PnS: A Plain 'n SIMPLE CFD Code

The Source Code Manual

Gerald Recktenwald

August 28, 2003

Department of Mechanical Engineering
Portland State University
P.O. Box 751
Portland, Oregon 97207
`www.me.pdx.edu`

Preface

The PnS code is a C language derivative of the "class" code provided by Professor S.V. Patankar at the University of Minnesota. My C implementation is more structured and a bit better documented. All the truly substantial ideas are those of Professor Patankar, and those of his intellectual contemporaries.

The "Plain 'n SIMPLE" moniker is a nod to my good friend Dr. Scott Forbes, who wrote the user interface for and coined the name *QUICK 'n SIMPLE*, for a Macintosh$^{\mathrm{TM}}$ version of this CFD code.

Contents

Chapter 1

Introduction

This document provides a minimal programmers manual for the PnS CFD code. The code is organized into several source files. All but one of these source files constitute the core code which provides the general solution capabilities. The one file outside of the core — the so-called "user" file — is used to define the features of the particular problem being solved. Each different physical problem requires the user, i.e. the programmer, to supply a "user" file that is compiled and linked with the core code to create an executable.

Chapter 2, *Overview* and Chapter 3 *Boundary Conditions* of this manual describe the core code. Chapter 5 *Test Problems* document a series of "user" files designed to highlight specific features of the code. In particular the procedures for solving the following types of physical problems are presented

- different coordinate systems
- different boundary conditions
- heat conduction only
- thermal source terms
- fully-developed flow
- isothermal flow
- buoyancy-induced flow

These notes assume you are familiar with the notation used by Patankar [2]. The PnS code has evolved from a code used by Patankar in his graduate CFD classes at the University of Minnesota.

Ferziger and Perić [1] present a more up-to-date description of the finite volume method. In addition, Perić has provided a large archive of codes that can be downloaded from

```
ftp://ftp.springer.de/pub/technik
```

Chapter 2

The PnS Model and Core Code

This chapter is an brief summary of the core of the PnS code. PnS is short for Plain 'n SIMPLE, which is the CFD code used in QnS (QUICK 'n SIMPLE) the Macintosh shareware CFD program by Scott Forbes and Gerald Recktenald.

The control-volume finite-difference model is discussed. Relationships between the symoblic variables and the code statements are sketched.

Summary of the CVFD Formulation

The so-called general ϕ equation is one of the organizing principles of PnS. In two-dimensional Cartesian coordinates convective transport of a scalar ϕ is governed by

$$\frac{\partial}{\partial x}(\rho u \phi) + \frac{\partial}{\partial y}(\rho v \phi) = \frac{\partial}{\partial x}\left(\Gamma \frac{\partial \phi}{\partial x}\right) + \frac{\partial}{\partial y}\left(\Gamma \frac{\partial \phi}{\partial y}\right) + S = 0 \qquad (2.1)$$

where ρ is the fluid density, u and v are the velocity components in the x and y directions, Γ is the diffusion coefficient and S is the volumetric source term. If the velocity components are zero, equation (2.1) reduces to the Poisson equation, which applies to heat conduction and simple fully-developed duct flow.

The control-volume finite-difference (CVFD) method obtains an approximate solution to equation (2.1) for a set of discrete φ values corresponding to nodes on a finite-volume mesh. Equation (2.1) is converted to a system of linearized algebraic equations with one equation for each nodal value of φ. The system of equations is solved iteratively using either line-based relaxation techniques or algebraic multigrid. Details of the solution algorithms are not presented here.

Equation (2.1) governs the conservation of a generic, scalar field variable, φ. For two-dimensional, incompressible flow there is one φ equation for each of the velocity components, u and v. A discrete equation for the pressure field is obtained by manipulating the two momentum equations and the continuity equation. The three field equations are coupled to each other and, in principle, must be solved simultaneously. The SIMPLE algorithm [2] allows these three coupled equations to be iteratively solved in a sequential procedure. In SIMPLE, each of the field equations (say the equation for u) is solved by temporarily freezing values from the other discrete fields (v and p) to which it is coupled.

The Finite Volume Grid

Figure 2.1 depicts a rectangular domain of length L_x in the x-direction and L_y in the y-direction. The domain is divided into non-overlapping control volumes by the grid lines. The control volumes are square in Figure 2.1, but this is not required. Use of similar grids in other coordinate systems requires the introduction of additional notation.

At the center of each control volume is a node, designated by an open circle. Two sets of grid lines can be identified: the grid lines that define the control volume faces, and the grid lines (not shown) that define the locations of the nodes. In the PnS code the grid lines that define node locations are stored in the variables `x[i]` and `y[j]`. The grid lines that define the control volume interfaces are stored in `xu[i]` and `yv[j]`.

Nodes in the domain may be identified by their (i, j) grid indices. By convention an additional naming convention based on the directions on a map is used to simplify the algebra. Figure 2.2 is a detailed sketch of one of the control volumes in the domain. A typical node (i, j) in Figure 2.1 is also referred to as node P in Figure 2.2. The $(i+1, j)$ and $(i-1, j)$ neighbors of P are designated E for east and W for west, respectively. The $(i, j+1)$ and $(i, j-1)$ neighbors of P are referred to as N and S for north and south, respectively. Figure 2.2 also defines numerous geometric variables. In general the width, Δx, of a control volume will not be equal to the distances δx_e and δx_w between P and its east and west neighbors. Regardless of the grid spacing P is always located in the geometric center of its control volume. Thus

$$x_P - x_w = x_e - x_P = \frac{\Delta x}{2} \tag{2.2}$$

$$y_P - y_s = y_n - y_P = \frac{\Delta y}{2} \tag{2.3}$$

In these expressions it is crucial to distinguish between upper and lower case letters used as subscripts. Lower case subscripts refer to the locations of the control volume faces. Upper case subscripts refer to the locations of the nodes.

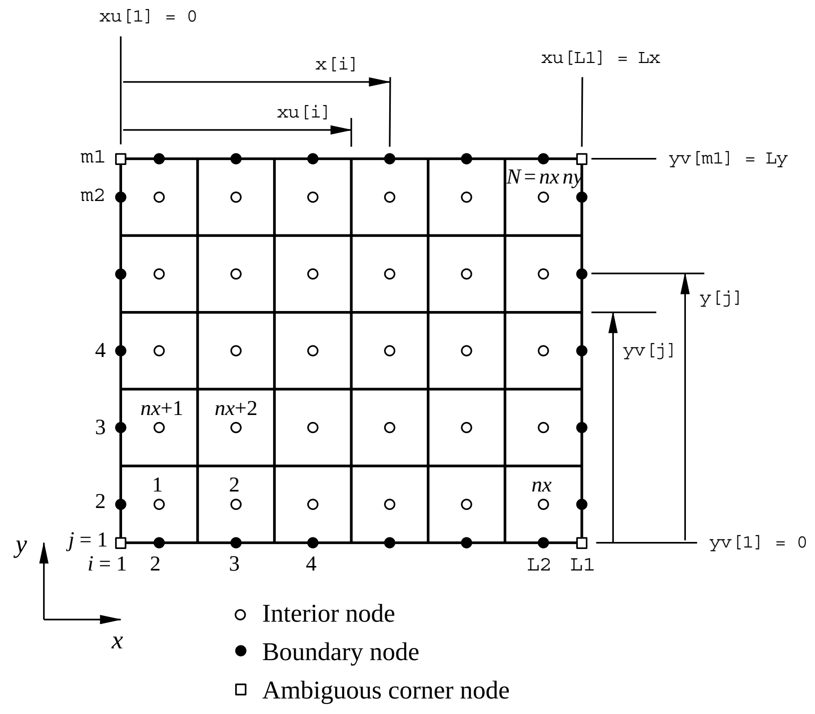

Figure 2.1: The finite volume grid.

