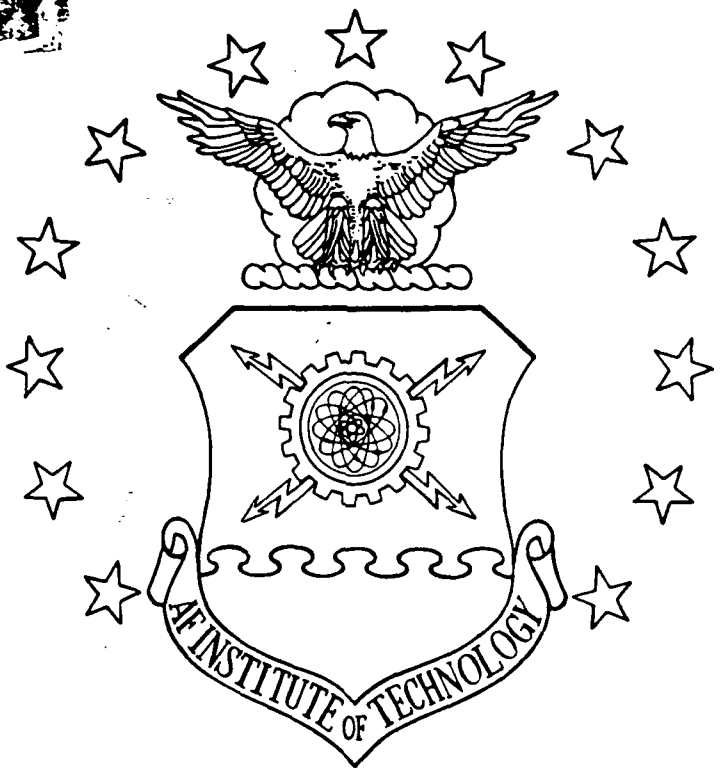

INCOMPRESSIBLE FLOW FRICTION FACTORS IN A SIMULATED HEAT PIPE

THESIS

James E. Mayhew
Captain, USAF

AFIT/GAE/ENY/89D-23

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GAE/ENY/89D-23

INCOMPRESSIBLE FLOW FRICTION FACTORS IN A SIMULATED HEAT PIPE

THESIS

James E. Mayhew
Captain, USAF

AFIT/GAE/ENY/89D-23

Approved for public release; distribution unlimited

AFIT/GAE/ENY/89D-23

INCOMPRESSIBLE FLOW FRICTION FACTORS IN A SIMULATED HEAT PIPE

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Aeronautical Engineering

James E. Mayhew, B.S.

Captain, USAF

December 1989

Acknowledgements

First, I would like to thank my advisor, Dr. James Hitchcock, for his guidance and encouragement during this project. Time after time I came to a standstill in my work, frustrated and convinced that no solution existed, and Dr. Hitchcock patiently got me going again, reminding me that research often involves trying a lot of things that don't work before finding one that does. His knowledge and experience will be sorely missed after he retires this year.

Second, thank-you to the other members of my thesis committee, Dr. William Elrod and Capt Daniel Fant, for taking the time to read and comment on this work.

Third, thank-you to Capts Stephen Zaiser and Mark Nowack, my classmates and friends, who by their friendship and encouragement have helped make this work enjoyable.

Thanks also to Capt Philip Beran for his suggestions on the numerical simulation program, and to my classmate Capt Richard Rockwell for sharing his broad knowledge of computing with me.

Finally, I thank my dear wife Mary, who has encouraged and believed in me during this often discouraging process of writing a thesis. The past 18 months have been very profitable for me largely because of her and the others mentioned, and I am grateful to God for bringing them into my path.

Soli Deo Gloria

James E. Mayhew

Table of Contents

List of Figures

List of Tables

Notation

A — area

f — friction factor, $\frac{\tau_w}{\frac{1}{2}\rho\bar{u}^2}$

fRe — product of friction factor and axial Reynolds number

L — length of pipe

NJ — number of radial grid points

p — static pressure inside the pipe

r — radial coordinate

R — inside radius of pipe

Re — axial Reynolds number, $\frac{\rho\bar{u}2R}{\mu}$

Re_w — radial Reynolds number, $\frac{\rho v_w 2R}{\mu}$

u — axial velocity

$\bar{u}$ — average axial velocity

v — radial velocity, positive out of pipe

v_w — radial velocity at pipe wall

z — axial coordinate

z/L — non-dimensional axial distance

Greek

Δr . . radial distance between grid points

Δz . . axial distance between grid points

μ . . . dynamic viscosity

ν . . . momentum diffusivity

ρ . . . density

τ_w . . shear stress at the pipe wall

ϕ . . . momentum flux factor, $\frac{\overline{u^2}}{\bar{u}^2}$

Abstract

Porous pipes with blowing and suction are used to simulate the vapor flow in heat pipes. A computer program for steady, two-dimensional, boundary-layer flow has been written to find friction factors using the axial pressure distributions measured in an experiment with incompressible flow in a porous pipe. The results for some existing experimental data are presented and compared to previously published solutions for fully-developed flow. Also, this program has been modified to numerically simulate porous pipe flow without using axial pressure distribution as an input. This 'simulation' program furnishes additional results for comparison to the original 'data reduction' program. It is observed that the simulation program accurately computes friction factors and flow separation points in a porous pipe with low radial Reynolds number.

INCOMPRESSIBLE FLOW FRICTION FACTORS IN A SIMULATED HEAT PIPE

I. Introduction

Background

A heat pipe is a device used to transfer large quantities of heat over finite distances with small temperature differences. It transfers heat by evaporating a working fluid at the hot end (evaporator), pushing the vapor to the cold end (condenser) with a small pressure gradient, condensing the vapor back into liquid in the condenser, and returning the liquid to the evaporator by means of a wick along the inside wall of the pipe (Figure 1).

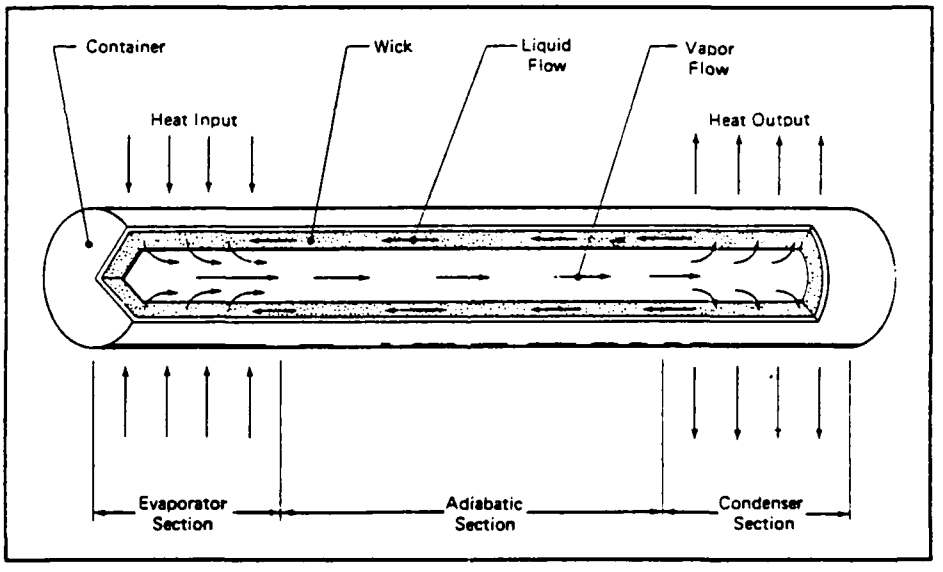

Figure 1. Conventional Heat Pipe (3:2)

For steady-state operation the liquid must be returned to the evaporator as fast as the vapor leaves it. If it does not, the heat pipe will dry out. One of the key factors in determining how fast the liquid returns to the evap-

orator is the frictional force between the vapor and the liquid, which flow in opposite directions. So, it is desirable to know the applicable friction factors when designing a heat pipe.

Since heat pipes are sealed from the environment, it is very difficult to use one to conduct an experiment to measure friction. Porous pipes with blowing and suction have been used to simulate the vapor flow in a heat pipe (Figure 2), and compute various flow characteristics, including friction factors.

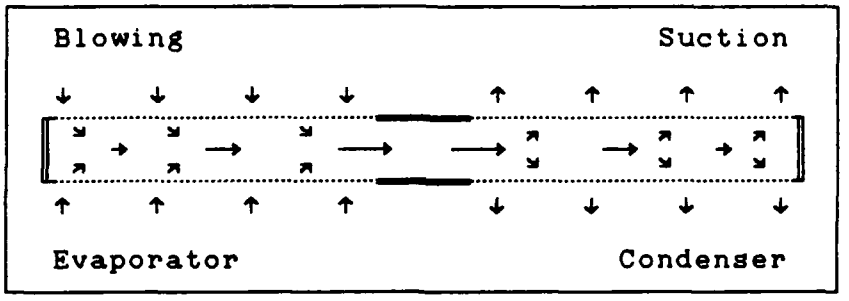

Figure 2. Simulated Heat Pipe

Literature Review

Kinney (4) found that for fully-developed flow, the friction factor in a porous pipe depends only on radial Reynolds number, Re_v. Re_v is based on the radial velocity at the wall (positive for blowing) and the internal diameter of the pipe. For fully developed flow, the velocity field is obtained from an ordinary differential equation. In the blowing region, the flow can become fully developed within a short distance of the beginning of the pipe, and Kinney's

results are useful there. However, the flow in the suction region is often undeveloped. In fact, Kinney showed that it is not possible for a flow to be fully developed if Re_v < -4.626. Since Kinney's results apply only to fully-developed flow, another method is necessary to compute friction factors in a porous pipe where the flow is not fully developed.

One method is to solve the full Navier-Stokes equations numerically, given Re_v as a boundary condition. Bowman (2) did this, and conducted a porous pipe experiment, for high Re_v ($100<Re_v<30000$). In solving the full Navier-Stokes equations, he found that the terms that are neglected for boundary-layer flow were very small in comparison to the other terms in the equations.

Manley (5) conducted a porous pipe experiment similar to Bowman's, but used low values of Re_v, from 1.8 to 12.6. Manley measured the axial pressure distribution in the pipe and used a modified one-dimensional numerical model to compute friction factors based on the pressures he measured. A significant assumption in his analysis was that fully developed velocity profiles could be used to determine the flow momentum. This was a good assumption for the blowing region, since the flow there develops quickly, but it was not accurate for the suction region, since the flow there is usually undeveloped.

Objective and Scope

The objective of this research is to create a two-

dimensional numerical model of Manley's porous pipe experiment, and use it to compute the friction factors in the pipe. Steady, two-dimensional, incompressible, boundary-layer flow with constant viscosity is assumed. This 2-D model will compute velocity profiles for the flow in the pipe, so it does not require the assumption of fully developed velocity profiles as Manley's 1-D model did.

Two methods of solution are used. First, a numerical solution of the governing equations is obtained using Manley's wall blowing and suction rates (i.e. Re_v) and his measured axial pressure distribution. This is known as the "data reduction" (DR) program. Second, this DR program is modified to compute the axial pressure distribution as well as the friction factors using Manley's wall blowing and suction rates as the only input to the program. This second solution method is known as the "simulation" (SIM) program.

The friction factors computed by the DR and SIM programs will be compared to each other, and to Kinney's friction factors for fully developed flow. Also, the pressure distribution computed by the SIM program will be compared to Manley's measured pressure distribution. Finally, the flow separation points computed by the SIM program will be compared to Manley's measured separation points.

II. Numerical Model

Governing Equations

The following equations govern the flow in the present analysis.

Continuity Equation

$$r \frac{\partial u}{\partial z} + \frac{\partial (rv)}{\partial r} = 0 \tag{1}$$

Equation of Motion in the Axial Direction

$$u \frac{\partial u}{\partial z} + v \frac{\partial u}{\partial r} = \frac{-1}{\rho} \frac{dp}{dz} + \nu \left[\frac{\partial^2 u}{\partial r^2} + \frac{1}{r} \frac{\partial u}{\partial r} \right] \tag{2}$$

with these boundary conditions

$$\begin{aligned} &u=0, \ r=R && v=0, \ r=0 \\ &\frac{\partial u}{\partial r} = 0, \ r=0 && v=v_w, \ r=R \end{aligned} \tag{3}$$

where

p = static pressure
r = radial distance
R = radius of pipe
u = axial velocity
v = radial velocity
v_w = radial velocity at wall of pipe
z = axial distance
ν = momentum diffusivity
ρ = density

The equations of continuity and motion shown above apply for a steady, incompressible, constant viscosity, axi-symmetric flow. The equation of motion is written in boundary-layer form, which is the result of the following assumptions.

$$\frac{\partial p}{\partial r} = 0 \tag{4}$$

so pressure is a function of z only, and

$$\frac{\partial^2 u}{\partial z^2} \ll \frac{\partial^2 u}{\partial r^2} + \frac{1}{r}\frac{\partial u}{\partial r} \qquad (5)$$

Bowman's numerical solutions to the full Navier-Stokes equations indicate that these are reasonable for all axial locations except near the ends of the pipe, where the radial velocities are the same order of magnitude as the axial velocities.

Solution Method

The objective of this research is to compute the friction factors, f, in a porous pipe. In order to compute f at some axial location z, the variation of u with r, i.e., the velocity profile, must be known at that z. The first task is to find the velocity profiles all along the pipe. As mentioned earlier, this is done two different ways, one using a 'data reduction' (DR) program, the other using a 'simulation' (SIM) program. First, the DR program will be described, then the SIM program, and finally the methods used to compute f once the velocity profiles are known.

Data Reduction (DR) Program. This method for computing friction factors in a porous pipe uses Manley's blowing and suction data and his experimentally measured axial pressure distribution. The pipe is divided into uniform grids as shown in Figure 3. Grids were kept uniform throughout the pipe and as close to square as geometry would permit.

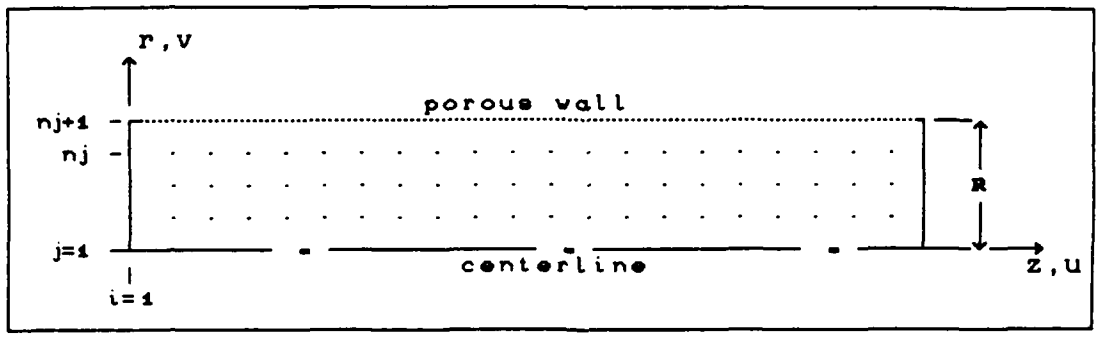

Figure 3. Pipe Divided into Grids

The equation of motion is written for a small control volume surrounding each grid point, shown in Figure 4. When the axial velocity gradient $\frac{\partial u}{\partial z}$ is written as the first-order upwind difference

$$\left.\frac{\partial u}{\partial z}\right]_{i,j} = \frac{u_{i,j} - u_{i-1,j}}{\Delta z} \tag{6}$$

the control volume expressions reduce to the following finite difference equations (FDE)

Interior Points (j=2 to j=nj)

$$u_{i,j}\left[\frac{u_{i,j} - u_{i-1,j}}{\Delta z}\right] + \left[\frac{v_{i,j+1} + v_{i,j}}{8}\right]\left[2 + \frac{\Delta r}{r_j}\right]\left[\frac{u_{i,j+1} - u_{i,j}}{\Delta r}\right]$$

$$+ \left[\frac{v_{i,j} + v_{i,j-1}}{8}\right]\left[2 - \frac{\Delta r}{r_j}\right]\left[\frac{u_{i,j} - u_{i,j-1}}{\Delta r}\right] + \left.\frac{dp}{dz}\right]_i \left[\frac{Re}{2}\right]_i$$

$$= \nu\left[\left[\frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{(\Delta r)^2}\right] + \frac{1}{r_j}\left[\frac{u_{i,j+1} - u_{i,j-1}}{2\Delta r}\right]\right] \tag{7}$$

