)^{0.25} \tag{22}$$

In addition, $\overline{Nu}$ can be used to predict average heat flux, $\bar{q}$

$$\bar{q} = \frac{1}{L}\int_0^L h(T_w - T_o)dx = \frac{k}{L}\overline{Nu}(T_w - T_o) = \frac{4k}{3}\left(\frac{\nu^2 g\beta}{4L}\right)^{0.25}(T_w - T_o)^{1.25}\left(-\frac{\partial \Theta}{d\eta}\right)_{\eta=0} \tag{23}$$

TABLE 4
Iterative Solver

```
% Shooting Method (Non-Linear Solver) for
        Boundary Value Problem
Pr=0.72; % Prandtl number for air
dF2=.001; % initial step size
dT1=.001; % initial step size
initF2=1.0; % initial guess
initT1=-0.5; % initial guess
K=zeros(2); % derivatives for Newton's method
etaspan=[0 10]; % solve ODEs over finite range
H=[1;1];
options=odeset('AbsTol',1e-7,'RelTol',1e-4);
%
while max(abs(H))>1e-6
 %
 % evaluate initial value ODEs
 [Eta,X]=ode45(@freeconvect,etaspan,...
   [0;0;initF2=dF2;1;initT1],options,Pr);
 n=size(Eta,1);
 X2=[X(n,2);X(n,4)];
 [Eta,X]=ode45(@freeconvect,etaspan,...
        [0;0;initF2;1;initT1+dT1],options,Pr);
 n=size(Eta,1);
 X3=[X(n,2);X(n,4);
 [Eta,X]=ode45(@freeconvect,etaspan,...
        [0;0;initF2;1;initT1],options,Pr);
 n=size(Eta,1);
 X1=[X(n,2);X(n,4)];
 %
 % estimate derivatives
 K(1,1)=(X2(1)-X1(1))/dF2; % dF1/dF2
 K(2,1)=(X2(2)-X1(2))/dF2; % dT0/dF2
 K(1,2)=(X3(1)-X1(1))/dT1; % dF1/dT2
 K(2,2)=(X3(2)-X1(2))/dT1; % dT0/dT2
 %
 % calculate new initial conditions
 a=cond(K);
 if a>10^6, display('matrix becoming singular')
        break, end
 H=inv(K)*(-X1);
 initF2=initF2+H(1);
 initT1=initT1+H(2);
 fprintf('initF2=%7.6f initT1=%7.6f
        cond(K)=%3.2e\n',initF2,...
      initT1,a)
 %
end
%
% Plot Results
figure(1), plot(Eta,X(:,2),'k-')
xlabel('\it{\eta}','Fontsize',16)
ylabel('{\it{F}}^1','Fontsize',16)
figure(2), plot(Eta,X(:,4),'k-')
xlabel('\it{\eta}','Fontsize',16)
ylabel('{\Theta}^0','Fontsize',16)
```

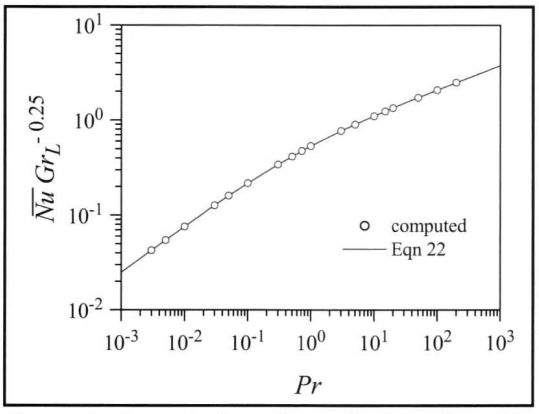


Figure 3. *Average Nusselt number as a function of Prandtl number. Values predicted using this model (circles) are compared to semi-empirical formula, Eq. 22 (solid line).*

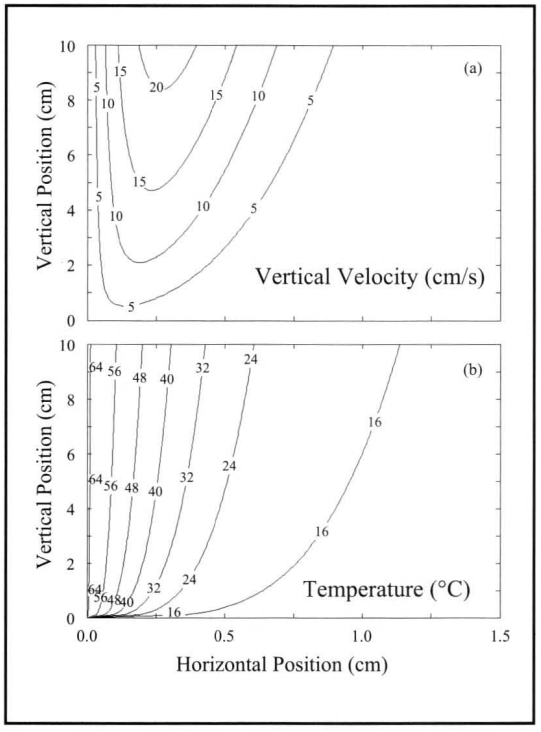


Figure 4. *Contour plots of a) vertical velocity and b) temperature of air (T_o=15°C) near a heated wall (T_w=65°C) as a function of vertical and horizontal position. These plots were generated by dimensionalizing the solution (Figures 2a,b) for fluid properties of air at T_f=40°C.*