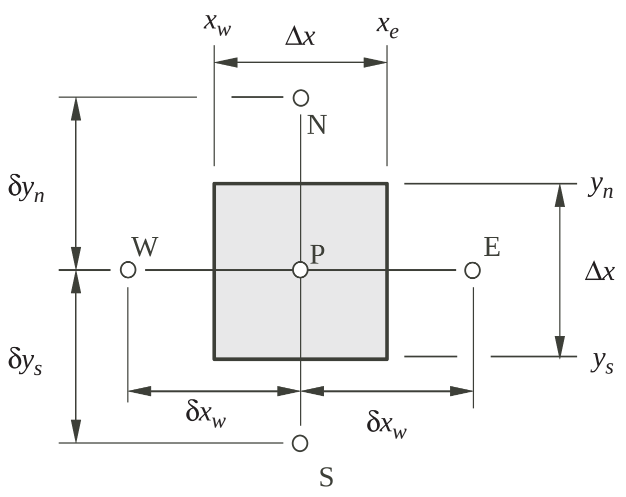

Figure 2.2: Geometric variables for a typical control-volume. The point P is located in the center of the control volume even if the grid is non-uniform.

Discrete Approximation for Interior Control Volumes

The control volume finite-difference method is used to transform equation (2.1) to a system of discrete equations for the nodal values of ϕ. First, equation (2.1) is integrated over the typical control volume depicted in Figure 2.2. This reduces the equation to one involving only first derivatives in space. Then these first derivatives are replaced with central difference approximations or upwind approxmations as appropriate. Consult Patankar [2] for details. The resulting discrete equation for node P can be written

$$-a_S\phi_S - a_W\phi_W + a_P\phi_P - a_E\phi_E - a_N\phi_N = b \tag{2.4}$$

This is a pentadiagonal matrix equation

$$Ax = b$$

where the elements of the x vector are the interior ϕ values. Patankar prefers to write

$$a_P\phi_P = a_E\phi_E + a_W\phi_W + a_N\phi_N + a_S\phi_S = b \tag{2.5}$$

Source Terms

The b term in equations (2.4) and (2.5) arise from discretization of the source term, S, in equation (2.1). In addition to true source term, additional effec-

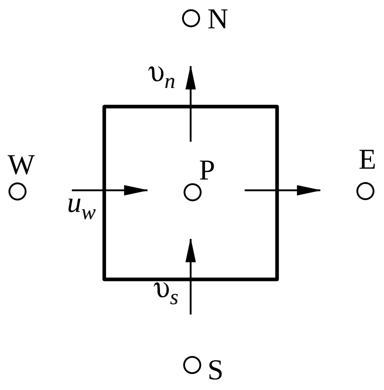

Figure 2.3: Velocity variables for a typical control volume.

tive source terms arise in the implementation of the boundary conditions (see Chapter 3).

Nonlinearities in the source terms must be treated with care. See Patankar [2] for details.

The SIMPLE and SIMPLER Algorithms

Let u_{ij}, v_{ij} and p_{ij} be the discrete field values on the two-dimensional mesh. The i and j subscripts are grid indices. A simplified version of the SIMPLE algorithm is

1. Obtain the system equations for the u_{ij} by assuming the the v_{ij} and p_{ij} values are known (constant).
2. Solve the system of equations for u_{ij} to obtain an updated field u^*_{ij}.
3. Obtain the system equations for the v_{ij} by assuming the the u^*_{ij} and p_{ij} values are known (constant).
4. Solve the system of equations for v_{ij} to obtain an updated field v^*_{ij}.
5. Given the u^*_{ij} and v^*_{ij} fields, obtain the system of equations for the pressure correction equation, p'_{ij}.
6. Solve the sytem of equations for p'_{ij}.
7. Update the velocity and pressure fields

$$\begin{aligned} u_{ij} &= u^*_{ij} + u'_{ij} \\ v_{ij} &= v^*_{ij} + v'_{ij} \\ p_{ij} &= p^*_{ij} + p'_{ij} \end{aligned}$$

ϕ	global index	Γ
u	`nUeqn`	`gammaf[nUeqn]` = `gammaf[nVeqn]` $\Leftrightarrow\ \mu$
v	`nVeqn`	`gammaf[nUeqn]` = `gammaf[nVeqn]` $\Leftrightarrow\ \mu$
p	`nPeqn`	not applicable
p'	`nPCeqn`	not applicable
ϕ_1	`FirstScalar`	`gammaf[FirstScalar]` $\Leftrightarrow\ \Gamma_{\phi_1}$
ϕ_2	`FirstScalar+1`	`gammaf[FirstScalar+1]` $\Leftrightarrow\ \Gamma_{\phi_2}$
$\vdots$	$\vdots$	$\vdots$
ϕ_m	`LastScalar`	`gammaf[LastScalar]` $\Leftrightarrow\ \Gamma_{\phi_m}$

Table 2.1: Variables associated with the scalar ϕ equations. Note that `gammaf[n]` is a pointer to a two dimensional array, not a scalar.

where u'_{ij} and v'_{ij} are related to p'_{ij}.

These steps are repeated until the discrete fields converge. More details are provided by Patankar [2].

Organization of the Dependent Variables

The PnS code uses a so-called segregated solution algorithm in which each field variable (u, v, p, T,...) is solved separately. Coupling between the variables is resolved by iteration. The kernels of the solution algorithms are designed to set up and solve one ϕ field at a time. Table 2.1 lists the program variables associated with each scalar ϕ field. The second column is the global index variable associated with that field.

Code Structure

PnS is divided into two primary sections: the core code and the user code. The user-defined code implements the geometry definitions, boundary conditions, source terms, intermediate print-out, and post-processing necessary for a particular physical problem.

The core code contains routines for

- Implementation of SIMPLE and SIMPLER
- Computation auxillary grid variables given the user-defined grid parameters.

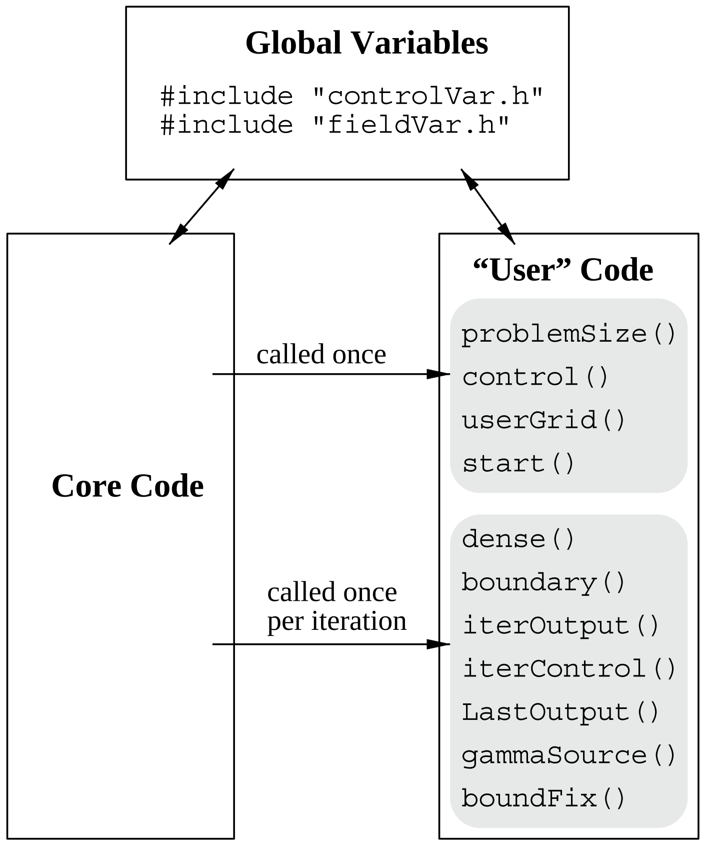

Figure 2.4: Geometric variables for a typical control-volume.