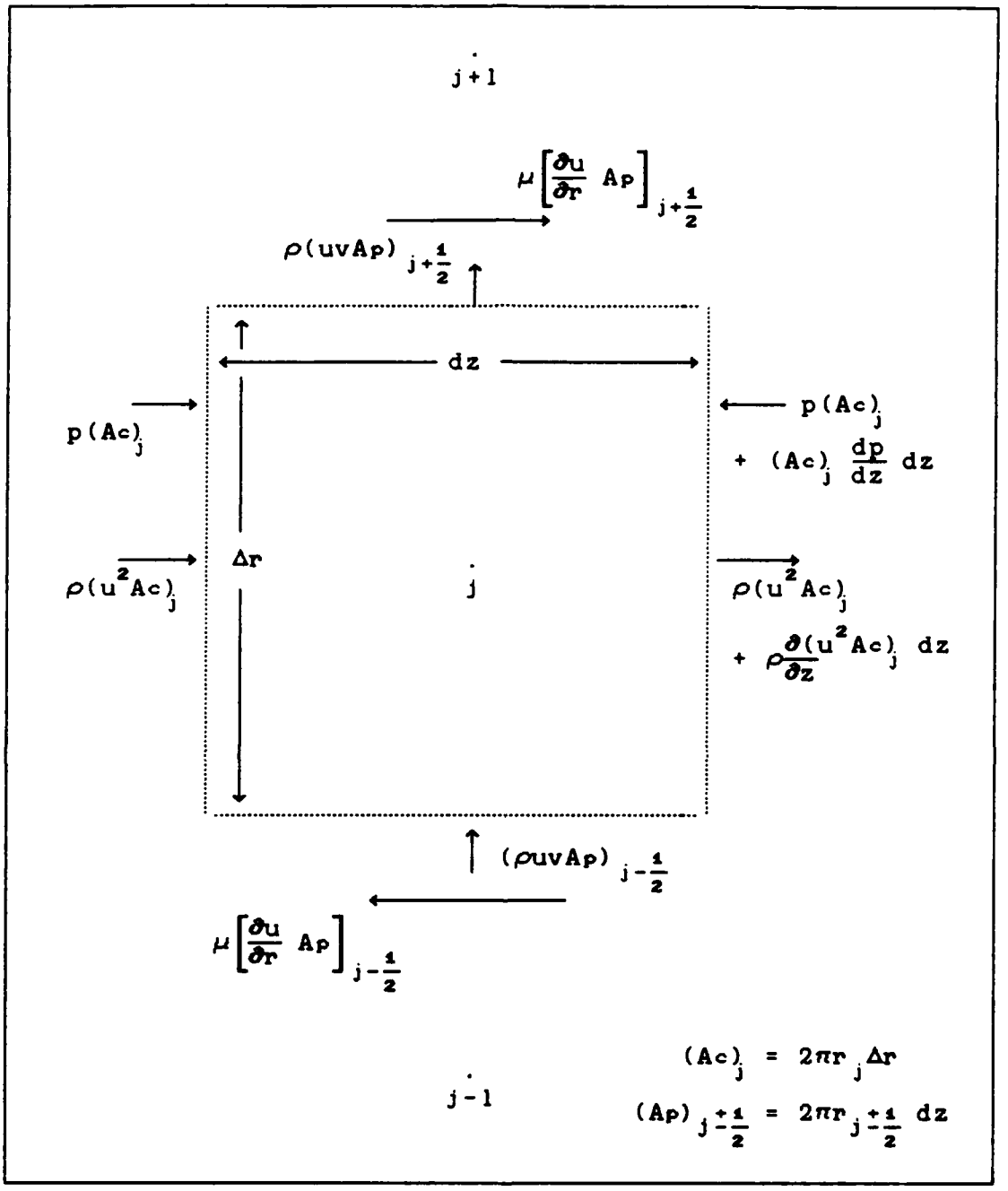

Figure 4. Control Volume Around Grid Points

Centerline Point (j=1)

$$u_{i,j}\left[\frac{u_{i,j} - u_{i-1,j}}{\Delta z}\right] + v_{i,j+1}\left[\frac{u_{i,j+1} - u_{i,j}}{\Delta r}\right]$$

$$= \left.\frac{dp}{dz}\right)_i \left[\frac{-Re}{2}\right]_i + 4\nu\left[\frac{u_{i,j+1} - u_{i,j}}{(\Delta r)^2}\right] \qquad (8)$$

Notice that equations 7 and 8 require the pressure gradient at each discrete point i. Manley's pressure measurements were taken at 30 axial locations along the pipe, so to get the pressure gradient at each discrete point i from his data, a curve fit of his data is necessary. A computer program was written to fit every set of three pressure readings to a quadratic function

$$p(z) = c_o + c_1 z + c_2 z^2 \qquad (9)$$

and the derivative with respect to z was taken analytically to give

$$\frac{dp}{dz} = c_1 + 2c_2 z \qquad (10)$$

Notice also that equations 7 and 8 are non-linear in the unknowns u and v. To linearize the equations, the coefficients u and v on the convective terms can be 'lagged,' i.e., written at the i-1,j location when the rest of the equation is written at i,j (1:337).

The equations are linearized in this manner and non-dimensionalized, and after separating knowns from unknowns become

$$B_{i,j}\, u_{i,j-1} + D_{i,j}\, u_{i,j} + A_{i,j}\, u_{i,j+1} = C_{i,j} \qquad (11)$$

where for the interior points

$$B_{i,j} = \left[-TL1_{i,j} - \frac{2}{\Delta r^2} + \frac{1}{\Delta r r_j}\right]$$

$$D_{i,j} = \left[\frac{u_{i-1,j}}{\Delta z} RV_i Re_i - TL2_{i,j} + TL1_{i,j} + \frac{4}{\Delta r^2}\right]$$

$$A_{i,j} = \left[TL2_{i,j} - \frac{2}{\Delta r^2} - \frac{1}{\Delta r r_j}\right]$$

$$C_{i,j} = \frac{dp}{dz}\Big)_i \left[\frac{-Re}{2}\right]_i + \frac{1}{\Delta z}\left[RV_i u_{i-1,j}\right]^2 Re_i$$

and for the centerline point

$$B_{i,j} = 0$$

$$D_{i,j} = \left[\frac{u_{i-1,j}}{\Delta z} - \frac{v_{i-1,j+1}}{\Delta r}\right] RV_i Re_i + \frac{8}{(\Delta r)^2}$$

$$A_{i,j} = \frac{v_{i-1,j+1}}{\Delta r} RV_i Re_i + \frac{8}{(\Delta r)^2}$$

$C_{i,j}$ = same as interior points shown above

with

$$TL1_{i,j} = \left[\frac{v_{i-1,j} + v_{i-1,j-1}}{8\Delta r}\right]\left[2 - \frac{\Delta r}{r_j}\right] RV_i Re_i$$

$$TL2_{i,j} = \left[\frac{v_{i-1,j} + v_{i-1,j+1}}{8\Delta r}\right]\left[2 + \frac{\Delta r}{r_j}\right] RV_i Re_i$$

$$Re_i = \frac{2R\bar{u}}{\nu}_i \qquad RV_i = \frac{\bar{u}_{i-1}}{\bar{u}_i} \qquad \frac{dp}{dz}\Big)_i = \underset{(dim)}{\frac{dp}{dz}\Big)_i} \frac{2R}{\rho \bar{u}_i^2}$$

Δr, r_j, Δz non-dimensionalized by R

$u,v)_{i,j}$ . . . $\bar{u}_i$

$u,v)_{i-1,j}$ . . . $\bar{u}_{i-1}$

This gives a tri-diagonal system of nj equations, and is solved implicitly for u using the Thomas algorithm (1:549-550).

Knowing u, v can be found by solving the continuity equation. Only nj-1 values of v are unknown, since v is known at the wall and the centerline. Therefore, only nj-1 equations are needed to find the remaining unknown v's. Using the same control volumes that were used for the equations of motion (Figure 4), the continuity equation (Equation (1)) is written in finite difference form for only the interior points, j=2 to j=nj, giving

$$\frac{u_{i,j} - u_{i-1,j}}{\Delta z} = \left[\frac{v_{i,j-1} + v_{i,j}}{2\Delta r}\right]\left[1 - \frac{\Delta r}{2r_j}\right] - \left[\frac{v_{i,j+1} + v_{i,j}}{2\Delta r}\right]\left[1 + \frac{\Delta r}{2r_j}\right] \qquad (12)$$

Note that the boundary conditions v=0, r=0 and $v=v_w$, r=R are still imposed, since these values of v are used when writing Equation (12) at j=2 and j=nj, respectively. After non-dimensionalizing and separating knowns from unknowns, this equation becomes

$$v_{i,j-1}\left[\frac{1}{4r_j} - \frac{1}{2\Delta r}\right] + v_{i,j}\left[\frac{1}{2r_j}\right] + v_{i,j+1}\left[\frac{1}{4r_j} + \frac{1}{2\Delta r}\right] = u_{i,j}\left[\frac{-1}{\Delta z}\right] + u_{i-1,j}\left[\frac{RV}{\Delta z}_i\right] \qquad (13)$$

When written at all nj-1 points, equation (13) also forms a tri-diagonal system of linear equations, and is solved for v using the Thomas algorithm again.

Having computed both u and v at some i, the solution 'marches' to the next i and repeats the process. As the solution marches, a check on overall mass flow through the cross-section can be performed. Since the velocity profile, $u_{i,j}$, is computed at each i, this velocity profile can be numerically integrated using the trapezoid rule to find the average velocity, and therefore mass flow, at that i. The mass flow at any i is also computed from the known blowing and suction rates at the wall, since all the air entering through the wall becomes part of the flow through the pipe. This mass flow is treated as a known, since it is computed directly from experimental data. The two are compared at several axial locations to ensure that the computed velocity profile is consistent.

This marching solution continues to the end of the pipe, or to the point where the flow separates, whichever comes first. For further details, see Appendix A for a listing of the DR program.

Starting Problem. Since the equation of motion in boundary-layer form does not apply near the ends of the pipe, the solution must be started part way down, where u dominates v. This was estimated to be five percent of the pipe length, i.e., at z/L = 0.05. Since u and v at (i-1,j) are required in the linearized equation of motion at

(i,j), u and v at (i-1,j) must be known to start the marching solution. The axial velocity profile at i-1 was approximated with a fourth-order polynomial $u(r) = a + br + cr^2 + dr^3 + er^4$. The five unknown coefficients a,b,c,d,e were found by satisfying the following boundary conditions:

(1) $\frac{\partial u}{\partial r} = 0, \quad r=0$

(2) $u=0, \quad r=R$,

(3) overall mass flow through the cross-section, known from the blowing at the wall,

$$\rho 2\pi R \int_0^z v_w dz = \rho 2\pi \int_0^R u(r)\, r\, dr$$

(4) the equation of motion at the wall

$$v_w \frac{\partial u}{\partial r} = -\frac{1}{\rho}\frac{dp}{dz} + \nu\left[\frac{\partial^2 u}{\partial r^2} + \frac{1}{r_j}\frac{\partial u}{\partial r}\right]_{wall}$$

and

(5) $\frac{\partial^2 u}{\partial z^2} = 0 \quad , \quad r=0$

This last assumption is also used at the edge of an external boundary-layer flow. So, knowing u from this approximate profile, v was found using Equation (13).

It has been shown how the velocity profiles were calculated using the DR program, and how the calculations were started 5% of the way down the pipe. Before showing how f is computed from the velocity profiles, the SIM program will now be described.

Simulation Program. The second method used to find

friction factors, called the SIM program, uses only Manley's blowing and suction rates, not his axial pressure distribution. This adds one more unknown to the problem, the pressure gradient, so one more equation is needed. To get one more equation, the continuity equation was written for a different set of control volumes, centered about the half-points shown in Figure 5.

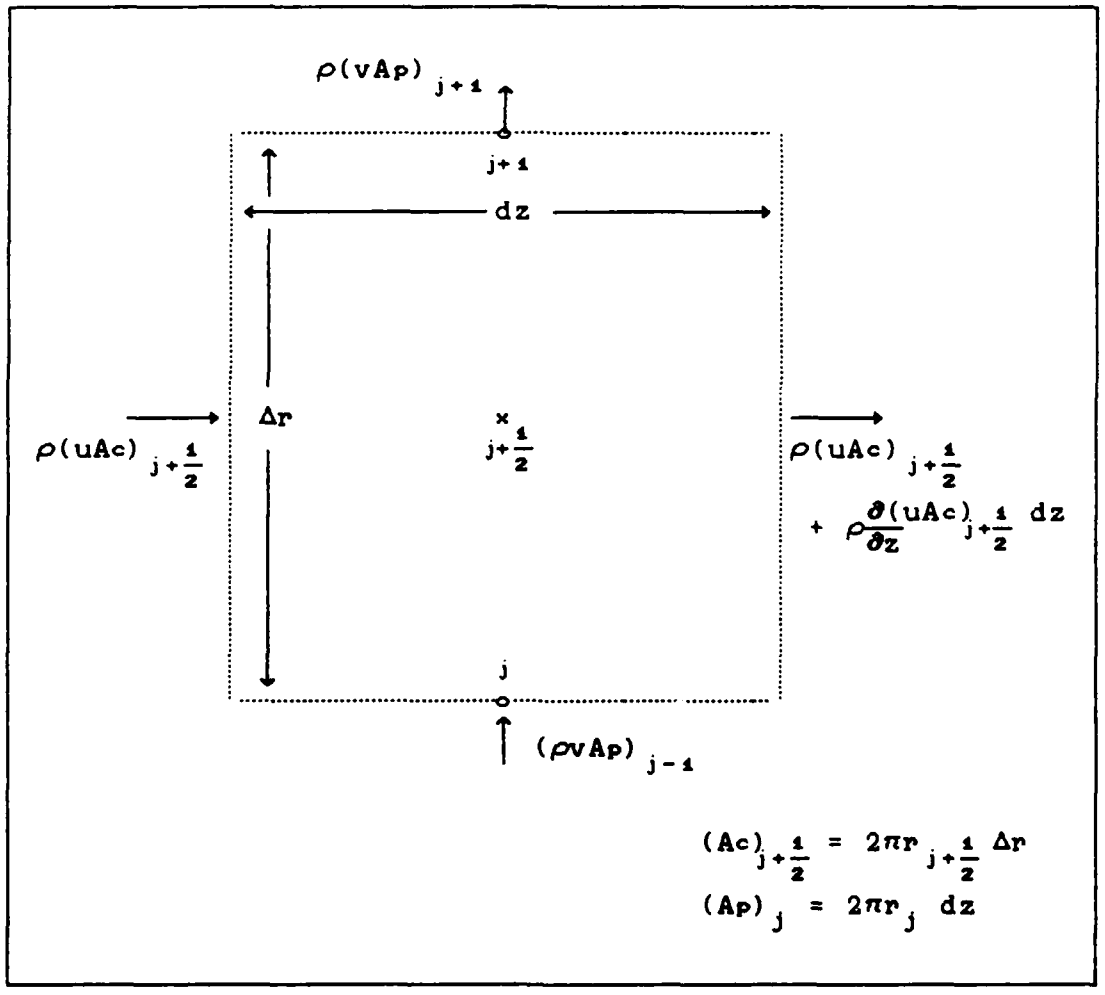

Figure 5. Control Volume Around Half-Points

This reduces to the following finite difference expression

$$\left[\frac{u_{i,j} + u_{i,j+1} - u_{i-1,j} - u_{i-1,j+1}}{2\Delta z}\right]\left[r_{j+1} + r_j\right]\Delta r$$

$$+ \quad 2\left[(rv)_{i,j+1} - (rv)_{i,j}\right] \quad = \quad 0 \qquad (14)$$

and after non-dimensionalizing and separating knowns and unknowns, becomes

$$u_{i,j} + v_{i,j}\left[ZR_j\ r_j\right] + u_{i,j+1} + v_{i,j+1}\left[ZR_j\ r_j\right]$$

$$= u_{i-1,j} + u_{i-1,j+1} \qquad (15)$$

$$\text{where } ZR_j = \frac{4\Delta z}{(r_{j+1}+r_j)\Delta r}$$

The same equations of motion that were used in the DR program are used here; but, now that pressure gradient is an unknown, the equations of continuity and motion are coupled. The coupled equations are linear in the unknowns u, v, and $\frac{dp}{dz}$, and must be solved simultaneously for every axial station i using a general matrix solver. An LU decomposition routine (6:31-38) was used to solve the matrix of equations. As with the first solution method, the second method also begins 5% of the way down the pipe with the quartic velocity profile approximation. For further details, see Appendix B for a listing of the SIM program.

The pressure gradient computed in this manner can be

numerically integrated to find pressures, and then compared to Manley's measured pressures. For an additional comparison, the DR program can be run using the pressure gradients computed by the SIM program. This checks to see if DR gives the same values of f when they both use the same pressure gradients.

Now velocity profiles are available at every station i using two different methods: one by the DR program, using Manley's blowing, suction and axial pressure data, the other by the SIM program, using only Manley's blowing and suction data. The objective now is to compute f at any i given the velocity profile there.

Friction Factors. For fully-developed flow, the product of f and Re is constant for a given uniform Re_v, as shown in ref. (5). So in the following, fRe is determined. The first way to compute fRe is by writing the equation of motion for a small control volume at the wall, shown in Figure 6, in the same way as was done to develop finite difference expressions (FDE) to find u.

The equation of motion has not been written at the wall yet because up to this point it didn't give useful information. Notice, however, that in the equation of motion for j = nj (equation (7)), the wall boundary condition $v_{nj+1} = v_v$ is still imposed on the FDE, since the average value of v must be taken at the $nj+\frac{1}{2}$ edge of the control volume.

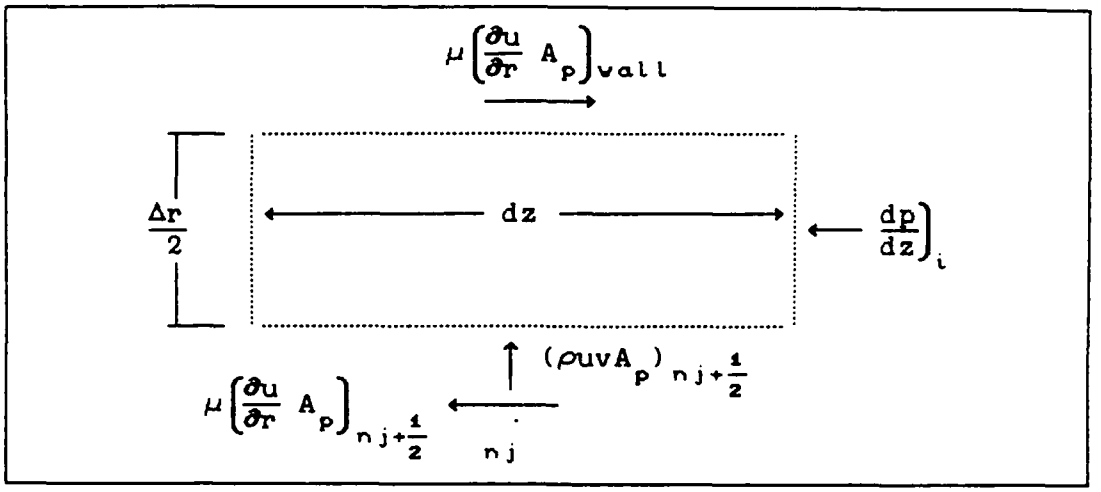

Figure 6. Control Volume at Wall

The above control volume expression reduces to the following non-dimensional equation

$$fRe)_i = -u_{i,nj}\left[\frac{Re_i}{2}\left(v_{i,nj+1} + v_{i,nj}\right) + \frac{4}{\Delta r}\right]\left(1 - \frac{\Delta r}{2}\right)$$

$$+ \left.\frac{dp}{dz}\right]_i Re_i\left[\frac{\Delta r}{2} - \frac{(\Delta r)^2}{8}\right] \qquad \text{where } fRe)_i = \left[\frac{2\tau_w}{\rho \bar{u}_i^2}\right] Re_i \qquad (16)$$

to give fRe as a function of known quantities.