Here we see that Pohlhausen's approximation predicts that $\bar{q}$ is proportional to $(T_w - T_o)^{1.25}$.

Although Figure 2 demonstrates that a solution can be found that matches the boundary conditions, and Figure 3 validates the solution, it may be difficult for undergraduate students to conceptualize this abstract solution. Therefore, converting the solution into dimensional form for a familiar fluid may be valuable for student learning. For this example, air of T_o = 15°C near a wall of T_w = 65°C is used because experimental data have been collected by Schmidt and Beckmann.[7] T(x,y) and ν_x(x,y) can be determined using the definitions of the dimensionless groups (Eq. 6), the equation for vertical velocity[2, p.500]

$$\nu_x\left[4^{0.75}\nu(Gr_x)^{0.25}\right]\left(\frac{dF}{d\eta}\right)\left(\frac{\partial\eta}{\partial y}\right) = (4\,\beta g\Delta Tx)^{0.5}\left(\frac{dF}{d\eta}\right) \qquad (24)$$

and fluid properties at the film temperature $[T_f=(T_w+T_o)/2]$, ν = 0.1692 cm²/s, and β = 0.00319 °K^{-1}. Under these conditions, flow is laminar for vertical positions x < 6 m ($Gr_x Pr < 10^9$). The resultant 2D contour plots (see Figure 4) provide interesting insights into free convection. First, the transport effects are localized to a region very close to the wall; at x = 10 cm, the thermal boundary layer, δ_T (defined where Θ = 0.01) is only 13 mm thick, and at x = 1 m, δ_T = 23 mm. Second, the velocity profile evolves in two ways: the velocity increases with vertical position, x, and the point of maximal velocity shifts away from the wall.

Finally, the results can be compared with experimental measurements of Schmidt and Beckmann[7,pp354-355] (see Figure 5, next page). Here, the predicted temperatures agree well with measurements at two different vertical positions (x = 2, 7 cm), but the predicted velocities are 10% below experimental measurements. A similar deviation between model and experimental velocities was reported by Ostrach.[5]

FINAL REMARKS

The example presented here is intended to lead the instructor through stages of problem solving (formulation, calculation, and interpretation), but contains too much content for a standard 60-minute lecture. To date, I have used portions of this example in two classes: transport phenomena at the graduate level and numerical methods at the junior/senior level. In both classes, students were already familiar with the concept of free convection and the objective was to gain experience numerically solving ODEs.

For the graduate course, I derived Eq. (8) and assigned as homework the task of generating Figure 2. I showed the students how to use Euler's method, suggested they employ a trial-and-error approach, and allowed them to use any software of their choosing (*e.g.,* Matlab, Excel).

For the undergraduate course, I developed a computational exercise to solve Eq. (5) and generate Figures 2 and 4 that the class could work through in a computational laboratory. The exercise focused on deriving Eq. (8) from Eq. (5) and implementing an ODE solver and a