- Computation of the discrete systems of equations for the dependent variables.
- Solution of the discrete system of equations.

Links between the core code and the user-defined code are represented schematically in Figure 2.4.

Chapter 3

Boundary Condition Implementation

Heat Conduction and Scalar Equations

Transport equations for scalars like internal energy (temperature), pollutant concentration are readily related to the general φ equation. In this section the implementation of boundary conditions for the energy equation will be used to demonstrate how boundary conditions for any scalar can be treated.

Dirichlet Boundary Conditions

The solution algorithms in PnS treats Dirichlet boundary conditions by default. All that is needed to specify the $T =$ constant boundary conditions is to assign temperature values to the appropriate boundary nodes.

Adiabatic (Neumann) Boundary Conditions

The adiabatic boundaries are implemented via a "trick" that is used throughout the code. Equation (2.4) corresponds to an energy balance on the control volume surrounding the "P" node. An adiabatic boundary, by definition, results in zero heat flow across the boundary. A control volume adjacent to a boundary has one of the faces coincident with the boundary surface, and therefore if the boundary is adiabatic there is no heat flow across that face of the control volume.

Consider the control volume shown in Figure 3.1. The node on the boundary has a temperature T_b and the first interior node adjacent to the boundary has temperature T_i. (The subscript "i" refers to "interior", not the grid index in the x direction.) The distance between these nodes is δ. Using the harmonic mean formulation [2] the heat flux into the control volume across the boundary is

$$q = \frac{\Gamma_{\mathit{eff}}}{\delta}(T_b - T_i)$$

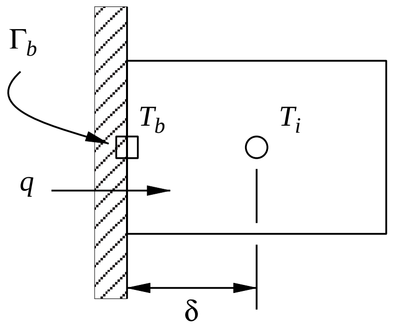

Figure 3.1: Control volume adjacent to an adiabatic boundary.

where

$$\Gamma_{\mathit{eff}} = \frac{\Gamma_b \Gamma_i}{\beta\Gamma_b + (1-\beta)\Gamma_i}$$

and β is a geometric weighting parameter. If we set Γ_b equal to zero in this expression (regardless of the value of β), Γ_{eff} will be zero, which leads to zero heat flux across the boundary.

Setting Γ_b to zero on the boundary guarantees that the energy balance, and therefore the coefficients in equation (2.4), for the near-boundary control volume will result in zero heat flow across the boundary. When the system of equations for all the internal ϕ values is obtained it will give the correct temperature distribution inside the solid. The solution algorithm in the PnS code only solves the discrete equations for the interior nodal values. The boundary nodal values are never changed. This makes it easier to write the solution routines, but it shifts some additional computational burden onto the "user" routines. So how do we find the value of T_b once the value of T_i is known?

An adiabatic boundary requires that the temperature gradient is zero in the direction normal to the boundary. A finite-difference approximation to the temperature gradient involving the nodes in Figure 3.1 is

$$\frac{\partial T}{\partial x} \approx \frac{T_i - T_b}{\delta} \tag{3.1}$$

A zero gradient requires $T_b = T_i$. This is easily handled by updating the boundary temperatures *after* the internal temperatures have been obtained by the built-in PnS solver.

In summary an adiabatic boundary is implemented in PnS with the following steps

1. Set $\Gamma_b = 0$ for the node *on the boundary*

2. Update the boundary temperature with $T_b = T_i$ after the internal temperatures have been computed.

Prescribed Flux Boundary Conditions

Prescribed flux boundary conditions are implemented as a variation on the adiabatic boundary condition described in the preceding section. For the control volume in Figure 3.1 the flux, q, is known, but not zero.

1. Add a fixed, distributed source term for the near wall control volume. The strength of the volumetric source term is

   $$S = \frac{qA}{V}$$

   where A is the area of the control volume face through which q passes, and V is the volume of the control volume.

2. Set $\Gamma_b = 0$ for the node *on the boundary.* This isolates the value of T_i from the value of T_b during the solution of the internal nodal temperatures. The correct contribution to the energy balance on the near-wall control volume is guaranteed by the preceding step.

3. Update the boundary temperature with

   $$T_b = T_i + \frac{q\delta}{\Gamma_b}$$

   after the internal temperatures have been computed.

Convective Boundary Conditions

Though convective boundary conditions are physically quite different, in the PnS code these are treated as an extension of the prescribed flux boundary condition.

A convective boundary condition applies when the heat flux from the surface to a surrounding fluid is given by

$$q = h(T_b - T_f) \tag{3.2}$$

where h is a heat transfer coefficient, T_b is the temperature of the boundary node, and T_f is the temperature of the fluid (a known constant). Continuity of heat flow at the surface of the domain also requires

$$q = \Gamma_b \frac{T_b - T_i}{\delta} \tag{3.3}$$

Combining equation (3.2) and (3.3) gives[1]

$$q = \Gamma_b \frac{T_f - T_i}{1/h + \delta/\Gamma_b} \tag{3.4}$$

[1]If $A = a/b = c/d$, then $A = (a + c)/(b + d)$

As with the prescribed flux boundary condition, equation (3.2) is used to specify source term for the near-wall control volume of the form.

$$S = \frac{qA}{V}$$

As indicated by equation (3.4), the source term depends on the unknown value of T_i. To accelerate convergence the source term is linearized according to the recommendation of Patankar [2] as

$$S = S_c + S_p T_i$$

where S_c is a constant term, and S_p is a coefficient that must be negative for a stable linearization. Linearizing equation (3.4) according to this recommendation gives

$$S_c = \frac{A}{V}\frac{T_f}{1/h + \delta/\Gamma_b} \tag{3.5}$$

$$S_p = -\frac{A}{V}\frac{1}{1/h + \delta/\Gamma_b} \tag{3.6}$$

The steps for implementing the convective boundary condition are

1. Add the source terms in equation (3.5) and (3.6) for the near wall control volume.
2. Set $\Gamma_b = 0$ for the node *on the boundary.* This isolates the value of T_i from the value of T_b during the solution of the internal nodal temperatures. The correct contribution to the energy balance on the near-wall control volume is guaranteed by the preceding step.
3. Update the boundary temperature with

   $$T_b = \frac{hT_f + (\Gamma_b/\delta)T_i}{h + (\Gamma_b/\delta)}$$

   after the internal temperatures have been computed.

Decoration of Boundary Values

Use of the source term modifications effectively eliminates the boundary values as unknowns. Once the problem has been solved we still need to know the values of the unknown φ variables at the boundary. In some cases these boundary values must be known during the solution procedure. Regardless of whether these boundary values are needed during the solution, it is usually desirable to compute the boundary values at the end of the solution to prepare contour plots or other means of displaying the solution. In the situations the calculation of the boundary values is like a decoration on the solution procedure–it is useful or nice, but not essential because the interior values are already known.