The second method of computing fRe is by using a three-point, second-order accurate FDE for the velocity gradient at the wall (1:44)

$$\left.\frac{\partial u}{\partial r}\right]_w = \frac{3u_{nj+1} - 4u_{nj} + u_{nj-1}}{2\Delta r} \qquad (17)$$

This is computed directly once u is known, and fRe is found from

$$fRe = \frac{\tau_w}{\frac{1}{2}\rho\bar{u}^2} Re = \frac{\mu \left(\frac{\partial u}{\partial r}\right)_w \rho\bar{u}2R}{\frac{1}{2}\rho\bar{u}^2 \mu} = \frac{\left(\frac{\partial u}{\partial r}\right)_w 4R}{\bar{u}} \qquad (18)$$

Finally, the third way to compute fRe is by writing the equation of motion for a control volume that spans the entire diameter of the pipe, as shown in Figure 7. The control volume equation reduces to

$$fRe_{i\sim i+1} = \left(\frac{1}{1+RV_i}\right)\left(\frac{1}{2\Delta z}\right)\left[4\left(\phi_{i+1}Re_{i+1} - \phi_i Re_i RV_i\right)\right. \qquad (19)$$

$$\left. + \Delta z\left(\frac{dp}{dz}\right)_{i+1} Re_{i+1} + \left(\frac{dp}{dz}\right)_i Re_i RV_i\right] \qquad RV = \frac{\bar{u}_i}{\bar{u}_{i+1}}$$

and it is seen that the momentum flux factor, ϕ, is needed at i and i+1. This is computed in both computer programs for every i using a numerical integration across the velocity profile, and is easily substituted into the Equation (19) to get fRe.

So, it has been shown how the data reduction (DR) program and the simulation (SIM) program compute the velocity profiles in the pipe, and how fRe's are computed once the velocity profiles are known. The DR program uses Manley's pressure data and solves tri-diagonal systems of equations; the SIM program treats pressure gradient as an unknown, and solves general systems of equations with an LU decomposition matrix solver. This requires significantly more computer time than the DR program.

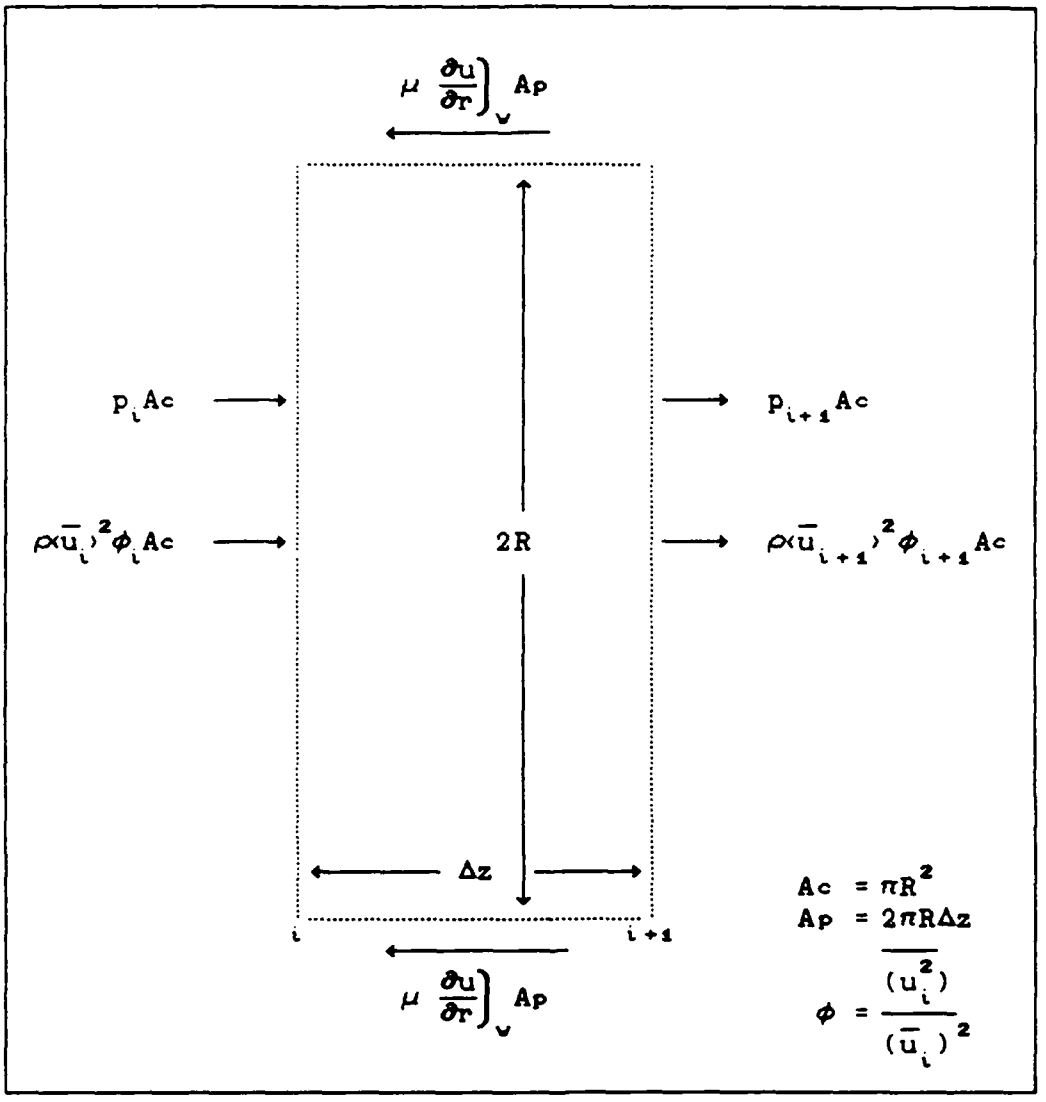

Figure 7. Control Volume Across Pipe Diameter

III. Results

Figure 8 shows three velocity profiles in a pipe, one for blowing, one for suction, and one for no blowing or suction (NBS). The blowing profile and suction profile shown were computed by the SIM program for $Re_v = 12.6$; the NBS profile shown is for fully-developed flow and has a parabolic shape, as can be shown analytically. The figure shows that the velocity at the centerline is smallest for the blowing case and largest for the suction case, with the NBS case in between. The figure also shows that the velocity gradient at the wall, $\frac{\partial u}{\partial r}\Big)_v$, is largest for the blowing case and smallest for the suction case, again with the NBS case in between. This means that the wall friction τ_v is greater for blowing than for suction, since $\tau_v = \mu \frac{\partial u}{\partial r}\Big)_v$.

This is opposite to the effect of blowing or suction on a flat plate in a free stream, where blowing decreases wall friction. This is due to the fact that in a pipe, the larger pressure gradient associated with blowing tends to accelerate the flow near the wall more than the flow elsewhere in the profile, resulting in a larger velocity gradient at the wall. However, on a flat plate in a free stream, no pressure gradient is present, and the blowing slows the flow near the plate, resulting in a smaller velocity gradient there.

Figures 9 through 12 show how fRe varies along the pipe. These figures show that in the blowing region ($0 < z/L < 0.49$), the fRe's computed by the SIM program compare well

with Kinney's fRe's for fully developed flow. Figures 9 and 10 show that in the two lowest Re_v cases, the flow eventually becomes fully developed in the suction region ($0.51 < z/L < 1.00$), and also compares well with Kinney's solutions. The fact that the fRe's computed by the SIM program compare so well with Kinney's fRe's strongly suggests that the SIM program accurately models porous pipe flow, and can be used to compute fRe in the developing-flow regions, where Kinney's solutions do not apply.

Figures 9 through 12 also show that the fRe's computed by the DR program are consistently lower than the fRe's computed by the SIM program. The check on mass flow indicated that the DR program had significantly less mass flow in the suction region than was computed from the wall blowing and suction rates. To find out if something was wrong with the DR program, the DR program was run using the pressure gradients computed by the SIM program rather than using Manley's. The resulting fRe's and separation points were identical to those computed in the SIM program (Figures 9-12). This shows that the DR program computes fRe as accurate as the SIM program does, provided they are given the same pressure gradients, and suggests a problem with Manley's pressure data.

Figures 13 through 16 show the pressure gradients computed by the SIM program and the 'measured' pressure gradients (actually the pressures were measured and curve fit to get pressure gradients). These figures show that in the blowing region, the 'measured' pressure gradients are only slightly

more positive than the computed pressure gradients. However, in the suction region, the 'measured' pressure gradients are significantly more positive than the computed pressure gradients, especially for the higher Re_v cases. This explains the difference in fRe's computed by DR and SIM, since a more positive pressure gradient implies less friction. This can be seen from Equation 16 and Figure 6. The pressure and friction forces must balance with the rate of momentum change. For a loss of momentum (suction), the pressure force and the friction force together act to cause this loss of momentum. If the loss of momentum is held constant while the pressure force is increased, the friction force must decrease to keep the forces balanced with the momentum loss. Similarly, for an increase of momentum (blowing), the pressure force both ov' comes the friction force and increases the momentum. If the pressure force is decreased (i.e., made more positive), but the momentum increase is held constant, the friction must decrease to keep the forces balanced with the increase of momentum.

The pressure gradients computed by the SIM program were integrated numerically using the trapezoid rule to get pressures. Figure 17 shows these 'computed' pressures and Manley's measured pressures. Consistent with the more positive pressure gradients seen in Figues 13 through 16, the measured pressures are higher than the computed pressures, and moreso for the higher Re_v cases. This difference between measured and computed pressures is the root cause of the difference between fRe's computed by DR and SIM. If the

measured pressures matched the computed pressures, then the pressure gradients would match, the velocity profile would be consistent with the mass flow, and the fRe's computed by DR and SIM would be the same. It is not clear why the measured and computed pressures differ, but three possible explanations are offered.

First, the pressure measurements could be incorrect. The guaranteed accuracy of the pressure transducer Manley used was +/-0.01 in. H_2O, which corresponds to p/p_o of +/-0.000025, i.e. over half of the largest pressure difference measured! However, two facts suggest that the transducer was much more accurate than it was guaranteed to be. First, the measured pressures follow a smooth curve, but one would expect a lot of fluctuations if the transducer was at the limit of its accuracy. Second, the measured and computed pressures compare very well in the blowing region, so the transducer apparently worked well there. It appears, then, that errors due to the accuracy of the transducer did not cause the difference in pressures shown in Figure 17.

Second, leaks in the pipe may have caused the difference in pressures. If some mass entered undetected near the end of the blowing region of the pipe, it follows that a greater pressure force in the suction region would be necessary to decelerate that extra mass. Manley did find and seal some leaks in the suction region, but it is possible that he missed some.

Third, the no-slip condition at the wall used in the numerical model may not be accurate for porous pipe flows.

It is not clear how slip at the walls would affect pressure, but it seems it would affect both pressure and friction. Recall that the pressure and friction forces must balance with the momentum change. If the flow velocity at the wall is some positive value, that implies that some axial momentum leaves the flow through the wall. That is, in the suction region, the mass extracted at the wall is not extracted perpendicular to the wall, but rather slightly angled in the direction of the axial flow. This loss in axial momentum out the wall means that less axial momentum remains in the pipe, and therefore less friction and pressure forces are required to slow it down. However, it is possible that due to the slip at the wall, the friction may decrease significantly, more than the axial momentum decreased, and so require a greater pressure force to balance with the momentum loss.

This is, of course, speculation, but it is not presently known how slip affects friction and pressure rise. It is a reasonably simple task to find out by modifying the SIM program to assume some slip at the wall.

One additional result concerning separation points is shown in Figures 11 and 12. The flow in the suction region for the higher two Re_v cases did not become fully developed, but instead fRe decreased until the flow separated. The separation points for these two cases are compared to Manley's measured separation points in Table 1.

The SIM program computed separation points very close to Manley's. Manley did not measure an exact point of

Table 1 Separation Points (z/L)

Re_ν	Manley's Data	DR program	SIM program
6.5	0.908 ~ 0.940	0.742	0.917
12.6	0.814 ~ 0.845	0.685	0.813

separation; he only knew that separation occurred after one pressure tap and before the next, as can be seen by the break in the smooth pressure rise in Figure 17. The DR program predicted separation much earlier than the SIM program, which is consistent with the more positive pressure gradients used in the DR program.

Figure 18 compares the three methods of computing fRe, and shows that they give answers very close to each other. The wall control volume method is preferred, and was used throughout. It is preferred because it was developed in the same way the finite-difference expressions used to solve the governing equations were developed, and therefore complements them.

Finally, Figure 19 shows the effect of decreasing the grid size used in the numerical solution. Two things are evident from this figure. First, that the solution converges for decreasing grid size, and second, that the solution for 25 grid points differs very slightly from that for 50 grid points. Since 50 grid points took several hours longer to run, 25 grid points were used throughout.