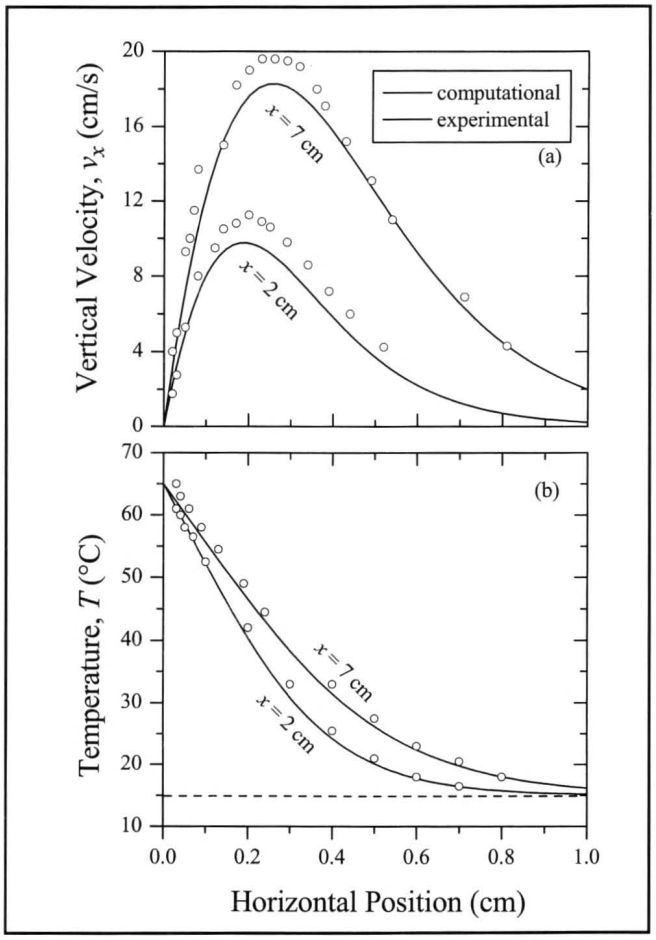


Figure 5. *Comparison of model predictions with experimental measurements for a) vertical velocity and b) temperature. Solid curves are profiles from Figure 4 at vertical positions of 2 and 7 cm. Circles are experimental data points. Dashed line is* T_o.

shooting method algorithm.

The exercise was made available electronically on the course website and the code was presented in a modular form (like Tables 2-5) that could be readily cut and pasted into Matlab. This minimized class time spent composing and debugging code. Active learning techniques were incorporated to address learning objectives (Table 1) and involved short answer questions such as identifying the relevant boundary conditions, testing different conditions (*e.g.,* "Xinit" in Table 3, "Pr" in Table 4), explaining the role of particular commands (*e.g.,* "Cond,max,axis"), and filling in missing fragments of code.

The value of this example as a computational exercise is that it provides the students with working code that can be adapted to solve related problems. Consequently, a useful extension of this in-class exercise would be to design a take-home assignment that requires modification of the model (*e.g.,* vary Pr to generate Figure 3) or application of the model predictions (*e.g.,* apply Eq. 23 to estimate heat flux).

REFERENCES

1. Jones, J.B., "The Non-Use of Computers in Undergraduate Engineering and Science Courses," *J. Eng. Ed.,* **87**, 11 (1998)
2. Deen. W.M., *Analysis of Transport Phenomena,* Oxford University Press, New York, NY (1998)
3. Chapra, S.C., and R.P. Canale, *Numerical Methods for Engineers,* 4th ed., McGraw-Hill, Boston, MA (2002)
4. Higham, D.J., and N.J. Higham, *Matlab Guide,* Society for Industrial and Applied Mathematics, Philadelphia, PA (2000)
5. Ostrach, S., "An Analysis of Laminar Free-Convection Flow and Heat Transfer About a Flat Plate Parallel to the Direction of the Generating Body Force," *NACA Rep.,* 1111 (1953)
6. Le Fevre, E.J., "Laminar Free Convection from a Vertical Plate," *Ninth International Cong. Appl. Mechanics,* **4**, 168 (1956)
7. Welty, J.R., C.E. Wicks, and R.E. Wilson, *Fundamentals of Momentum, Heat and Mass Transfer,* 3rd ed., John Wiley & Sons, New York, NY (1984) ❐

TABLE 5
Conversion of Results to 2-Dimensional Solutions with Contour Plots

```
% Plot results in dimensional space
To=15;, Tw=65; % ambient and wall temp. in degC
g=980; % gravity constant in cm/s^2
b=1/(273+0.5*(Tw+To)); % average coefficient of expansivity
nu=0.1692; % kinematic viscosity at film temp. in cm2/s
%
% calculate x and y space
n=size(Eta,1);
xx=linspace(0,10,101)'*ones(1,n);
yy=(Eta*(4*nu^2*xx(:,1)'/g/b/(Tw-To)).^0.25)';
%
% calculate temperature in x and y space
T=To+(Tw-To)*X(:,4)';
TT=ones(101,1)*T;
figure(3)
cvals=linspace(16,64,7);
[c,h]=contour(yy,xx,TT,cvals,'k-');, clabel(c,h,cvals)
xlabel('Horizontal position (cm)','Fontsize',16)
ylabel('Vertical position (cm)','Fontsize',16)
title('Temperature (\circC)','Fontsize',16)
axis([0 1.5 0 10])
%
% calculate velocity in x and y space
VV=(4*b*g*(Tw-To)*xx(:,1)).^0.5*X(:,2)';
figure(4)
cvals=linspace(5,25,5);
[c,h]=contour(yy,xx,,VV,cvals,'k-');, clabel(c,h,cvals)
xlabel('Horizontal position (cm)','Fontsize',16)
ylabel('Vertical position (cm)','Fontsize',16)
title('Vertical Velocity (cm/s)','Fontsize',16)
axis([0 1.5 0 10])
```