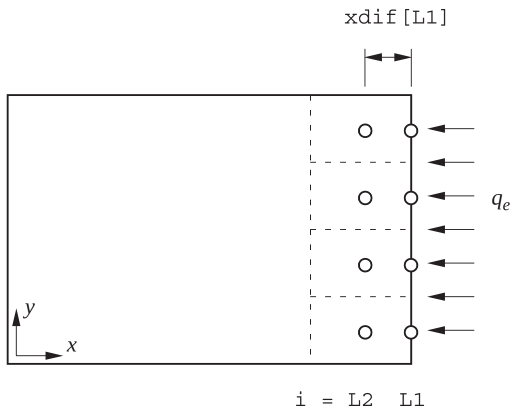

Figure 3.2: Hypothetical problem with imposed heat flux q_E imposed on the far eastern boundary of the domain.

Constant flux boundary conditions impose a relationship between the boundary values and the interior values. Once the interior values are known the boundary values are computed from imposed boundary condition.

A Neumann boundary condition is the easiest to decorate. Since the condition

$$\frac{\partial \varphi}{\partial n} = 0$$

implies that there is no change in φ in the direction n, normal to the boundary, the boundary value is set equal to its nearest interior neighbor.

The computed boundary value obviously depends on the type of imposed boundary conditions, but it also depends on the discretization procedure for the interior control volumes. This is the case for constant flux boundary conditions. Consider the constant flux boundary condition depicted in Figure 3.2. For heat conduction (no flow) the imposed flux requires

$$q_x\big|_{x=x_{\mathrm{L1}}} = -k\left.\frac{\partial T}{\partial x}\right|_{x=x_{\mathrm{L1}}} = q_e \tag{3.7}$$

on the eastern boundary of the domain ($x = x_{\mathrm{L1}}$). The boundary value of T is decorated with the discrete equivalent of equation (3.7). In the CVFD code the diffusion terms are approximated by central differences so equation (3.7) becomes

$$-\mathtt{qe} = \mathtt{conduct} * (\mathtt{T}[\mathtt{L1}][\mathtt{j}] - \mathtt{T}[\mathtt{L2}][\mathtt{j}])/\mathtt{xdif}[\mathtt{L1}] \tag{3.8}$$

where `qe` is the applied heat flux (negative because for positive `qe` the heat flows in the negative x direction), conduct is the thermal conductivity, `T[L1][j]` is the temperature on the far eastern boundary, `T[L2][j]` is the temperature at the next interior node, and `xdif[L1]` is x-direction grid spacing between these two nodes as shown in Figure 3.2. Solving equation (3.8) for the unknown boundary temperature gives

$$\texttt{T[L1][j]} = \texttt{T[L2][j]} - \texttt{qe} * \texttt{xdif[L1]/conduct}$$

which is the formula used to decorate the boundary after the source term modifications have been used to compute the internal temperature field.

Constraints on Interior Values

Complex physical problems often require irregular domains and internal blockages to flow. Solution of conjugate heat transfer problems is possible by allowing fluid flow in part of the domain and discontinuities in thermal conductivity at the fluid/solid interface. Another use of interior constraints is the approximation of irregular domains by blocking off regions in one of the regular coordinate systems (Cartesian, axisymmetric, and polar) used in the CVFD code. Each of these applications requires the imposition of internal constraints on internal nodes values.

Patankar [2, Chapter 7, p. 145] gives a method for constraining interior values of φ. This technique is based on modifications to the discrete φ equation so that the effect of neighboring values of φ are negligible. The discrete φ equation can be written

$$\left(\sum a_{nb} - S_P \Delta V\right)\varphi_P = \sum a_{nb}\varphi_{nb} + S_C \Delta V \tag{3.9}$$

If the desired internal constraint is $\varphi = \varphi_{\text{desired}}$, then Patankar recommends setting

$$S_P = H$$
$$S_C = H\varphi_{\text{desired}}$$

where H is some huge number (say, $H \sim 10^{30}$). With this substitution equation (3.9) reduces to

$$H\varphi = H\varphi_{\text{desired}} \tag{3.10}$$

where the influence of the neighboring coefficients and φ values has been swamped by the magnitude of H. The obvious solution to equation (3.10) is $\varphi = \varphi_{\text{desired}}$. The advantage of this trick is that it achieves the desired result without requiring modification of solution algorithm or the basic calculation of the coefficients. An important undesirable consequence is that the norm of the residual, $\|\mathbf{r}\|$, is rendered meaningless because the constrained elements of the solution vector, $\mathbf{x}$ (in $\mathbf{A}\mathbf{x} = \mathbf{b}$) is a factor of H larger than the unconstrained values, and the diagonal of the $\mathbf{A}$ matrix contains coefficients that have no meaningful relationship to the other diagonal coefficients or the other coefficients in the same row.

A simple modification (Van Doormaal and Raithby, 1984) to the trick in equation (3.10) removes this undesirable side-effect. Let $a_{P,ref}$ be a value of a_P

from a control volume that is physically close to the control volume in which the constraint on φ is imposed. Then

$$S_P = -a_{P,ref}$$
$$S_C = -a_{P,ref}\,\varphi_{\text{desired}}$$

has the effect of constraining φ without changing the character of the **A** matrix or the magnitude of the residual for the φ equation. With this trick the residual reduction criteria can be applied to all φ equations. The following code fragment implements this constraint.

```
constant = ...;
apref = ap[...][...];

for (i=...; i<=...; i++)
    for (j=...; j<=...; j++ )
        ap[i][j] = apref;
        con[i][j] = constant*apref;
        ae[i][j] = aw[i][j] = an[i][j] = as[i][j] = 0.0;
```

The ellipses (...) should be replaced by appropriate values, indices or index limits, as appropriate. This code should be in function `boundFix()`. Note that `ap[i][j]` = `apref` is consistent with the specification

Fluid Flow

Outflow Boundary Conditions

The conservative nature of the CVFD formulation and the SIMPLE-based algorithms provides major advantages at the cost of the exacting discipline required to maintain mass conservation. The *u*-*v*-*p* coupling algorithms enforce strict mass conservation. An apparently trivial error that leads to mass balance errors will cause the code to fail. The outflow boundary treatment must preserve mass conservation at each iteration.

The SIMPLE and SIMPLER algorithms use the pressure correction equation to bring the velocity fields obtained by solving the momentum equation into exact agreement with the continuity equation on a control-volume by control-volume basis. Because the outflow velocities are used in the source terms to the pressure correction equation these boundary velocities must be consistent with overall mass conservation.

Two basic techniques are used to specify the velocity at the outflow boundary for incompressible flows. These techniques will be referred to as the additive factor and multiplicative factor methods. Both methods relate the velocities at the outflow boundary to the interior nodes adjacent to the outflow boundary. The sketch in Figure V.2 represents at typical flow problem in which the outflow

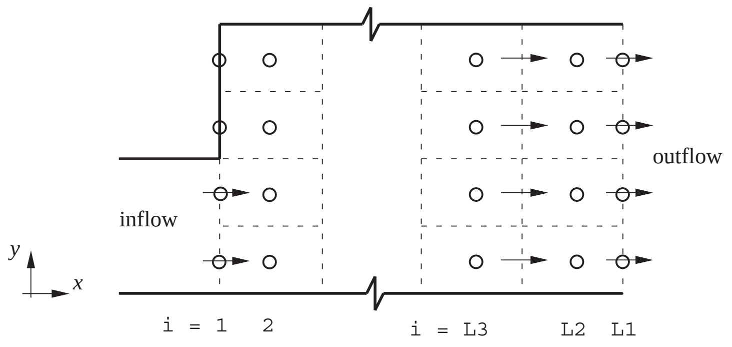

Figure 3.3: Hypothetical problem with an outflow boundary on the far eastern boundary of the domain.

boundary is at the far eastern boundary of the domain. For clarity only the x-direction velocities at the inflow and outflow are shown.