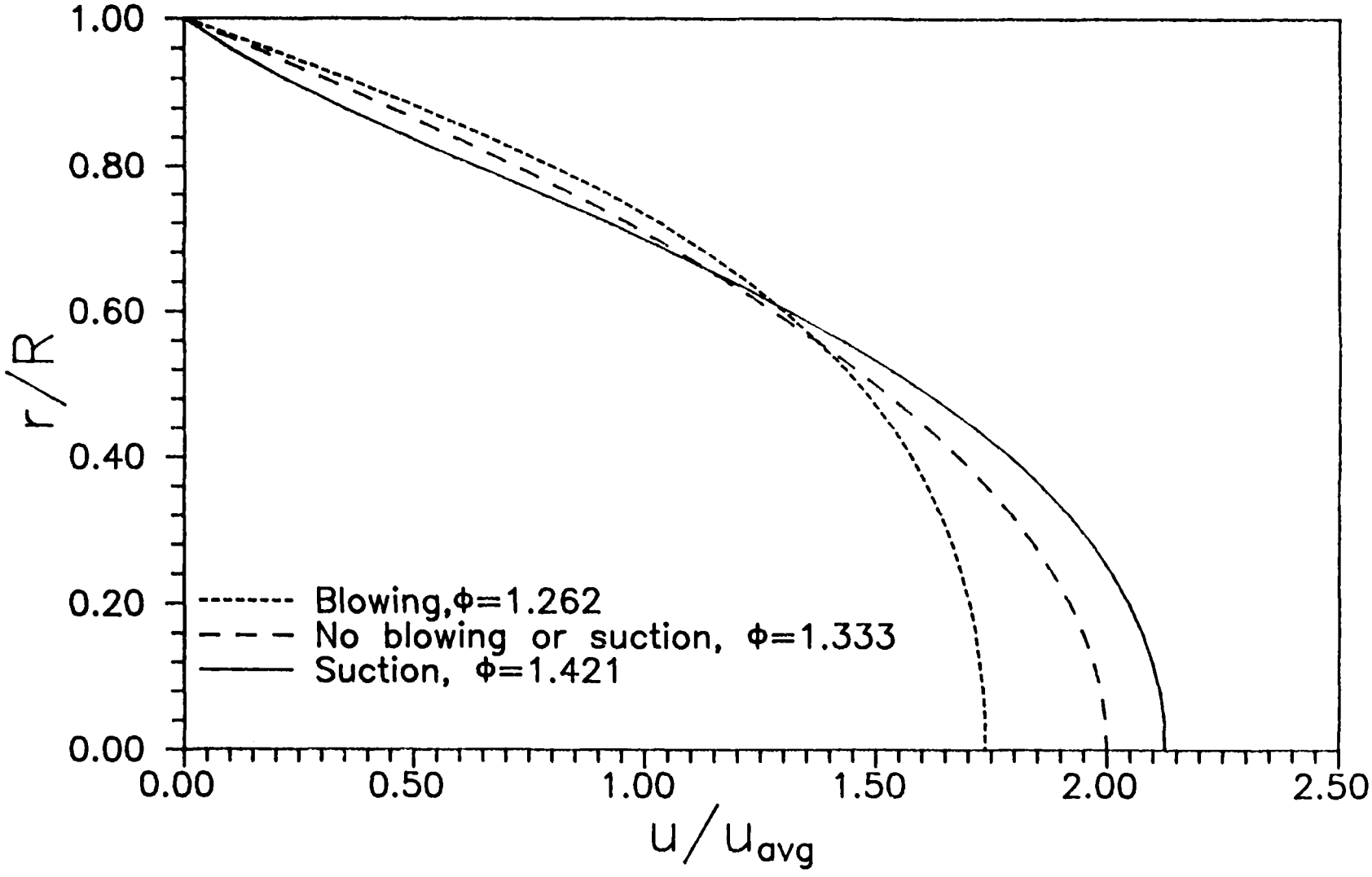

Figure 8. Comparison of Velocity Profiles

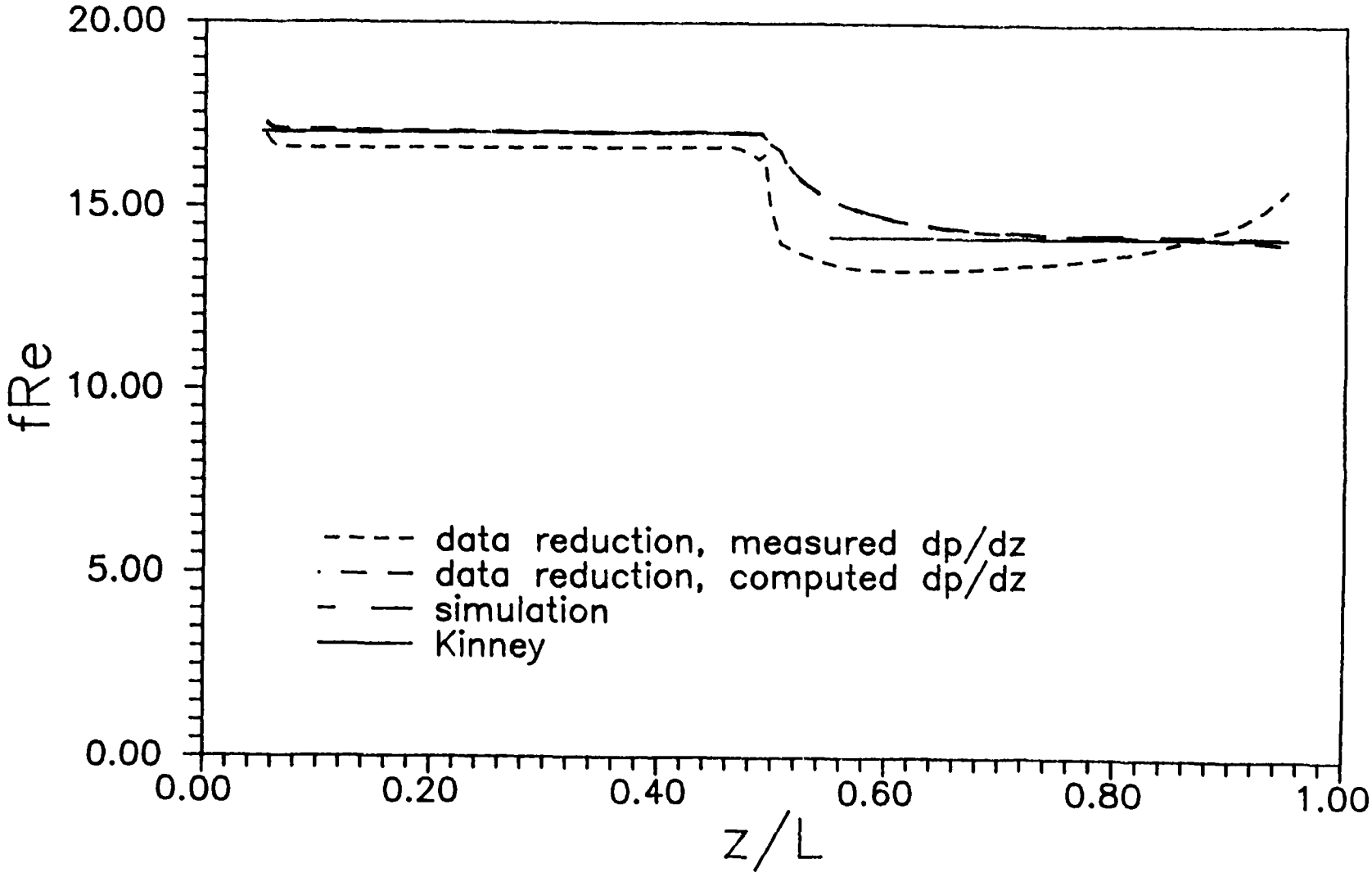

Figure 9. Friction Factor – Reynolds Number Distribution, Re_w=1.8

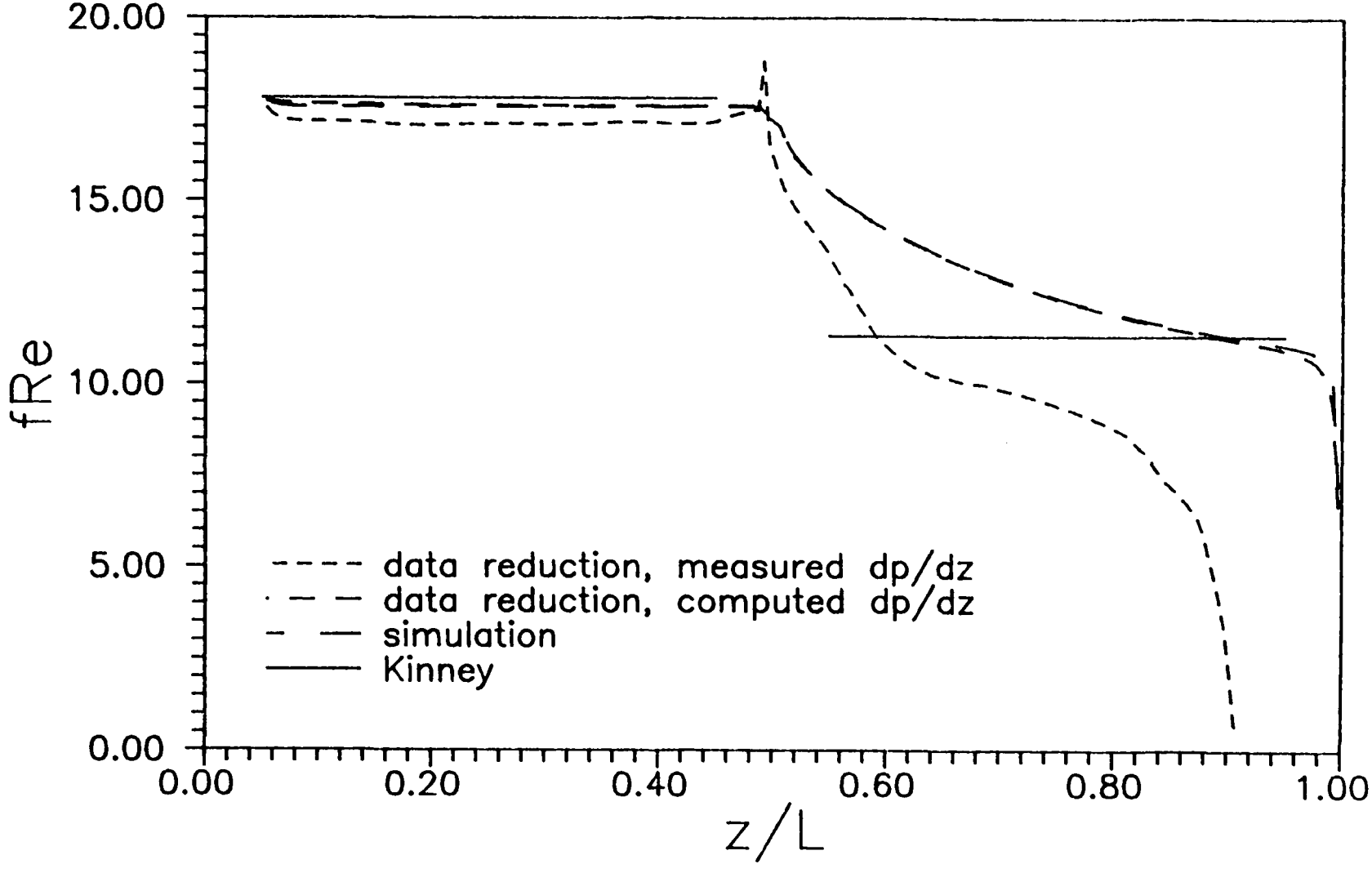

Figure 10. Friction Factor – Reynolds Number Distribution, Re_w=3.5

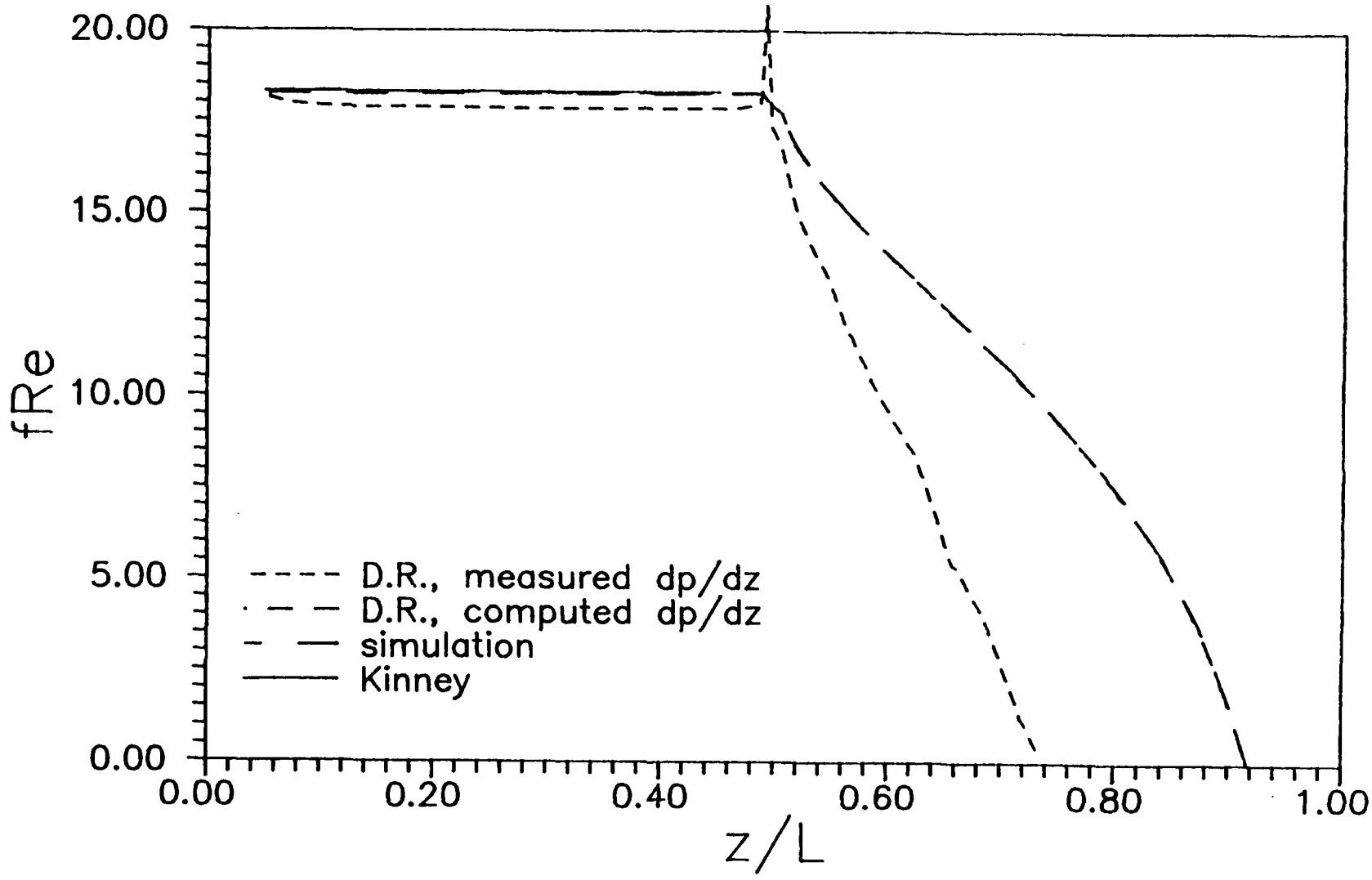

Figure 11. Friction Factor – Reynolds Number Distribution, Re_w=6.5

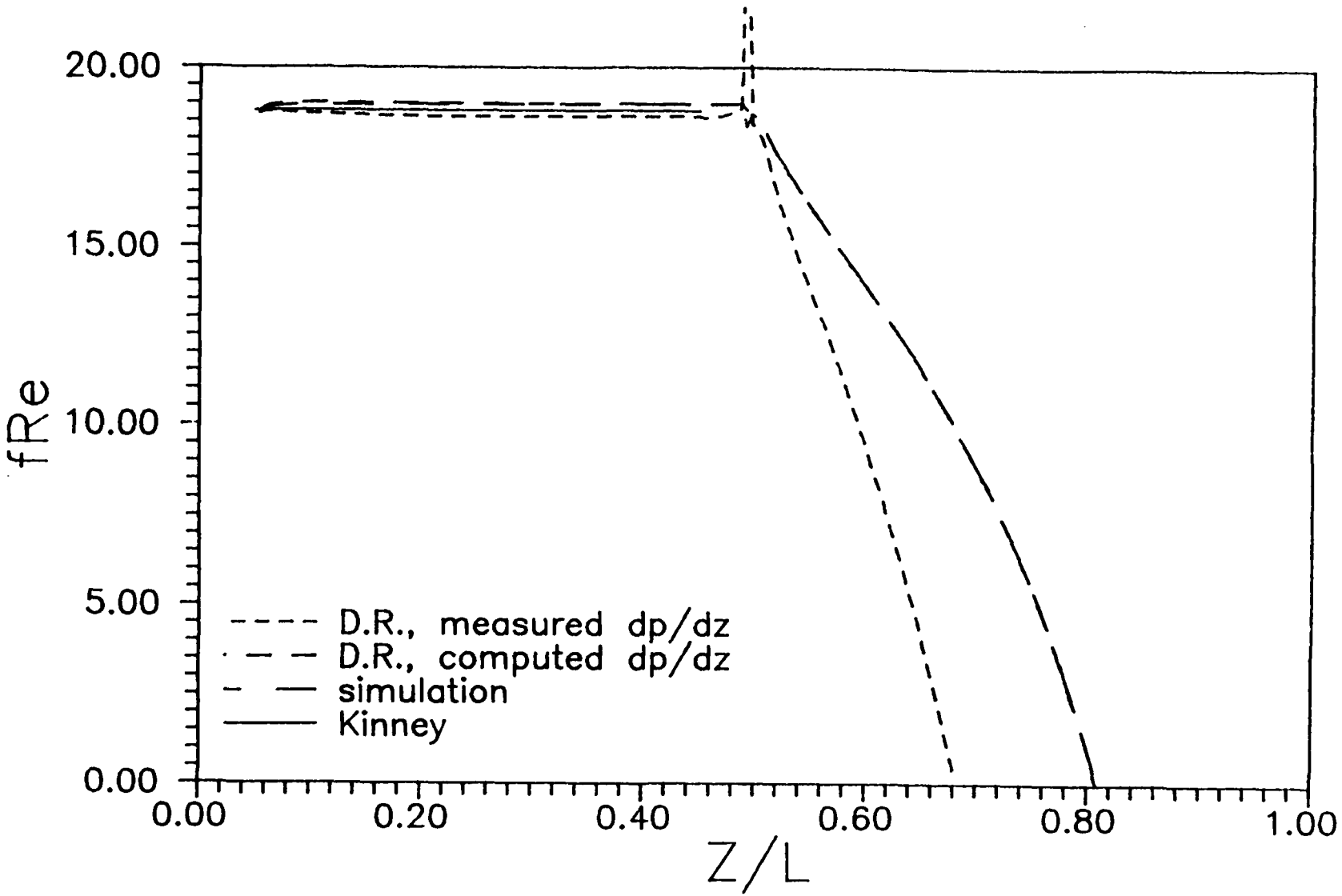

Figure 12. Friction Factor – Reynolds Number Distribution, Re_w=12.6

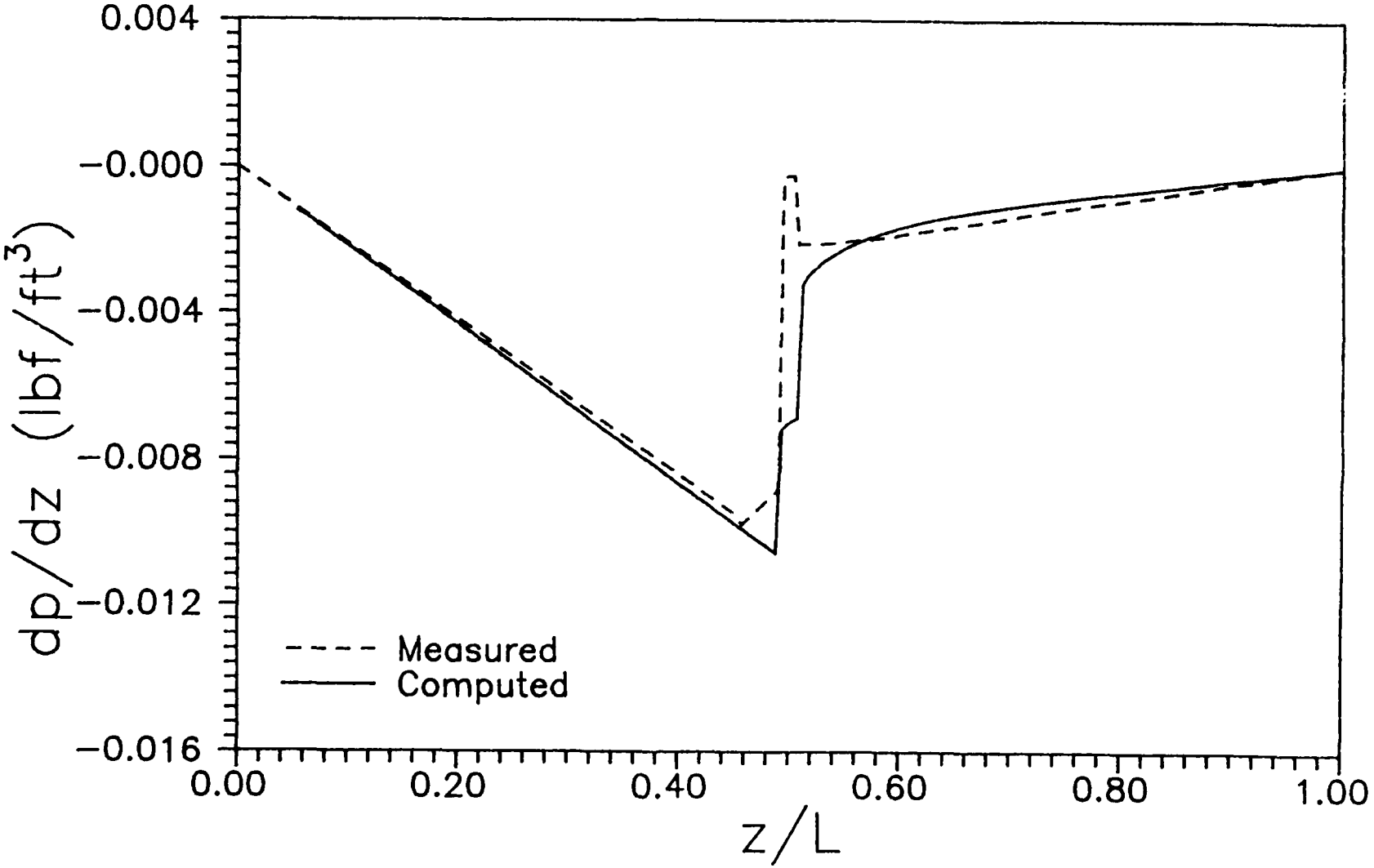

Figure 13. Pressure Gradient Distribution, Re_w=1.8

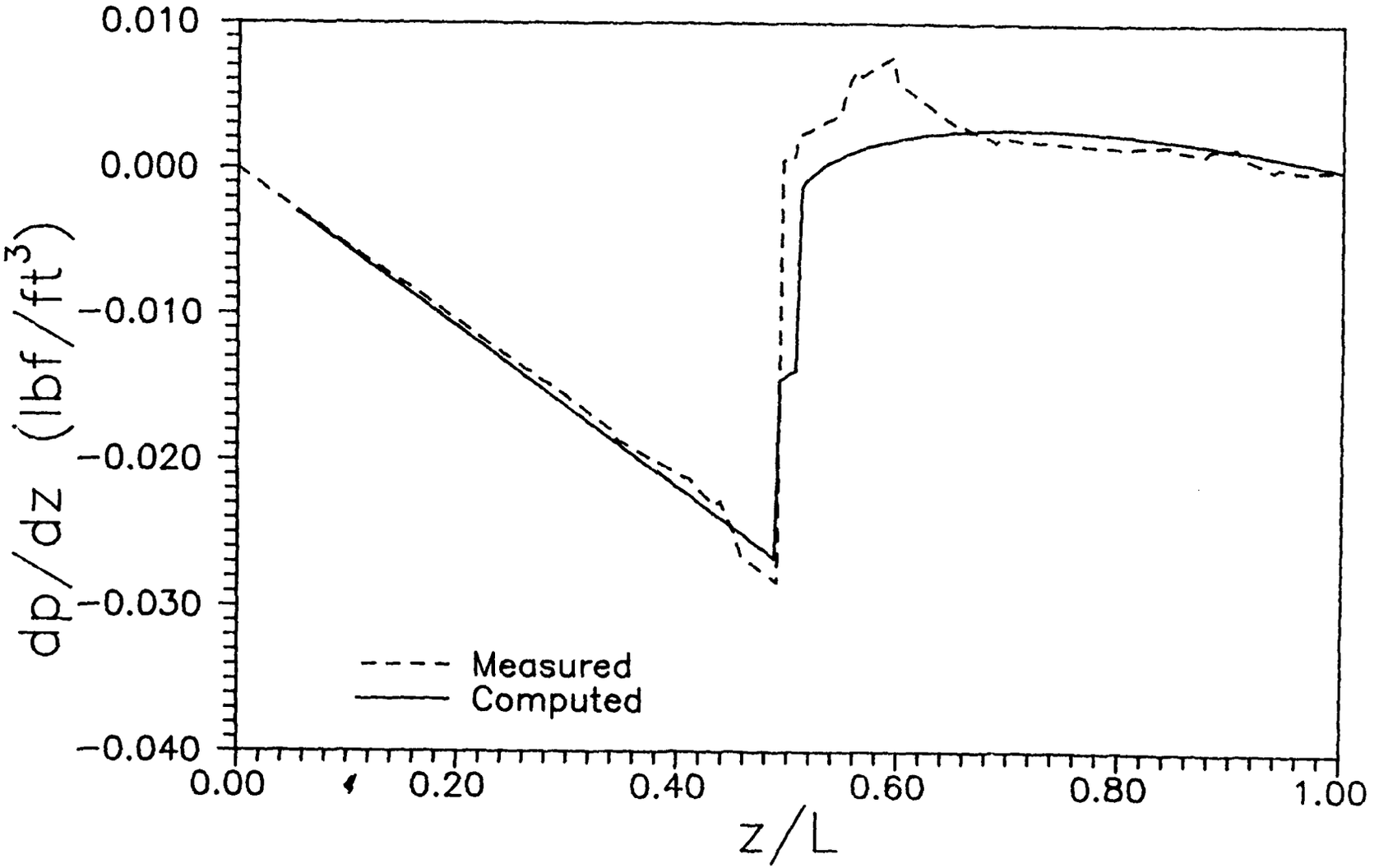

Figure 14. Pressure Gradient Distribution, Re_w=3.5

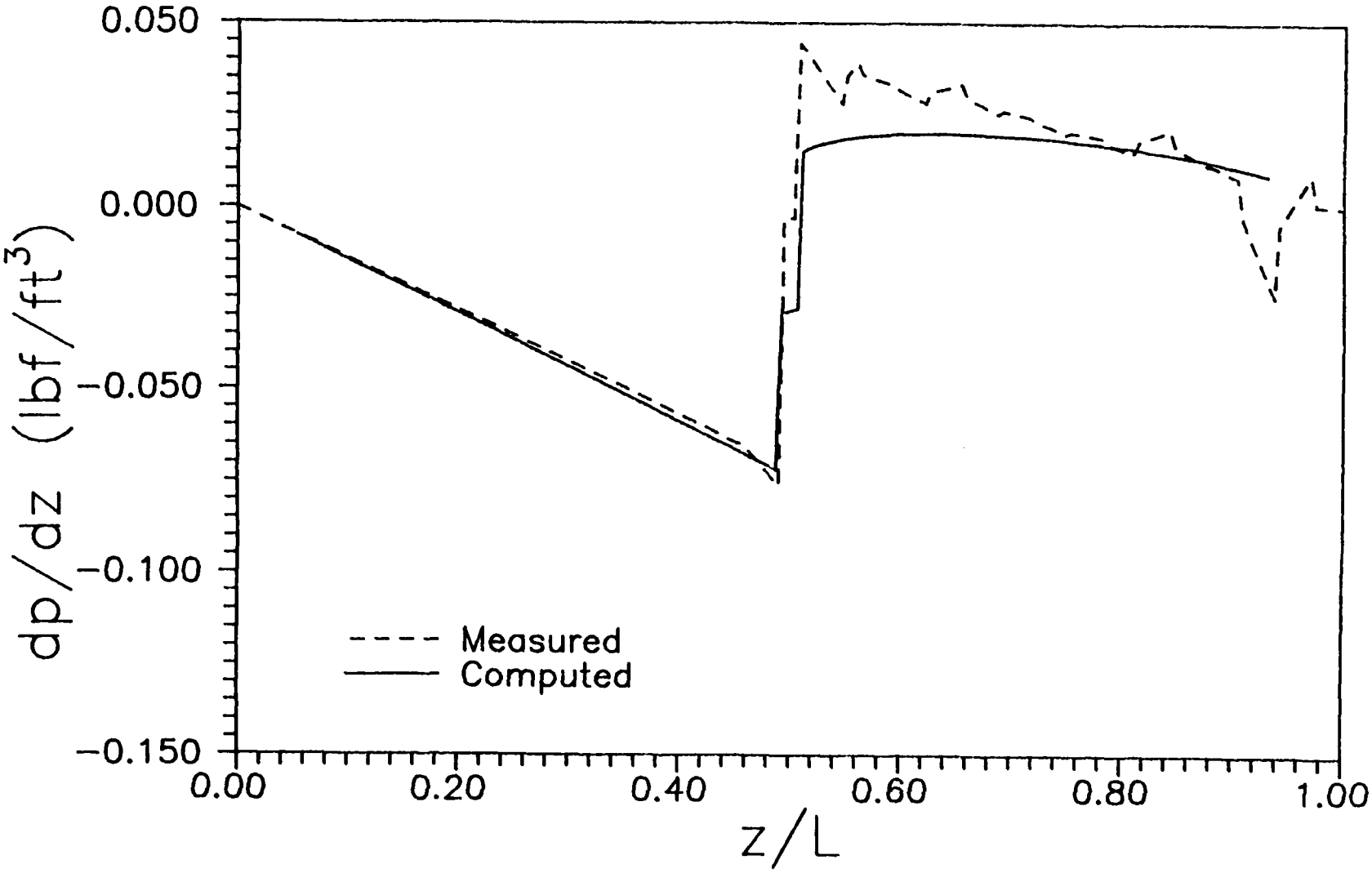

Figure 15. Pressure Gradient Distribution, Re_w=6.5

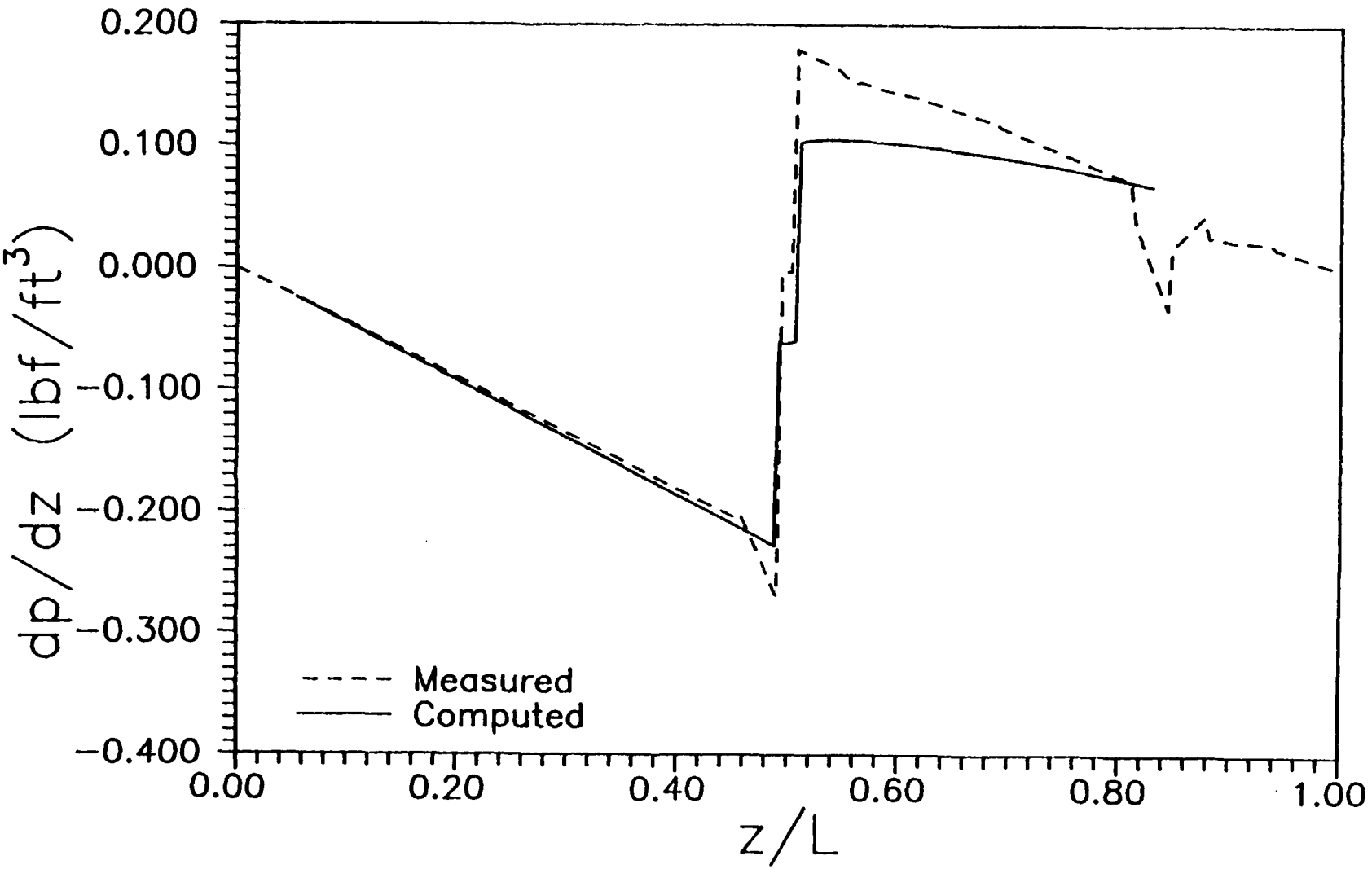

Figure 16. Pressure Gradient Distribution, Re_w=12.6