Conservation of mass in a steady, incompressible flow problem requires that the total outflow equals the total inflow. This is an integral constraint on the velocities on the outflow boundary. One way of satisfying to this constraint would be to prescribe an uniform velocity at the outflow boundary such that

$$\int_{A_{in}} \rho \mathbf{u} \cdot d\mathbf{A} = \int_{A_{out}} \rho \mathbf{u} \cdot d\mathbf{A}$$

where A_{in} is the area of the inlet, and A_{out} is the area of the outlet. For the situation depicted in Figure 3.3 this integral would be evaluated as

$$\sum_{j_{in}} \texttt{rho[1][j]} * \texttt{u[2][j]} * \texttt{ycv[j]} = \sum_{j_{out}} \texttt{rho[L1][j]} * \texttt{u[L1][j]} * \texttt{ycv[j]}$$

substituting, `uout`, an unknown but uniform value of the velocity at the outflow boundary gives (set $\texttt{u[L1][j]} = \texttt{uout}$)

$$\texttt{uout} = \frac{\sum_{j_{in}} \texttt{rho[1][j]} * \texttt{u[2][j]} * \texttt{ycv[j]}}{\sum_{j_{out}} \texttt{rho[L1][j]} * \texttt{ycv[j]}}$$

Thus, global mass conservation is used to obtain the boundary condition at the outflow boundary. Clearly, a uniform velocity is not expected to give very realistic flow field near the outflow boundary. The additive factor and multiplicative factor methods presented in the following sections provide methods of using the interior flow field as guidance for the velocity profile at the outflow boundary.

Additive Factor

Instead of assuming that the outflow velocity profile is uniform, suppose that the velocity profile at the outflow boundary is related to the velocity profile at the nearest upstream (interior) control volumes. Specifically, assume that these velocity values are related by an additive factor. For the problem depicted in Figure V.2 this would be

$$\texttt{u[L1][j]} = \texttt{u[L2][j]} + \texttt{C}$$

where `C` is an unknown that is uniform across the outflow boundary. Thus, conservation of mass (inflow = outflow) requires

$$\sum_{j_{in}} \texttt{rho[1][j]} * \texttt{u[2][j]} * \texttt{ycv[j]} = \sum_{j_{out}} \texttt{rho[L1][j]} * (\texttt{u[L1][j]} + \texttt{C}) * \texttt{ycv[j]}$$

and solving for `C` yields

$$C = \frac{\sum_{j_{in}} \texttt{rho[1][j]} * \texttt{u[2][j]} * \texttt{ycv[j]} - \sum_{j_{out}} \texttt{rho[L1][j]} * \texttt{ycv[j]}}{\sum_{j_{out}} \texttt{rho[L1][j]} * \texttt{ycv[j]}}$$

Multiplicative Factor

This is similar to the additive factor treatment except that the velocity profile at the outflow boundary is related to the velocity profile at the nearest upstream location by a multiplicative factor. For the problem depicted in Figure V.2 this would be

$$\texttt{u[L1][j]} = \texttt{C} * \texttt{u[L2][j]}$$

Conservation of mass (inflow = outflow) requires

$$\sum_{j_{in}} \texttt{rho[1][j]} * \texttt{u[2][j]} * \texttt{ycv[j]} = \sum_{j_{out}} \texttt{rho[L1][j]} * \texttt{C} * \texttt{u[L2][j]} * \texttt{ycv[j]}$$

and solving for `C` yields

$$C = \frac{\sum_{j_{in}} \texttt{rho[1][j]} * \texttt{u[2][j]} * \texttt{ycv[j]}}{\sum_{j_{out}} \texttt{rho[L1][j]} * \texttt{ycv[j]}}$$

Chapter 4

Fully Developed Flow in Ducts

Hydrodynamically Fully-Developed Flow

Consider the situation depicted in Figure 4.1. Fluid is flowing through a duct of arbitrary but constant cross-section. If the duct is long enough, the flow field becomes invariant with z and the pressure gradient, dp/dz, becomes constant. The flow is then said to be hydrodynamically fully developed.

In a fully-developed flow, one is interested in the friction factor f

$$f = \frac{D_h}{\frac{1}{2}\rho \bar{w}^2}\left(-\frac{dp}{dz}\right) \tag{4.1}$$

where D_h is the hydraulic diameter, $\bar{w}$ is the average velocity in the z direction. Also of interest is the duct Reynolds number

$$\text{Re} = \frac{\rho \bar{w} D_h}{\mu}. \tag{4.2}$$

where D_h is the hydraulic diameter, D_h. For fully-developed laminar flow in a duct the product $f\text{Re}$ is a constant. In other words, $f\text{Re}$ is only determined by the geometry of the duct.

In general the velocity field will be a vector valued function

$$\mathbf{u} = \hat{e}_x u + \hat{e}_y v + \hat{e}_z w$$

where $\hat{e}_x$, $\hat{e}_y$, and $\hat{e}_z$ are the unit basis vectors, and u, v, and w are the velocity components in the x, y, and z directions, respectively. For fully-developed flow u and v can be non-zero, but w must not be a function of z.

For incompressible flow the average velocity in the duct is

$$\bar{w} \equiv \frac{1}{A_c}\int_{A_c} w\, dA \tag{4.3}$$

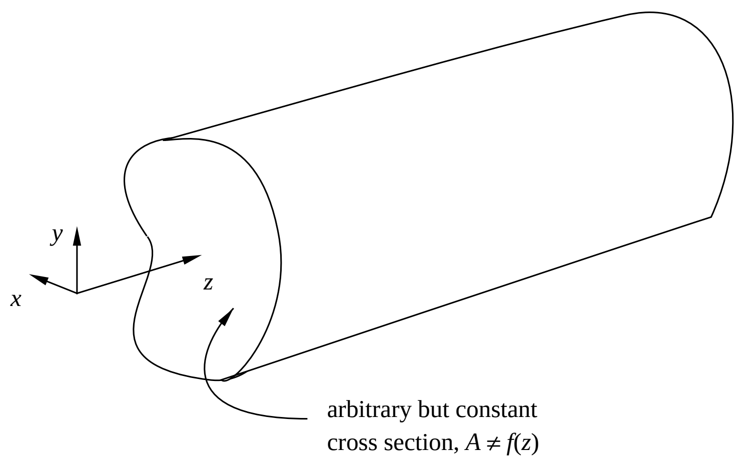

Figure 4.1: Fully-developed flow and heat transfer in a duct of arbitrary cross-section.

where A_c is the cross-sectional area of the duct normal to the z direction.

For so-called *simple* fully-developed flow, $u = v = 0$, and the Navier-Stokes equations reduce to

$$0 = \mu \left(\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} \right) - \frac{dp}{dz} \tag{4.4}$$

with boundary conditions $w = 0$ on the duct walls. To solve the flow problem, one imposes an arbitrary pressure gradient and solves Equation (4.4). The solution gives $w(x, y)$ which is used to compute $\bar{w}$ from Equation (4.3). Different values of dp/dz yield different values of $\bar{w}$. However the $f\text{Re}$ product

$$f\text{Re} = -\frac{dp}{dz}\frac{D_h^2}{\mu \bar{w}} \tag{4.5}$$

is a constant for a given cross section.

Thermally Fully-Developed Flow

If the flow is hydrodynamically fully-developed, then under certain types of thermal boundary conditions, the flow can also become thermally fully-developed. In that case, one can define a dimensionless temperature that is invariant with z. In a thermally fully-developed flow, the *dimensional* temperature, e.g. the temperature in °C or °F, will vary with z.

Remember that the flow must be hydrodynamically fully-developed in order for it to become thermally fully-developed. On the other hand if the fluid properties are unaffected by temperature the flow can be hydrodynamically fully-developed without being thermally fully-developed.

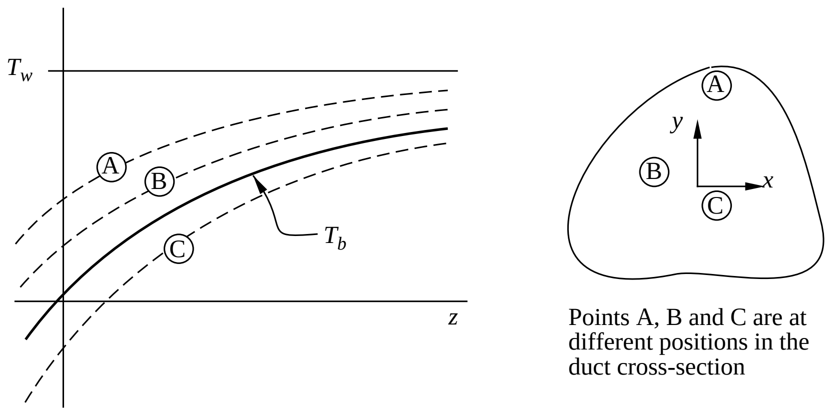

Figure 4.2: Temperature variation in the flow direction for thermally fully-developed flow in a duct with constant wall temperature.

Define the fluid bulk temperature (mixing cup temperature)

$$T_b \equiv \frac{\displaystyle\int_{A_c} \rho\, w\, c_p\, T\, dA}{c_{p,ref} \displaystyle\int_{A_c} \rho\, w\, dA}$$

The numerator in this expression is the energy flowing through the duct cross-section at some position z. The denominator is $c_{p,ref}$ times the mass flowing through the duct cross section at z. For incompressible flow of fluid with uniform properties the expression for the bulk temperature reduces to

$$T_b \equiv \frac{\displaystyle\int_{A_c} w\, T\, dA}{\displaystyle\int_{A_c} w\, dA}$$

If the duct wall temperature is uniform at T_w then T_b asymptotically approaches T_w as shown in Figure 4.2. Note that the temperature is *not* uniform across the duct.

For thermally fully-developed flow there exists is a suitably defined dimensionless temperature variable, θ, that is independent of z. When T_w is uniform θ is

$$\theta = \frac{T(x,y,z) - T_w}{T_b(z) - T_w} \qquad \text{uniform } T_w \text{ only} \tag{4.6}$$

Under thermally fully-developed flow conditions the energy equation is

$$\rho c_p w \frac{\partial T}{\partial z} = k\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right) \tag{4.7}$$

Solve equation (4.6) for T and differentiate with respect to z to get

$$\frac{\partial T}{\partial z} = \theta \frac{dT_b}{dz} \tag{4.8}$$

Note that this equation is not restricted to any point in the cross-section. Therefore the temperature curves in on the left side of Figure 4.2 are all parallel. Substitute equation (4.6) and equation (4.8) into equation (4.7) to get the governing equation for θ

$$k\left(\frac{\partial^2\theta}{\partial x^2} + \frac{\partial^2\theta}{\partial y^2}\right) = \rho\, c_p\, w \frac{\partial T_b}{\partial z}$$

or

$$\frac{\partial^2\theta}{\partial x^2} + \frac{\partial^2\theta}{\partial y^2} = \left(\frac{\rho\, c_p}{k}\frac{\partial T_b}{\partial z}\right) w\, \theta \tag{4.9}$$

From Figure 4.2 we see that the value of dT_b/dz is determined by the z position in the duct. Though the actual value of dT_b/dz is arbitrary, and changes with z, the value of $\bar{\theta}$

$$\bar{\theta} \equiv \frac{\displaystyle\int_{A_c} w\theta dA}{\displaystyle\int_{A_c} w dA} \tag{4.10}$$

is constant for thermally fully-developed flow.

Example Problem

In this example problem we obtain the thermally fully-developed temperature field for the case of uniform wall temperature in a rectangular duct. Remember that there are other boundary conditions that admit thermally fully-developed flow.

PnS Implementation

To solve equation (4.4) with the PnS code we need to

1. choose and arbitrary dp/dz
2. solve the Poisson equation for $w(x, y)$

The pressure gradient is a source term in the w equation.

Because the PnS code uses an iterative two-dimensional solution algorithm the discrete w equation is solved until $f\text{Re}$ is a constant. If the code used a direct solution technique, such as LU decomposition with back-substitution, no iterations would be necessary.
To solve equation (4.9)

1. choose an arbitrary value of dT_b/dz
2. guess $T(x, y)$
3. compute $\theta(x, y)$ from the guessed $T(x, y)$
4. solve equation (4.9)

and repeat steps 3 and 4 until convergence.

Chapter 5

Test Problems

This chapter describes a series of test problems for the PnS codes. Each problem involves a hand-coded set of routines that work with the core code to analyze a specific situation. Table 5.1 lists these test problems by the source file that contains the implementation.

One-Dimensional Heat Conduction: `user0.c`

This trivial problem is designed for testing and debugging the solution algorithms used to solve Equation (2.4). When the boundary conditions are correctly specified the PnS code will yield the exact solution to one-dimensional heat conduction with uniform properties.

With $\phi = T$, $\rho = 1$, $u = v = 0$, $\Gamma = 1$ and $S = 0$, Equation (2.1) reduces to

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0$$

which governs steady heat conduction with no source term and uniform properties. `user0.c` involves solution to this equation for two different orientations. Version (a) involves the one-dimensional temperature variation in the x-direction

$$T(x, y) = T_1 + (T_2 - T_1)\frac{x}{L}$$

and version (b) involves the corresponding problem in the y-direction

$$T(x, y) = T_1 + (T_2 - T_1)\frac{y}{L}$$

The physical problems are represented by the sketch in Figure 5.1. Uniform temperatures are imposed on opposite ends of the domain and the remaining boundaries are adiabatic. The `user0a.c` and `user0b.c` codes implement the solution to version (a) and version (b), respectively. Details of implementing version (a) are described in Chapter 3.

Table 5.1: User-defined problems described in this chapter. Problems are listed in order of increasing complexity

source file	Description
`user0a.c`	1-D conduction in the y-direction
`user0b.c`	1-D conduction in the x-direction
`user1.c`	2-D conduction with an exact solution
`user2.c`	2-D conduction with complex boundary conditions
`user3.c`	fully-developed flow and heat transfer
`user4.c`	fully-developed flow with conjugate heat transfer
`user5.c`	convective transport in a corner
`cavity.c`	flow in a cavity having a sliding lid
`box.c`	flow through a box
`user6.c`	sudden expansion with buoyancy

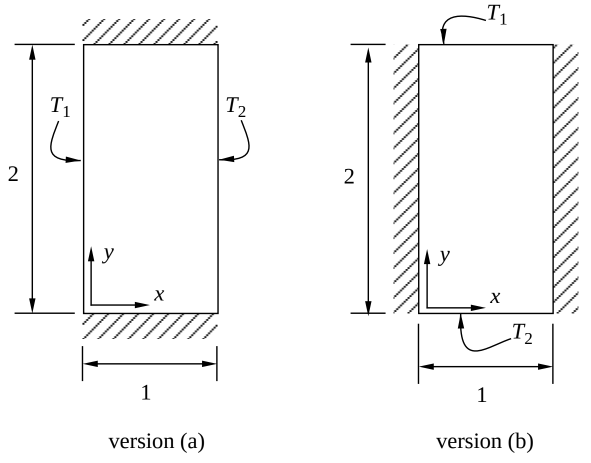

Figure 5.1: Schematic of problems solved with `user0a.c` and `user0b.c`. Cross-hatched boundaries are adiabatic.