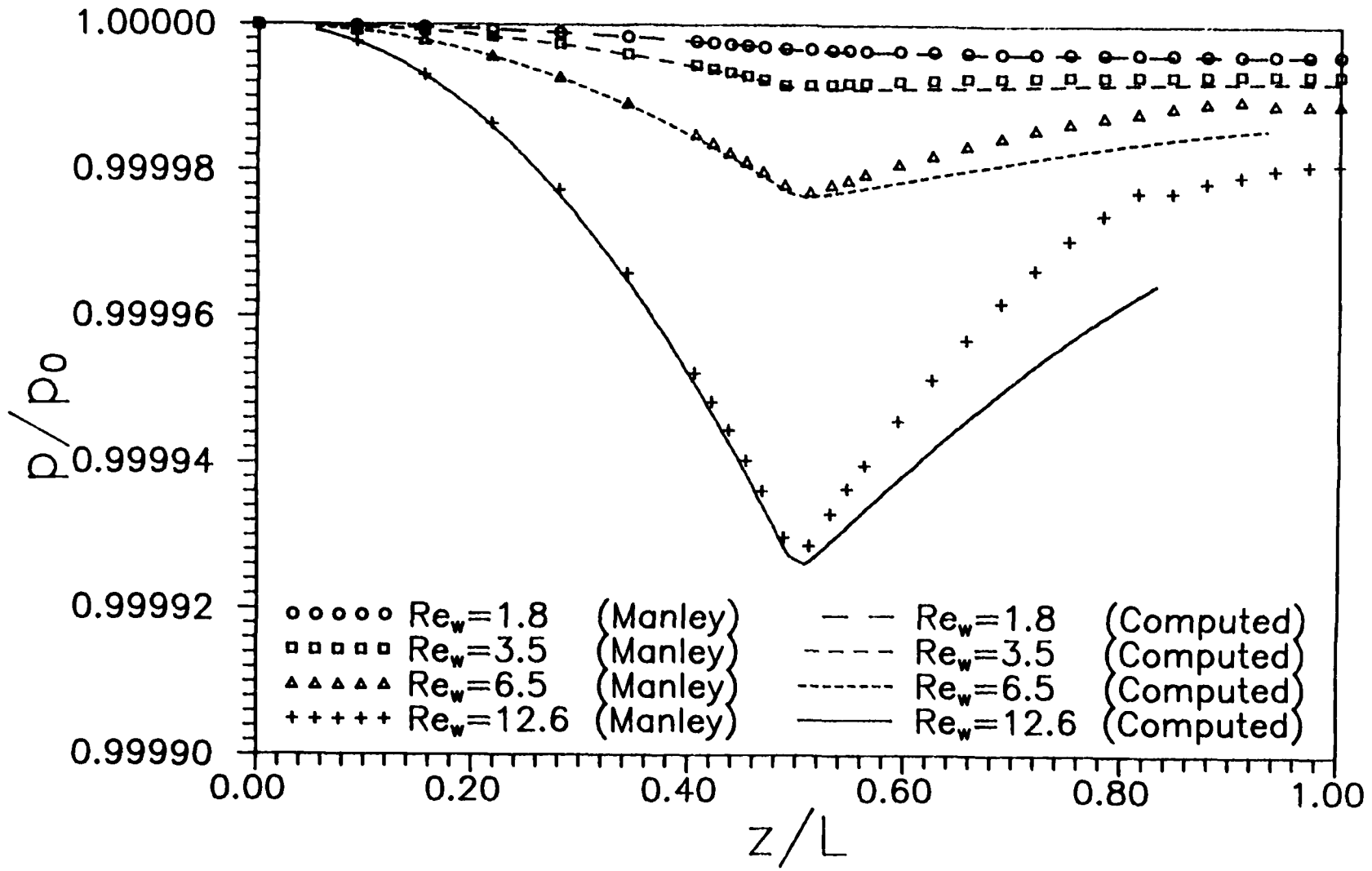

Figure 17. Axial Pressure Distributions

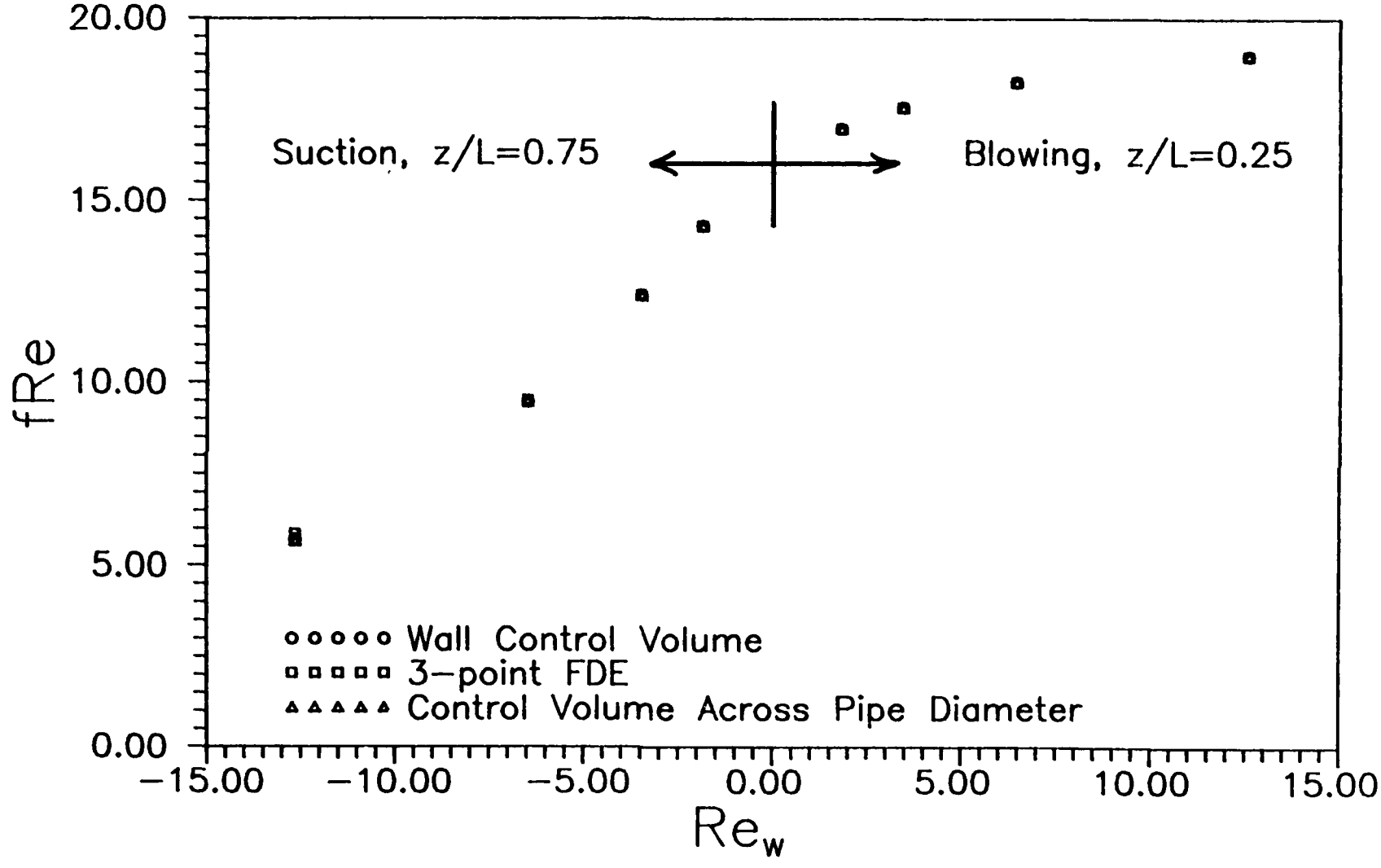

Figure 18. Comparison of Methods of Computing fRe

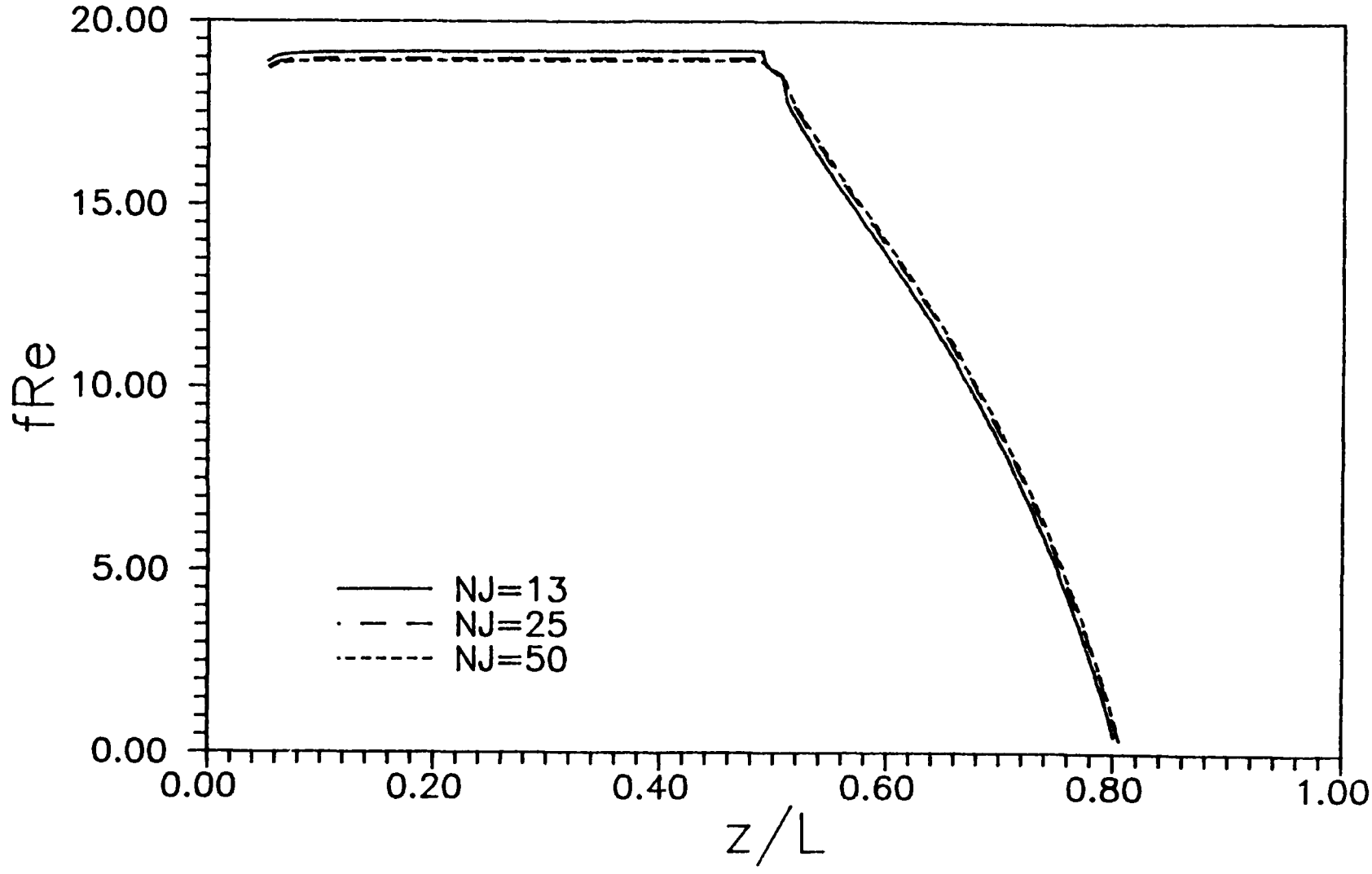

Figure 19. Radial Grid Size Comparison, Re_w=12.6

IV. Conclusions and Recommendations

Conclusions

The SIM program accurately computes the friction factors and separation points in Manley's porous pipe experiment. The boundary-layer assumption made in developing the SIM program is valid. The linearized equation of motion was also accurate, and it was not necessary to try to iteratively update the lagged coefficients to improve the answers.

The DR program accurately computes friction factors and separation points when it is given correct pressure gradients. The 'measured' pressure gradients used in the DR program were more positive than the ones computed in the SIM program, causing the fRe's to be lower.

Some difference exists between the measured pressures and the computed pressures, but it did not cause the fully-developed fRe's in the suction region (Re_v=1.8,3.5) to differ from Kinney's solutions nor the separation points (Re_v=6.5,12.6) to differ from Manley's measured separation points.

Recommendations

1. Run the SIM program assuming some slip at the wall in the suction region, to see if the computed pressures increase.
2. Determine how close to the beginning of the pipe the marching solution can be started and still give accurate results. Five percent was chosen as a conservative value

for this work.

3. Try running SIM and DR using simpler approximations for the velocity profiles at the starting point. A cubic, parabolic, linear, or even flat profile may work, and they would be easier to compute than the quartic profile.

Appendix A

```
      PROGRAM DR
C
C     THIS PROGRAM COMPUTES fRe AND PHI ASSUMING INCOMPRESSIBLE FLOW
C
C     DEFINITIONS OF VARIABLES
C
C     ANU             MOMENTUM DIFFUSIVITY
C     COEF4D          COEFFICIENT OF QUARTIC APPROXIMATION
C     COEF4E                .       .       .       .
C     DPDZ            AXIAL PRESSURE GRADIENT
C     DR              DIFFERENTIAL DISTANCE IN RADIAL DIR'N.
C     DZ                    .       .     . AXIAL   .
C     DZRW3           VELOCITY GRADIENT AT WALL FROM 3-PT DERIV
C     fRe             f*Re
C     FRE3            fRe from 3-POINT DERIVATIVE
C     FRECV           fRe from CONTROL VOLUME
C     FREPHI          fRe from PHI
C     NE              NUMBER OF EQUATIONS
C     NGADIA          # GRIDS IN ADIABATIC REGION
C     NGCOND          # GRIDS IN CONDENSER, = NGEVAP
C     NGEA            NGEVAP + NGADIA
C     NGEVAP          # GRIDS IN EVAPORATOR, = NGCOND
C     NGTOT           TOTAL # GRIDS ALONG AXIS OF PIPE
C     NJ              # RADIAL GRIDS
C     PHI             MOMENTUM FLUX FACTOR
C     R(J)            RADIUS
C     RE              AXIAL REYNOLDS NUMBER
C     REW             RADIAL REYNOLDS NUMBER
C     RHO             DENSITY
C     RR              RADIUS OF PIPE
C     RV              VZBM(I-1)/VZBM(I)
C     VR              RADIAL VELOCITY
C     VW              WALL RADIAL VELOCITY
C     VZ              AXIAL VELOCITY
C     VZ2B            AVERAGE OF SQUARE OF VELOCITY (FROM PROFILE)
C     VZB             AVERAGE VELOCITY (FROM PROFILE)
C     VZBM            AVERAGE VELOCITY (FROM VW)
C     VZBMN           AVERAGE VELOCITY AT NEXT I
C     VZBMP           VZBM AT PREVIOUS I
C     ZL              NON-DIMENSIONAL AXIAL DISTANCE Z/L
C
C
      IMPLICIT DOUBLE PRECISION(A-H, O-Z)
      CHARACTER*8 DATAFILE
C
      DIMENSION VR(12800,26),VZ(12800,26),DPDZ(12800),R(26),
     +  VW(12800),BD(26),D(26),C(26),A(26),VRL(26),VZL(26)
C
C     ****************************************
C
```

```
C       GRID SIZE
C
        WRITE(*,*) 'ENTER # GRIDS IN R-DIRECTION'
        READ(*,*) NJ
 78     WRITE(*,*) 'ENTER # GRIDS IN EVAP (783,1566,3132,6264,12528)'
        READ(*,*) NGEVAP
C
        IF (NGEVAP .EQ. 783) THEN
            NGADIA = 25
        ELSEIF (NGEVAP .EQ. 1566) THEN
            NGADIA = 50
        ELSEIF (NGEVAP .EQ. 3132) THEN
            NGADIA = 100
        ELSEIF (NGEVAP .EQ. 6264) THEN
            NGADIA = 200
        ELSEIF (NGEVAP .EQ. 12528) THEN
            NGADIA = 400
        ELSE
            GO TO 78
        ENDIF
C
        NGCOND = NGEVAP
        NGTOT = NGEVAP + NGADIA + NGCOND
C
        RR = 2.5D-1/12.0D0
        DR = 1.0D0/NJ
        DZ = 15.66D0/2.5D-1/NGEVAP
        ZR = DZ/DR/DR
C
C
C
C       ****************************************
C
C       READ PRESSURE DATA  - VW,DPDZ,ANU,RHO - FROM PRELIM.F
C
        WRITE(*,*) 'ENTER DATA FILE TO READ'
        READ(*,'(A8)') DATAFILE
C
        OPEN (8,FILE=DATAFILE,STATUS='UNKNOWN')
C
        DO 990 I=1,NGTOT
        READ(8,*) VW(I),DPDZ(I)
 990    CONTINUE
        READ(8,*) ANU,RHO
C
        OPEN (10,FILE='dr.d',STATUS='UNKNOWN')
        OPEN (11,FILE='drfre.d',STATUS='UNKNOWN')
C
C       CONSTANTS
C
        WRITE(10,*) 'NJ=',NJ
        WRITE(10,*) 'NGEVAP=',NGEVAP
        WRITE(11,*) 'NJ=',NJ
        WRITE(11,*) 'NGEVAP=',NGEVAP
```

```
      WRITE(10,*) 'NGADIA=',NGADIA
      WRITE(10,*) 'DR=',DR
      WRITE(10,*) 'DZ=',DZ
      WRITE(10,*) 'DZ/DR2=',ZR
      WRITE(10,*) 'ANU,RHO=',ANU,RHO
      WRITE(10,*)
C
C     ************************************************************************
C
C     INITIALIZE VZ,VR
C
      DO 10 I=1,NGTOT+1
C       NO-SLIP ON WALL
          VZ(I,NJ+1) = 0.0D0
C
C       CENTERLINE CONDITION
          VR(I,1) = 0.0D0
 10   CONTINUE
C
      R(NJ+1) = 1.0D0
      DO 207 J=1,NJ
          R(J) = DR*DFLOAT(J-1)
C
C       END OF PIPE CONDITION
          VZ(1,J) = 0.0D0
          VR(1,J) = 0.0D0
 207  CONTINUE
C
C     ************************************************************************
C
C     START MARCHING USING QUARTIC APPROX

      PREPR = 4.0D0*RR*RR/ANU/RHO
      PREPW = 2.0D0*RR/ANU
C
      WRITE(*,*) 'ENTER NSTART'
      READ(*,*) NSTART
      WRITE(11,*) 'NSTART=',NSTART
      VZBM = 0.0D0
      DO 16 I=2,NSTART-1
          VZBM = DZ*(VW(I-1) + VW(I)) + VZBM
 16   CONTINUE
C
C     PROFILE AT NSTART-1, USED FOR CALC. OF VR(NSTART,J)
C
      I = NSTART-1
          REDPDZ = PREPR*DPDZ(I)/VZBM
          REW = -PREPW*VW(I)
          COEF4D = (5.0D0*REDPDZ/6.0D0 - 1.0D1*REW + 8.0D1)/(1.8D1 - REW)
          COEF4E = (-3.0D0*REDPDZ/4.0D0 + 7.5D0*REW - 4.5D1)/(1.8D1 - REW)
          DO 12 J=1,NJ
              VZ(I,J) = COEF4D*(1.0D0-(R(J))**3) + COEF4E*(1.0D0-(R(J))**4)
 12       CONTINUE
C
```

```
C       GET SET UP AT NSTART
C
        I=NSTART
            VZBMP = VZBM
            VZBM = VZBMP + DZ*(VW(I-1) + VW(I))
            RV = VZBMP/VZBM
            REDPDZ = PREPR*DPDZ(I)/VZBM
            REW = -PREPW*VW(I)
            COEF4D = (5.0D0*REDPDZ/6.0D0 - 1.0D1*REW + 8.0D1)/(1.8D1 - REW)
            COEF4E = (-3.0D0*REDPDZ/4.0D0 + 7.5D0*REW - 4.5D1)/(1.8D1 - REW)
            DO 15 J=1,NJ
                VZ(I,J) = COEF4D*(1.0D0-(R(J))**3) + COEF4E*(1.0D0-(R(J))**4)
 15         CONTINUE
C           DZR(I,NJ+1) = -3.0D0*COEF4D - 4.0D0*COEF4E
C
C       SOLVE FOR VR(I,J)
C
            VR(I,NJ+1) = -VW(I)/VZBM
            VRNJ = VR(I,NJ+1)*(2.0D0+DR)/(2.0D0-DR)
            DO 19 J=2,NJ
                BD(J) = (DR/2.0D0/R(J) - 1.0D0)/2.0D0/DR
                D(J) = 5.0D-1/R(J)
                A(J) = BD(J) + 1.0D0/DR
                C(J) = VZ(I-1,J)*RV/DZ - VZ(I,J)/DZ
 19         CONTINUE
            C(NJ) = C(NJ) - A(NJ)*VR(I,NJ+1)
            CALL THOMAS(2,NJ,BD,D,A,C)
            DO 18 J=2,NJ
                VR(I,J) = C(J)
 18         CONTINUE
C
C       COMPUTE VZBAR,VZ2BAR AT I, GET PHI
C
            VZBD = 0.0D0
            VZ2BD = 0.0D0
            DO 80 J=1,NJ
                VVV = (VZ(I,J) + VZ(I,J+1))*(R(J) + R(J+1))
                VZBD = VZBD + VVV
                VZ2BD = VZ2BD + VVV*(VZ(I,J) + VZ(I,J+1))
 80         CONTINUE
            VZB = VZBD*DR/2.0D0
            VZ2B = VZ2BD*DR/4.0D0
            PERMD = (VZB - 1.0D0)*1.0D2
C
C       At this point I have found mass flow and phi at i=NSTART using the
C       assumed distribution for vz(NSTART,j).
C
C
C       ************************************************************************
C
C
        WRITE(10,*) 'I=',I
        WRITE(10,*) 'PREPR=',PREPR
        WRITE(10,*) 'PREPW=',PREPW
```

```
      WRITE(10,*) 'DPDZ(I)=',DPDZ(I)
      WRITE(10,*) 'VW=',VW(I)
      WRITE(10,*) 'REDPDZ=',REDPDZ
      WRITE(10,*) 'REW=',REW
      WRITE(10,*) 'COEF4D,E=',COEF4D,COEF4E
      WRITE(10,*) 'AVG VZ FROM MASS FLOW IN, FT/SEC  =',VZBM
      WRITE(10,*) 'AVG VZ FROM PROFILE,      NON-DIM =',VZB
      WRITE(10,*) 'PERCENT DIFFERENCE FROM 1.0D0=',PERMD
      WRITE(10,*) 'PHI=', VZ2B
      WRITE(10,*) 'VRNJ FROM SNJ+1 =',VRNJ
      WRITE(10,*)
      WRITE(10,*) '  J       VZ(I,J)          VR(I,J)'
      DO 39 J=NJ+1,1,-1
          WRITE(10,38) J,VZ(I,J),VR(I,J)
 39   CONTINUE
 38   FORMAT(I3,3X,2(F12.6,3X))
      WRITE(10,*)
      WRITE(10,*)
C
C
C
C
C
C     ********************************************************************************
C
C     NOW COMPUTE THE REST OF THE VZs AND VRs
C
      WRITE(11,*) '  Z/L       REW          fRecv,3          PHI'
      NK = NGTOT/10
      IW = NGTOT/200
      IC = 0
      RE = PREPW*VZBM
      DO 40 I=NSTART+1,NGTOT+1
          REW = -PREPW*VW(I)
          IC = IC + 1
          VZBMP = VZBM
          VZBM = VZBMP + DZ*(VW(I) + VW(I-1))
          REP = RE
          REDPDZP = REDPDZ
          REDPDZ = PREPR*DPDZ(I)/VZBM
          RE = PREPW*VZBM
          VZ2BP = VZ2B
          TD = 2.0D0/DR
          RV = VZBMP/VZBM
          VR(I,NJ+1) = -VW(I)/VZBM
          VRL(NJ+1) = VR(I,NJ+1)
          VRL(1) = 0.0D0
          DO 62 J=2,NJ
              VRL(J) = VR(I-1,J)
              VZL(J) = VZ(I-1,J)
 62       CONTINUE
          VZL(1) = VZ(I-1,1)
C
C     SET UP TRI-DIAGONAL MATRIX TO SOLVE ZMOTION FOR VZ(I,J)
```

```
C
            DO 60 J=2,NJ
                TL1 = (VRL(J)+VRL(J-1))*(2.0D0-DR/R(J))*RE*RV/8.0D0/DR
                TL2 = (VRL(J+1)+VRL(J))*(2.0D0+DR/R(J))*RE*RV/8.0D0/DR
                BD(J) = -TL1 - TD/DR + 1.0D0/R(J)/DR
                D(J) = RE*RV*VZL(J)/DZ -TL2 +TL1 +2.0D0*TD/DR
                A(J) = TL2 - TD/DR - 1.0D0/R(J)/DR
                C(J) = -REDPDZ/2.0D0 + RE*VZL(J)*RV*VZ(I-1,J)*VZBMP/VZBM/DZ
 60         CONTINUE
            D(1) = VZL(1)*RE*RV/DZ - VRL(2)*RE*RV/DR + 4.0D0*TD/DR
            A(1) = VRL(2)*RE*RV/DR - 4.0D0*TD/DR
            C(1) = -REDPDZ/2.0D0 + RE*VZL(1)*RV*VZ(I-1,1)*VZBMP/VZBM/DZ
C
C       CALL THOMAS ALGORITHM TO SOLVE TRI-DIAGONAL MATRIX
C
            CALL THOMAS(1,NJ,BD,D,A,C)
            DO 45 J=1,NJ
                VZ(I,J) = C(J)
 45         CONTINUE
C
C       SOLVE FOR VR(I,J) USING CONTINUITY
C
            DO 47 J=2,NJ
                BD(J) = (DR/2.0D0/R(J) - 1.0D0)/2.0D0/DR
                D(J) = 5.0D-1/R(J)
                A(J) = BD(J) + 1.0D0/DR
                C(J) = VZ(I-1,J)*RV/DZ - VZ(I,J)/DZ
 47         CONTINUE
            C(NJ) = C(NJ) - A(NJ)*VR(I,NJ+1)
            CALL THOMAS(2,NJ,BD,D,A,C)
            DO 46 J=2,NJ
                VR(I,J) = C(J)
 46         CONTINUE
            VR2ET1 = DR*(-REDPDZ/2.0D0 + 8.0D0*(VZ(I,2)-VZ(I,1))/DR/DR)/
     +             (VZ(I,2) - 3.0D0*VZ(I,1))/RE
            VR2ES1 = (VZ(I-1,1)*VZBMP/VZBM/DZ - VZ(I,1)/DZ)*DR/2.0D0
            VRNJ = VR(I,NJ+1)*(2.0D0+DR)/(2.0D0-DR)
C
C       COMPUTE VZB, VZ2B TO SEE IF VZ(I,J) SATISFIES CONTINUITY AT I
C
            VZBD = 0.0D0
            VZ2BD = 0.0D0
            VZBMN = VZBM + DZ*(VW(I) + VW(I+1))
            DO 59 J=1,NJ
                VVV = (VZ(I,J) + VZ(I,J+1))*(R(J) + R(J+1))
                VZBD = VZBD + VVV
                VZ2BD = VZ2BD + VVV*(VZ(I,J) + VZ(I,J+1))
 59         CONTINUE
            VZB = VZBD*DR/2.0D0
            VZ2B = VZ2BD*DR/4.0D0
            DPR = REDPDZ*VZBM/PREPR
            ERRMD = VZB - 1.0D0
            PERMD = ERRMD*1.0D2
            DZRW3 = (-4.0D0*VZ(I,NJ) + VZ(I,NJ-1))/2.0D0/DR
```

```
            FRE3 = -RE*2.0D0*ANU*DZRW3/RR/VZBM
            FRECV = VZ(I,NJ)*(1.0D0-DR/2.0D0)*(RE*(VR(I,NJ+1)+VR(I,NJ))/2.0D0
     +
     +              4.0D0/DR) - REDPDZ*DR*(1.0D0 - DR/4.0D0)/2.0D0
            FREPHI = -(4.0D0*(VZ2B*RE - RV*VZ2BP*REP) + DZ*(REDPDZ +
     +               RV*REDPDZP))/(1.0D0 + RV)/2.0D0/DZ
            ZL = DFLOAT(I-1)/NGTOT
C
C       ***************************************************************************
C
C
        IF ((IC .EQ. IW) .OR. (VZ(I,NJ) .LE. 0.0D0)) THEN
            WRITE(11,21) ZL,REW,FRECV,FRE3,FREPHI,VZ2B
 21         FORMAT(F6.3,5(1X,F12.6))
            IC = 0
        ENDIF
        IF ((I .EQ. NK+1) .OR. (I .EQ. 2*NK+1) .OR. (I .EQ. 3*NK+1) .OR.
     +  (I .EQ. 4*NK+1) .OR. (I .EQ. 5*NK+1) .OR. (I .EQ. 6*NK+1) .OR.
     +  (I .EQ. 7*NK+1) .OR. (I .EQ. 8*NK+1) .OR. (I .EQ. 9*NK+1) .OR.
     +  (VZ(I,NJ) .LE. 0.0D0)  ) THEN
              WRITE(*,*) 'I=',I
              WRITE(10,*) 'I=',I
              WRITE(10,83) ZL
 83           FORMAT('Z/L=',F6.3)
              WRITE(10,*) 'f              REW           Re             fRe
     +            PHI'
              WRITE(10,84) F,REW,RE,FRE3,VZ2B
 84           FORMAT(2(F10.6,3X),F12.6,3X,2(F10.6,3X))
              WRITE(10,*) 'FRECV=',FRECV
              WRITE(10,*) 'INITIAL DPDZ(I) FROM DATA=    ',DPDZ(I)
              WRITE(10,*) 'FINAL DPDZ(I) AFTER ITERATION=',DPR
              WRITE(10,*) '# ITERATIONS ON DPDZ=',L
              WRITE(10,*) 'AVG VZ FROM MASS FLOW, STATION I+1 =',VZBM
              WRITE(10,*) 'AVG VZ FROM PROFILE,   STATION I+1 =',VZB
              WRITE(10,*) 'PERCENT DIFFERENCE=',PERMD
              WRITE(10,*) 'VR(I,2) T1,S1=',VR2ET1,VR2ES1
              WRITE(10,*) 'VR(I,NJ) SNJ+1=',VRNJ
              WRITE(10,*)
C
              WRITE(10,*) '  J       VZ(I,J)       VR(I,J)'
C
              DO 58 J=NJ+1,1,-1
                   WRITE(10,54) J,VZ(I,J),VR(I,J)
 58           CONTINUE
C
 54           FORMAT(I3,3X,2(F12.6,3X))
C
              WRITE(10,*)
              WRITE(10,*)
        ENDIF
C
C       ***************************************************************************
C
        IF (VZ(I,NJ) .LE. 0.0D0) THEN
```

```
            GO TO 99
         ENDIF
C
 40      CONTINUE
 99      CONTINUE
         STOP
         END
C
C
C
C
C
C        ************************************************************************
         SUBROUTINE THOMAS(IL,IU,BB,DD,AA,CC)
C
         IMPLICIT DOUBLE PRECISION(A-H, O-Z)
         DIMENSION CC(26),DD(26),AA(26),BB(26)
C
C        ESTABLISH UPPER TRIANGULAR MATRIX
C
         LP=IL+1
         DO 10 I=LP,IU
                 RR=BB(I)/DD(I-1)
                 DD(I)=DD(I)-RR*AA(I-1)
 10              CC(I) = CC(I)-RR*CC(I-1)
C
C        BACK SUBSTITUTION
C
         CC(IU) = CC(IU)/DD(IU)
         DO 20 I=LP,IU
                 J=IU-I+IL
 20      CC(J) = (CC(J) - AA(J)*CC(J+1))/DD(J)
C
C        SOLUTION STORED IN CC
C
         RETURN
         END
```