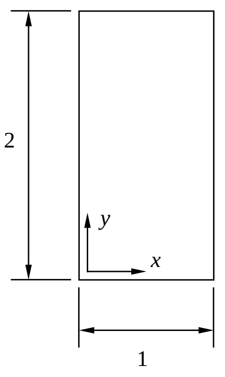

Figure 5.2: Calculation domain for sample problem solved with `user1.c`.

Two-Dimensional Heat Conduction: `user1.c`

Solve the steady-state heat conduction problem in a rectangle with uniform heat conduction and no source term. As shown in Figure 5.2 the calculation domain is one unit long in the x direction and two units long in the y direction.

The boundary temperatures are given by

$$T(x, y) = x + y + xy$$

which is also the exact solution to the temperature field. This problem is a simple extension to `user0`. All boundary conditions are Dirichlet.

The computer program will obtain the exact solution. Although the temperature distribution is nonlinear in x and y, the temperature gradients

$$\frac{\partial T}{\partial x} = 1 + y \qquad \frac{\partial T}{\partial y} = 1 + x$$

are linear. Thus, the central difference formulation yields the exact local temperature gradients.

Heat Conduction with Complex Boundary Conditions: `user2.c`

Highlights

- use cylindrical coordinates
- include a thermal source term
- use a temperature-dependent thermal conductivity

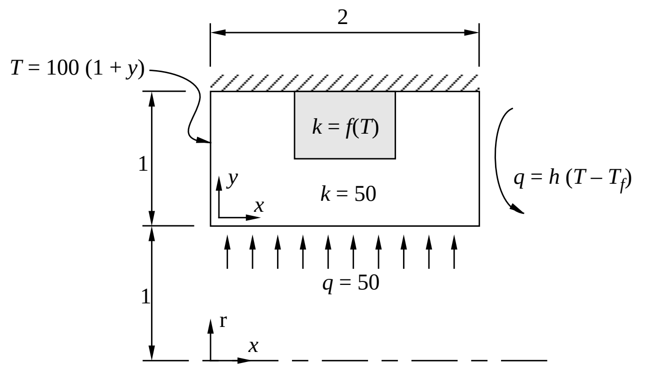

Figure 5.3: Calculation domain for sample problem 2, and implemented in `user2.c`.

- implement various thermal boundary conditions, including two variants on convective boundary conditions

This problem involves solution of the heat conduction equation

$$\frac{1}{r}\frac{\partial}{\partial r}\left(rk\frac{\partial T}{\partial r}\right)+\frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right)+S=0 \tag{5.1}$$

in the cylindrical (x, r) domain illustrated in Figure 5.3. Thermal conductivity in shaded region is temperature dependent

$$k=0.2\left(1+\frac{T}{100}\right)$$

The volumetric heat source term throughout the solid is

$$S=100-0.5T$$

The boundary conditions are

$$q=50 \quad \text{on } r=1 \qquad q=0 \quad \text{on } r=2$$

$$T=100(1+y) \quad \text{on } x=0 \qquad q=h(T-T_f) \quad \text{on } x=2$$

The primary emphasis in this problem is on the handling of boundary conditions. The constant temperature and adiabatic boundary conditions are applied with the same techniques used for `user0.c`. The boundary conditions at $r = 1$ (constant heat flux) and $x = 2$ (convective boundary condition) are new, and are described in Chapter 3.

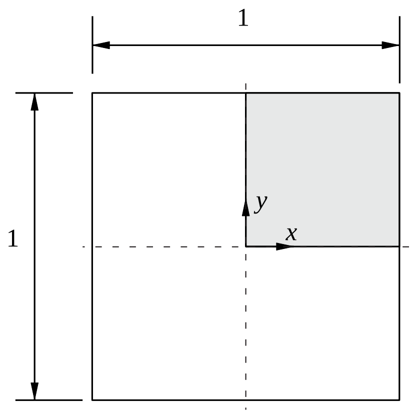

Figure 5.4: Fully-developed flow and heat transfer in a square duct.

Material Properties

Comparing Equation (5.1) with Equation (2.1) one finds that the thermal conductivity is the Γ variable. Within each control volume Γ is assumed to be uniform, but it can vary from control volume to control volume. This is implemented by giving each dependent variable (each ϕ) an associated array of Γ values. The array is pointed to by `gammaf[n]` as summarized in Table 2.1.

For example, the following snippet of code specifies value of $\Gamma = 3.2$ over a subset of control volumes

```
for (i=3; i<=7; i++)
    for (j=2; j<=6; j++)  gammaf[nPhi][i][j] = 3.2;
```

where `nPhi` must be an appropriate index into the master ϕ data structure (cf. Table 2.1). The variable Γ field is treated with the harmonic mean formulation [2] in the core part of the PnS code.

Fully-Developed Flow and Heat Transfer: `user3.c`

Highlights

- solve fully-developed flow problem with uniform wall temperature
- show how multiple ϕ equations can be solved at one time

Background for the solution to this problem is provide in Chapter 4.

The equation governing the flow in the duct is (since $u = v = 0$)

$$0 = \frac{\partial}{\partial x}\left(\mu\frac{\partial w}{\partial x}\right) + \frac{\partial}{\partial y}\left(\mu\frac{\partial w}{\partial y}\right) - \frac{dp}{dz}$$

Comparing this to the general φ equation leads to the observation that

$$\Gamma = \mu \qquad S_c = \frac{dp}{dz}$$

The friction factor and Reynolds numbers are defined by Equations (4.1) and (4.2), respectively. The flow is uncoupled from the heat transfer so the equation for w may be solved first. The equation for w requires an iterative solution

1. Specify an arbitrary dp/dz
2. Solve for the velocity distribution, w_{ij}
3. Compute the friction factor
4. Return to step 2 until f converges

The energy equation is

$$\rho c_p w \frac{\partial T}{\partial z} = \frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(k\frac{\partial T}{\partial y}\right)$$

For the constant wall temperature boundary condition the appropriate dimensionless temperature is

$$\Theta = \frac{T - T_w}{T_b - T_w}$$

where $T = T(x, y)$ is the temperature of the fluid in the duct at position (x, y), $T_w = T_w(z)$ is the wall temperature, and $T_b = T_b(z)$ is the bulk fluid temperature. For fully-developed flow Θ is independent of z. Solving the preceding equation for T gives

$$T = T_w + \Theta(T_b - T_w)$$

and from $\partial\Theta/\partial z = 0$ we get

$$\frac{\partial T}{\partial z} = \Theta\,\frac{\partial T_b}{\partial z}$$

The source term in the energy equation is, therefore,

$$S_C = -\rho c_p w(x, y)\Theta(x, y)\frac{dT_b}{dz}$$

The procedure for solving the energy (temperature) equation is

1. Choose a value of dT_b/dz. This corresponds to specifying an axial position in the duct.
2. Guess a T_{ij} field.
3. Calculate a Θ_{ij} distribution for use in the energy equation source term.
4. Calculate a new T_{ij} field by solving the discrete system of equations for the nodal T values.
5. Return to step 3 until convergence.

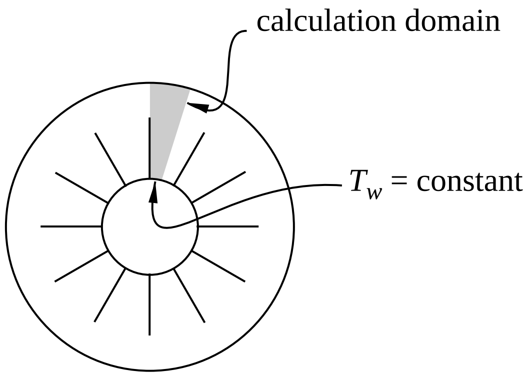

Figure 5.5: `user4.c` involves fully-developed flow in an annulus with radial fins. Because of the symmetry, only a small part of the flow needs to be analyzed.