Appendix B

```
      PROGRAM SIM
C
C     THIS PROGRAM COMPUTES fRe ASSUMING INCOMPRESSIBLE FLOW
C
C     DEFINITIONS OF VARIABLES
C
C     ANU             MOMENTUM DIFFUSIVITY
C     COEF4D          COEFFICIENT OF QUARTIC APPROXIMATION
C     COEF4E                  .       .       .       .
C     DPDZ            AXIAL PRESSURE GRADIENT
C     DR              DIFFERENTIAL DISTANCE IN RADIAL DIR'N.
C     DZ                      .       .       . AXIAL .
C     DZRW3           VELOCITY GRADIENT AT WALL FROM 3-PT DERIV
C     fRe             f*Re
C     FRE3            fRe from 3-POINT DERIVATIVE
C     FRECV           fRe from CONTROL VOLUME
C     FREPHI          fRe from PHI
C     NE              NUMBER OF EQUATIONS
C     NGADIA          # GRIDS IN ADIABATIC REGION
C     NGCOND          # GRIDS IN CONDENSER, = NGEVAP
C     NGEA            NGEVAP + NGADIA
C     NGEVAP          # GRIDS IN EVAPORATOR, = NGCOND
C     NGTOT           TOTAL # GRIDS ALONG AXIS OF PIPE
C     NJ              # RADIAL GRIDS
C     PHI             MOMENTUM FLUX FACTOR
C     R(J)            RADIUS
C     RE              AXIAL REYNOLDS NUMBER
C     REW             RADIAL REYNOLDS NUMBER
C     RHO             DENSITY
C     RR              RADIUS OF PIPE
C     RV              VZBM(I-1)/VZBM(I)
C     VR              RADIAL VELOCITY
C     VW              WALL RADIAL VELOCITY
C     VZ              AXIAL VELOCITY
C     VZ2B            AVERAGE OF SQUARE OF VELOCITY (FROM PROFILE)
C     VZB             AVERAGE VELOCITY (FROM PROFILE)
C     VZBM            AVERAGE VELOCITY (FROM VW)
C     VZBMN           AVERAGE VELOCITY AT NEXT I
C     VZBMP           VZBM AT PREVIOUS I
C     ZL              NON-DIMENSIONAL AXIAL DISTANCE Z/L
C
C
C
      IMPLICIT DOUBLE PRECISION(A-H, O-Z)
      CHARACTER*8 DATAFILE
C
      DIMENSION VR(6365,51),VZ(6365,51),DPDZ(6365),R(51),
     +  VW(6365),BT(51),DT(51),CT(51),AT(51),DS(51),AS(51),
     +  CS(51),A(100,100),C(100),INDX(100)
C
C     ****************************************
```

```
C
C       GRID SIZE
C
        WRITE(*,*) 'ENTER # GRIDS IN R-DIRECTION'
        READ(*,*) NJ
 78     WRITE(*,*) 'ENTER # GRIDS IN EVAP (783,1566,3132)'
        READ(*,*) NGEVAP
C
        IF (NGEVAP .EQ. 783) THEN
            NGADIA = 25
        ELSEIF (NGEVAP .EQ. 1566) THEN
            NGADIA = 50
        ELSEIF (NGEVAP .EQ. 3132) THEN
            NGADIA = 100
        ELSE
            GO TO 78
        ENDIF
C
 65     WRITE(*,*) 'ENTER RUN NUMBER'
        READ(*,*) IRN
        IF (IRN .EQ. 1) THEN
            P = 14.263613999968
        ELSEIF (IRN .EQ. 2) THEN
            P = 14.238297320471
        ELSEIF (IRN .EQ. 3) THEN
            P = 14.266756940340
        ELSEIF (IRN .EQ. 4) THEN
            P = 14.203597320072
        ELSE
            GO TO 65
        ENDIF
        P = P*1.44D2
        P1 = P
C
        NGCOND = NGEVAP
        NGTOT = NGEVAP + NGADIA + NGCOND
C
        RR = 2.5D-1/12.0D0
        DR = 1.0D0/NJ
        DZ = 15.66D0/2.5D-1/NGEVAP
        ZR = DZ/DR/DR
C
C
C
C       ****************************************
C
C       READ PRESSURE DATA  - VW,DPDZ,ANU,RHO - FROM PRELIM.F
C
        WRITE(*,*) 'ENTER DATA FILE TO READ'
        READ(*,'(A8)') DATAFILE
C
        OPEN (8,FILE=DATAFILE,STATUS='UNKNOWN')
C
        DO 990 I=1,NGTOT
```

```
      READ(8,*) VW(I),DPDZ(I)
 990  CONTINUE
      READ(8,*) ANU,RHO
C
      OPEN (10,FILE='sim.d',STATUS='UNKNOWN')
      OPEN (11,FILE='simfre.d',STATUS='UNKNOWN')
      OPEN (12,FILE='dpdz.d',STATUS='UNKNOWN')
C
C     CONSTANTS
C
      WRITE(10,*) 'NJ=',NJ
      WRITE(10,*) 'NGEVAP=',NGEVAP
      WRITE(11,*) 'NJ=',NJ
      WRITE(11,*) 'NGEVAP=',NGEVAP
      WRITE(10,*) 'NGADIA=',NGADIA
      WRITE(10,*) 'DR=',DR
      WRITE(10,*) 'DZ=',DZ
      WRITE(10,*) 'DZ/DR2=',ZR
      WRITE(10,*) 'ANU,RHO=',ANU,RHO
      WRITE(10,*)
C
C     ************************************************************************
C
C     INITIALIZE VZ,VR
C
      DO 10 I=1,NGTOT+1
C       NO-SLIP ON WALL
          VZ(I,NJ+1) = 0.0D0
C
C       CENTERLINE CONDITION
          VR(I,1) = 0.0D0
 10   CONTINUE
C
      R(NJ+1) = 1.0D0
      DO 207 J=1,NJ
          R(J) = DR*DFLOAT(J-1)
C
C       END OF PIPE CONDITION
          VZ(1,J) = 0.0D0
          VR(1,J) = 0.0D0
 207  CONTINUE
C
C     ************************************************************************
C
C     START MARCHING USING QUARTIC APPROX

      PREPR = 4.0D0*RR*RR/ANU/RHO
      PREPW = 2.0D0*RR/ANU
C
      NSTART = NGTOT/20
      VZBM = 0.0D0
      DO 16 I=2,NSTART-1
          P = P + (DPDZ(I-1) + DPDZ(I))*DZ*RR/2.0D0
          VZBM = DZ*(VW(I-1) + VW(I)) + VZBM
```

```
 16     CONTINUE
C
C       PROFILE AT NSTART-1, USED FOR CALC. OF VR(NSTART,J)
C
        I = NSTART-1
        WRITE(*,*) 'I=',I
            REDPDZ = PREPR*DPDZ(I)/VZBM
            REW = -PREPW*VW(I)
            COEF4D = (5.0D0*REDPDZ/6.0D0 - 1.0D1*REW + 8.0D1)/(1.8D1 - REW)
            COEF4E = (-3.0D0*REDPDZ/4.0D0 + 7.5D0*REW - 4.5D1)/(1.8D1 - REW)
C
C       VZ DISTRIBUTION AT I=NSTART-1.  ASSUME QUARTIC DISTRIBUTION.
C
            DO 12 J=1,NJ
                VZ(I,J) = COEF4D*(1.0D0-(R(J))**3) + COEF4E*(1.0D0-(R(J))**4)
 12         CONTINUE
C
C       GET SET UP AT NSTART
C
        I=NSTART
        WRITE(*,*) 'I=',I
            P = P + (DPDZ(I-1) + DPDZ(I))*DZ*RR/2.0D0
            VZBMP = VZBM
            VZBM = VZBMP + DZ*(VW(I-1) + VW(I))
            RV = VZBMP/VZBM
            REDPDZ = PREPR*DPDZ(I)/VZBM
            REW = -PREPW*VW(I)
            COEF4D = (5.0D0*REDPDZ/6.0D0 - 1.0D1*REW + 8.0D1)/(1.8D1 - REW)
            COEF4E = (-3.0D0*REDPDZ/4.0D0 + 7.5D0*REW - 4.5D1)/(1.8D1 - REW)
C
C       VZ DISTRIBUTION AT I=NSTART.  ASSUME QUARTIC DISTRIBUTION.
C
            DO 15 J=1,NJ
                VZ(I,J) = COEF4D*(1.0D0-(R(J))**3) + COEF4E*(1.0D0-(R(J))**4)
 15         CONTINUE
C           DZR(I,NJ+1) = -3.0D0*COEF4D - 4.0D0*COEF4E
C
C       SOLVE FOR VR(I,J)
C
            VR(I,NJ+1) = -VW(I)/VZBM
            DO 19 J=NJ,2,-1
                VR(I,J) =  VR(I,J+1)*R(J+1)/R(J) + (VZ(I,J) + VZ(I,J+1) -
     +          RV*(VZ(I-1,J)+VZ(I-1,J+1)))*(R(J+1)+R(J))*DR/R(J)/DZ/4.0D0
 19         CONTINUE
C
C       COMPUTE VZBAR,VZ2BAR AT I, GET PHI
C
            VZBD = 0.0D0
            VZ2BD = 0.0D0
            DO 80 J=1,NJ
                VVV = (VZ(I,J) + VZ(I,J+1))*(R(J) + R(J+1))
                VZBD = VZBD + VVV
                VZ2BD = VZ2BD + VVV*(VZ(I,J) + VZ(I,J+1))
 80         CONTINUE
```

```
            VZB = VZBD*DR/2.0D0
            VZ2B = VZ2BD*DR/4.0D0
            PERMD = (VZB - 1.0D0)*1.0D2
C
C       At this point I have found mass flow and phi at i=NSTART using the
C       assumed distribution for vz(NSTART,j).
C
C
C       ************************************************************************
C
C
        WRITE(10,*) 'I=',I
        WRITE(10,*) 'PREPR=',PREPR
        WRITE(10,*) 'PREPW=',PREPW
        WRITE(10,*) 'DPDZ(I)=',DPDZ(I)
        WRITE(10,*) 'VW=',VW(I)
        WRITE(10,*) 'REDPDZ=',REDPDZ
        WRITE(10,*) 'REW=',REW
        WRITE(10,*) 'COEF4D,E=',COEF4D,COEF4E
        WRITE(10,*) 'AVG VZ FROM MASS FLOW IN, FT/SEC  =',VZBM
        WRITE(10,*) 'AVG VZ FROM PROFILE,      NON-DIM =',VZB
        WRITE(10,*) 'PERCENT DIFFERENCE FROM 1.0D0=',PERMD
        WRITE(10,*) 'PHI=', VZ2B
        WRITE(10,*)
        WRITE(10,*) '  J       VZ(I,J)          VR(I,J)'
        DO 39 J=NJ+1,1,-1
            WRITE(10,38) J,VZ(I,J),VR(I,J)
 39     CONTINUE
 38     FORMAT(I3,3X,2(F12.6,3X))
        WRITE(10,*)
        WRITE(10,*)
C
C
C
C
C
C       ************************************************************************
C
C       NOW COMPUTE THE REST OF THE VZs AND VRs
C
        WRITE(11,*) '  Z/L       REW          fRecv,3,phi          PHI'
        NK = NGTOT/10
        IW = NGTOT/200
        IC = 0
        TD = 2.0D0/DR
        RE = PREPW*VZBM
        DO 40 I=NSTART+1,NGTOT+1
            P = P + (DPDZ(I-1) + DPDZ(I))*DZ*RR/2.0D0
            IC = IC+1
            VZBMP = VZBM
            VZBM = VZBMP + DZ*(VW(I) + VW(I-1))
            REP = RE
            RE = PREPW*VZBM
            REDPDZP = REDPDZ
```

```
            VZ2BP = VZ2B
            RV = VZBMP/VZBM
            VR(I,NJ+1) = -VW(I)/VZBM
C
C       FORM A MATRIX
C
            NE = 2*NJ
            DO 57 M=1,NE
                DO 56 N=1,NE
                    A(M,N) = 0.0D0
 56             CONTINUE
 57         CONTINUE
C
C           CALCULATE COEFFICIENTS
C
            DO 55 K=1,NE-1,2
                A(K,NE) = 5.0D-1
 55         CONTINUE
C
            DO 24 J=2,NJ
                RRV = 4.0D0*DZ/(R(J+1) + R(J))/DR
                DS(J) = -RRV*R(J)
                AS(J) = RRV*R(J+1)
                CS(J) = (VZ(I-1,J) + VZ(I-1,J+1))*RV
                TL1 = (VR(I-1,J)+VR(I-1,J-1))*(2.0D0-DR/R(J))*RE*RV/8.0D0/DR
                TL2 = (VR(I-1,J+1)+VR(I-1,J))*(2.0D0+DR/R(J))*RE*RV/8.0D0/DR
                BT(J) = -TL1 - TD/DR + 1.0D0/R(J)/DR
                DT(J) = RE*RV*VZ(I-1,J)/DZ -TL2 +TL1 +2.0D0*TD/DR
                AT(J) = TL2 - TD/DR - 1.0D0/R(J)/DR
                CT(J) = RE*VZ(I-1,J)*RV*VZ(I-1,J)*RV/DZ
 24         CONTINUE
C
            A(1,1) = RE*RV*(VZ(I-1,1)/DZ - VR(I-1,2)/DR) + 4.0D0*TD/DR
            A(1,3) = RE*RV*VR(I-1,2)/DR - 4.0D0*TD/DR
            C(1) = RE*VZ(I-1,1)*RV*VZ(I-1,1)*RV/DZ
            ...,1) = 1.0D0
            A(2,2) = 4.0D0*DZ/DR
            A(2,3) = 1.0D0
            C(2) = (VZ(I-1,1) + VZ(I-1,2))*RV
            DO 53 J=2,NJ-1
                K = 2*J-1
                A(K,K-2) = BT(J)
                A(K,K) = DT(J)
                A(K,K+2) = AT(J)
                C(K) = CT(J)
 53         CONTINUE
            DO 52 J=2,NJ-1
                K = 2*J
                A(K,K-2) = DS(J)
                A(K,K) = AS(J)
                A(K,K-1) = 1.0D0
                A(K,K+1) = 1.0D0
                C(K) = CS(J)
 52         CONTINUE
```

```
C
C           NOW FOR LAST TWO ROWS
C
            A(NE,NE-2) = DS(NJ)
            A(NE,NE-1) = 1.0D0
            C(NE) = CS(NJ) - AS(NJ)*VR(I,NJ+1)
            A(NE-1,NE-3) = BT(NJ)
            A(NE-1,NE-1) = DT(NJ)
            C(NE-1) = CT(NJ)
C
            CALL LUDCMP (A,NE,50,INDX,YD)
            CALL LUBKSB (A,NE,50,INDX,C)
C
            DO 49 J=2,NJ
                K=2*J-2
                VR(I,J) = C(K)
                VZ(I,J) = C(K+1)
 49         CONTINUE
            VZ(I,1) = C(1)
            REDPDZ = C(NE)
C
C       COMPUTE VZB, VZ2B TO SEE IF VZ(I+1,J) SATISFIES CONTINUITY AT I+1
C
            VZBD = 0.0D0
            VZ2BD = 0.0D0
            VZBMN = VZBM + DZ*(VW(I) + VW(I+1))
            DO 59 J=1,NJ
                VVV = (VZ(I,J) + VZ(I,J+1))*(R(J) + R(J+1))
                VZBD = VZBD + VVV
                VZ2BD = VZ2BD + VVV*(VZ(I,J) + VZ(I,J+1))
 59         CONTINUE
            VZB = VZBD*DR/2.0D0
            VZ2B = VZ2BD*DR/4.0D0
            DPR = REDPDZ*VZBM/PREPR
            ERRMD = VZB - 1.0D0
            PERMD = ERRMD*1.0D2
            DZRW3 = (-4.0D0*VZ(I,NJ) + VZ(I,NJ-1))/DR/2.0D0
            FRE3 = -4.0D0*DZRW3
            FRECV = VZ(I,NJ)*(1.0D0-DR/2.0D0)*(RE*(VR(I,NJ+1)+VR(I,NJ))/2.0D0
     +              + 4.0D0/DR) - REDPDZ*DR*(1.0D0 - DR/4.0D0)/2.0D0
            FREPHI = -(4.0D0*(VZ2B*RE - RV*VZ2BP*REP) + DZ*(REDPDZ +
     +               RV*REDPDZP))/(1.0D0 + RV)/2.0D0/DZ
            ZL = DFLOAT(I-1)/NGTOT
            WRITE(12,*) DPR
C
C       ************************************************************************
C
        IF ((IC .EQ. IW) .OR. (VZ(I,NJ) .LE. 0.0D0)) THEN
            WRITE(11,21) ZL,REW,FRECV,FRE3,FREPHI,VZ2B,P/P1
 21         FORMAT(F5.3,1X,5(F10.6),1X,F11.8)
            IC = 0
        ENDIF
C
        IF ((I .EQ. NK+1) .OR. (I .EQ. 2*NK+1) .OR. (I .EQ. 3*NK+1) .OR.
```

```
     +  (I .EQ. 4*NK+1) .OR. (I .EQ. 5*NK+1) .OR. (I .EQ. 6*NK+1) .OR.
     +  (I .EQ. 7*NK+1) .OR. (I .EQ. 8*NK+1) .OR. (I .EQ. 9*NK+1) .OR.
     +  (I .EQ. NGTOT*0.95) .OR. (VZ(I,NJ) .LE. 0.0D0) ) THEN
              WRITE(*,*) 'I=',I
              WRITE(10,*) 'I=',I
              WRITE(10,83) ZL
 83           FORMAT('Z/L=',F6.3)
              WRITE(10,*) 'f              REW           Re             fRe
     +            PHI'
              WRITE(10,84) F,REW,RE,FRECV,VZ2B
 84           FORMAT(2(F10.6,3X),F12.6,3X,2(F10.6,3X))
              WRITE(10,*) 'INITIAL DPDZ(I) FROM DATA=     ',DPDZ(I)
              WRITE(10,*) 'FINAL DPDZ(I) AFTER ITERATION=',DPR
              WRITE(10,*) 'AVG VZ FROM MASS FLOW, STATION I+1 =',VZBM
              WRITE(10,*) 'AVG VZ FROM PROFILE,   STATION I+1 =',VZB
              WRITE(10,*) 'PERCENT DIFFERENCE=',PERMD
              WRITE(10,*)
C
              WRITE(10,*) '  J        VZ(I,J)        VR(I,J)'
C
              DO 58 J=NJ+1,1,-1
                   WRITE(10,54) J,VZ(I,J),VR(I,J)
 58           CONTINUE
C
 54           FORMAT(I3,3X,2(F12.6,3X))
C
              WRITE(10,*)
              WRITE(10,*)
         ENDIF
C
C        ********************************************************************

         IF (VZ(I,NJ) .LE. 0.0D0) THEN
             GO TO 99
         ENDIF
C
C        PREPARE FOR NEXT I
C
             REW = -PREPW*VW(I+1)
             IF (I .EQ. NGEVAP+NGADIA+1) THEN
                 REDPDZ = PREPR*DPDZ(I+1)/VZBM
             ENDIF
C
 40      CONTINUE
 99      CONTINUE
         STOP
         END
C
C
C
C
C        ********************************************************
C
C
```

```
      SUBROUTINE LUDCMP(A,N,NP,INDX,D)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      PARAMETER (NMAX=100,TINY=1.0E-20)
      DIMENSION A(100,100),INDX(100),VV(100)
      D=1.
      DO 12 I=1,N
        AAMAX=0.
        DO 11 J=1,N
          IF (ABS(A(I,J)).GT.AAMAX) AAMAX=ABS(A(I,J))
11      CONTINUE
        IF (AAMAX.EQ.0.) PAUSE 'Singular matrix.'
        VV(I)=1./AAMAX
12    CONTINUE
      DO 19 J=1,N
        IF (J.GT.1) THEN
          DO 14 I=1,J-1
            SUM=A(I,J)
            IF (I.GT.1)THEN
              DO 13 K=1,I-1
                SUM=SUM-A(I,K)*A(K,J)
13            CONTINUE
              A(I,J)=SUM
            ENDIF
14        CONTINUE
        ENDIF
        AAMAX=0.
        DO 16 I=J,N
          SUM=A(I,J)
          IF (J.GT.1)THEN
            DO 15 K=1,J-1
              SUM=SUM-A(I,K)*A(K,J)
15          CONTINUE
            A(I,J)=SUM
          ENDIF
          DUM=VV(I)*ABS(SUM)
          IF (DUM.GE.AAMAX) THEN
            IMAX=I
            AAMAX=DUM
          ENDIF
16      CONTINUE
        IF (J.NE.IMAX)THEN
          DO 17 K=1,N
            DUM=A(IMAX,K)
            A(IMAX,K)=A(J,K)
            A(J,K)=DUM
17        CONTINUE
          D=-D
          VV(IMAX)=VV(J)
        ENDIF
        INDX(J)=IMAX
        IF(J.NE.N)THEN
          IF(A(J,J).EQ.0.)A(J,J)=TINY
          DUM=1./A(J,J)
          DO 18 I=J+1,N
```

```
            A(I,J)=A(I,J)*DUM
18          CONTINUE
          ENDIF
19      CONTINUE
        IF(A(N,N).EQ.0.)A(N,N)=TINY
        RETURN
        END
C
C
C       ****************************************
C
C
        SUBROUTINE LUBKSB(A,N,NP,INDX,B)
        IMPLICIT DOUBLE PRECISION(A-H,O-Z)
        DIMENSION A(100,100),INDX(100),B(100)
        II=0
        DO 12 I=1,N
          LL=INDX(I)
          SUM=B(LL)
          B(LL)=B(I)
          IF (II.NE.0)THEN
            DO 11 J=II,I-1
              SUM=SUM-A(I,J)*B(J)
11          CONTINUE
          ELSE IF (SUM.NE.0.) THEN
            II=I
          ENDIF
          B(I)=SUM
12      CONTINUE
        DO 14 I=N,1,-1
          SUM=B(I)
          IF(I.LT.N)THEN
            DO 13 J=I+1,N
              SUM=SUM-A(I,J)*B(J)
13          CONTINUE
          ENDIF
          B(I)=SUM/A(I,I)
14      CONTINUE
        RETURN
        END
```

Bibliography

1. Anderson, D.A., Tannehill, J.C., and Pletcher, R.H. Computational Fluid Mechanics and Heat Transfer. Hemisphere Publishing Copororation, 1984.

2. Bowman, Capt W.J. Heat Pipe Vapor Dynamics. PhD Dissertation, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, May 1987 (88182592).

3. Chi, S.W. Heat Pipe Theory and Practice. Hemisphere Publishing Company, 1976.

4. Kinney, R.B. "Fully Developed Frictional and Heat Transfer Characteristics of Laminar Flow in Porous Tubes," International Journal of Heat and Mass Transfer, Vol. 11, pp 1393-1401, 1968.

5. Manley, Capt D.A. Incompressible Flow Friction Factors in a Simulated Heat Pipe. MS Thesis AFIT/GAE/AA/88D-22. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1988.

6. Press, W.H. et al. Numerical Recipes. Cambridge University Press, 1986

Vita

Capt James E. Mayhew w[redacted] He graduated from high school in Averill Park, New York, in 1980 and attended Syracuse University. While enrolled at Syracuse, he participated in a Year-Abroad program and studied Aeronautics at The City University - London, England from October 1982 to July 1983. Upon graduation from Syracuse in May 1985, he received the degree of Bachelor of Science in Aerospace Engineering, and a commission in the USAF through the ROTC program. He was assigned to Wright-Patterson AFB, OH in July 1985 as a Tactical Air Command Liaison Officer to the Aeronautical Systems Division, and remained there until entering the School of Engineering, Air Force Institute of Technology, in May 1988. He and his wife, [redacted] have two children, [redacted]

[redacted]A

E[redacted]

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GAE/ENY/89D-23	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Engineering 6b. OFFICE SYMBOL (If applicable) AFIT/ENY	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) Air Force Institute of Technology Wright-Patterson AFB OH 45433-6583	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
INCOMPRESSIBLE FLOW FRICTION FACTORS IN A SIMULATED HEAT PIPE

12. PERSONAL AUTHOR(S)
James E. Mayhew, B.S., Capt, USAF

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
MS Thesis	FROM _____ TO _____	1989 December	67

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	
20	04		Heat Pipe, Porous Pipe, friction factor, friction coefficient, Blowing, Suction, Injection, Extraction

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Thesis Advisor: Dr. James E. Hitchcock
Professor of Mechanical Engineering

ABSTRACT ON BACK

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Dr. James E. Hitchcock, Professor	22b. TELEPHONE (Include Area Code) (513)-255-3517 22c. OFFICE SYMBOL AFIT/ENY

DD Form 1473, JUN 86 *Previous editions are obsolete.* SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

19. Porous pipes with blowing and suction are used to simulate the vapor flow in heat pipes. A computer program for steady, two-dimensional, boundary-layer flow has been written to find friction factors using the axial pressure distributions measured in an experiment with incompressible flow in a porous pipe. The results for some existing experimental data are presented and compared to previously published solutions for fully-developed flow. Also, this program has been modified to numerically simulate porous pipe flow without using axial pressure distribution as an input. This "simulation" program furnishes additional results for comparison to the original "data reduction" program. It is observed that the simulation program accurately computes friction factors and flow separation points in a porous pipe with low radial Reynolds number.