Fully-Developed Flow with Conjugate Heat Transfer: `user4.c`

Background for the solution to this problem is provided in Chapter 4. Highlights:

- use polar coordinates
- solve fully-developed flow problem with uniform heat flux boundary condition
- solve a conjugate heat transfer problem
- evaluate the heat flux with the harmonic mean
- impose internal constraints
- demonstrate how to define a variable grid

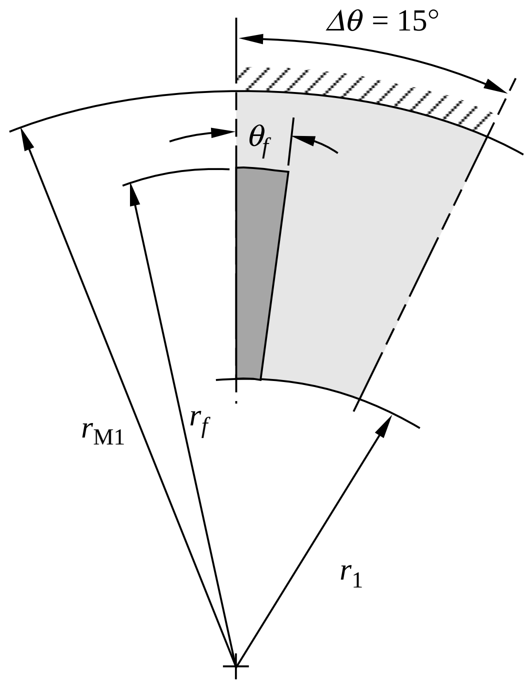

Figure 5.6: Detail of the thick radial fin as part of the calculation domain for `user4.c`.

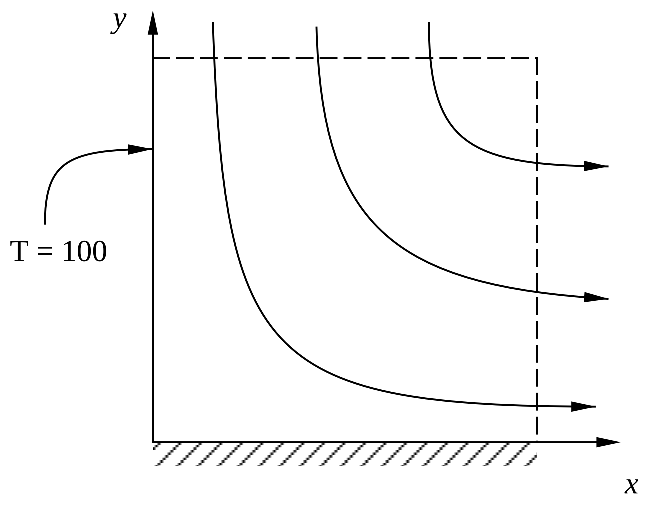

Figure 5.7: Flow in a corner with prescribed velocities.

Convective Transport in a Corner: `user5.c`

Highlights:

- solve convection without solving the flow field
- convective outflow boundary conditions
- energy balance

The flow situation is depicted by the sketch in Figure 5.7

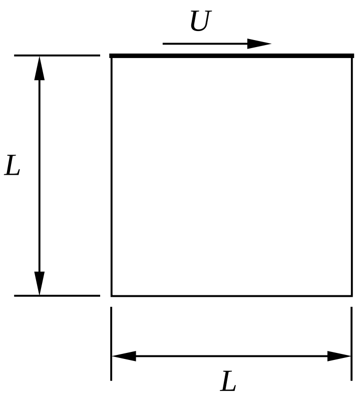

Figure 5.8: Flow in a driven cavity.

The Driven Cavity: `cavity.c`

Highlights:

- solve a flow field

The driven cavity is a standard benchmark problem for CFD codes. The physical problem is depicted by the sketch in Figure 5.8. The fluid occupies a square domain of dimension L. The top surface of the domain moves to the right with velocity U. The only parameter of this problem is the Reynolds number

$$Re = \frac{\rho U L}{\mu}$$

where ρ and μ are, respectively, the density and dynamic viscosity of the fluid.

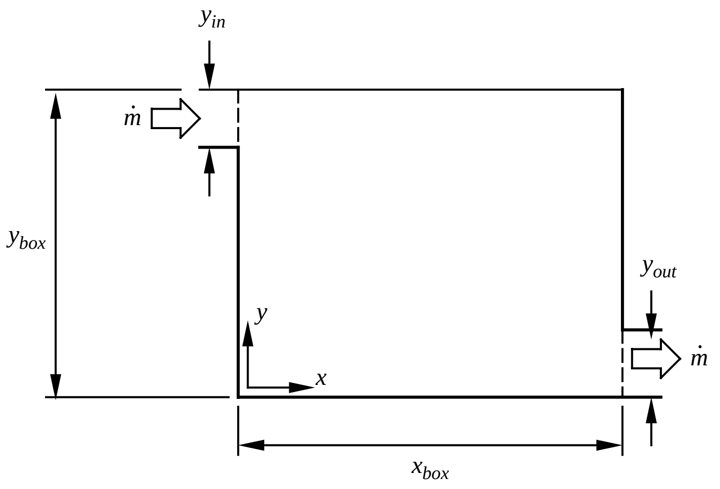

Figure 5.9: Flow through a box.

Flow through a Box: `box.c`

Highlights:

- solve a flow field
- specify inflow and outflow boundaries

This problem involves flow of water through a infinitely long chamber. There is one inlet and one outlet. Figure 5.9 is a schematic of this flow.

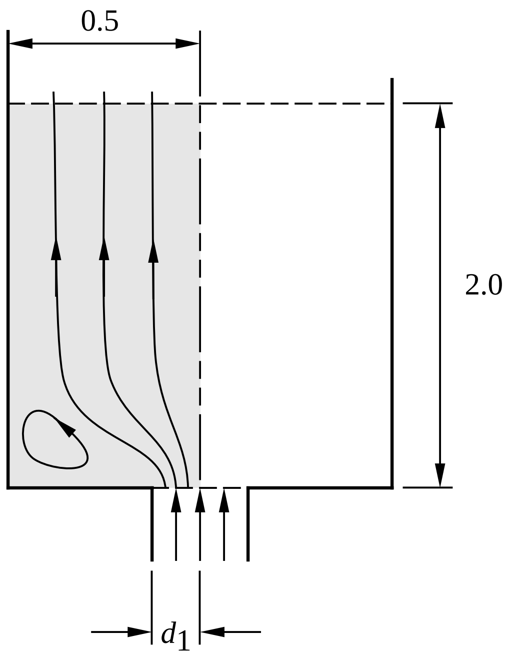

Figure 5.10: Sudden expansion in a channel. Because of symmetry only the left half of the problem needs to be simulated.

Sudden Expansion with Buoyancy Effects: `user6.c`

Highlights:

- solve buoyancy-driven flow: energy equation is coupled with the momentum and mass conservation equatoins
- convective outflow boundary conditions
- energy balance

The flow situation is depicted by the sketch in Figure 5.10

Bibliography

[1] J. H. Ferziger and M. Perić. *Computational Methods for Fluid Dynamics.* Springer-Verlag, Berlin, third edition, 2001.

[2] S. Patankar. *Numerical Heat Transfer and Fluid Flow.* Hemisphere, Washington D.C., 1980.