LA-12984

Introduction to Finite-Difference Methods for Numerical Fluid Dynamics

Los Alamos
NATIONAL LABORATORY

Edited by Patricia W. Mendius, Group CIC-1

LA-12984

UC-700
Issued: September 1995

Introduction to Finite-Difference Methods for Numerical Fluid Dynamics

Evan Scannapieco
Francis H. Harlow

Los Alamos
NATIONAL LABORATORY
Los Alamos, New Mexico 87545

TABLE OF CONTENTS

INTRODUCTION TO FINITE-DIFFERENCE METHODS FOR NUMERICAL FLUID DYNAMICS

by

Evan Scannapieco and Francis H. Harlow

ABSTRACT

This work is intended to be a beginner's exercise book for the study of basic finite-difference techniques in computational fluid dynamics. It is written for a student level ranging from high-school senior to university senior. Equations are derived from basic principles using algebra. Some discussion of partial-differential equations is included, but knowledge of calculus is not essential. The student is expected, however, to have some familiarity with the FORTRAN computer language, as the syntax of the computer codes themselves is not discussed. Topics examined in this work include: one-dimensional heat flow, one-dimensional compressible fluid flow, two-dimensional compressible fluid flow, and two-dimensional incompressible fluid flow with additions of the equations of heat flow and the $K-\epsilon$ model for turbulence transport. Emphasis is placed on numerical instabilities and methods by which they can be avoided, techniques that can be used to evaluate the accuracy of finite-difference approximations, and the writing of the finite-difference codes themselves. Concepts introduced in this work include: flux and conservation, implicit and explicit methods, Lagrangian and Eulerian methods, shocks and rarefactions, donor-cell and cell-centered advective fluxes, compressible and incompressible fluids, the Boussinesq approximation for heat flow, cartesian tensor notation, the Boussinesq approximation for the Reynolds stress tensor, and the modeling of transport equations. A glossary is provided which defines these and other terms.

I. INTRODUCTION

One of the most important techniques used in the computer modeling of physical systems, finite differencing represents an essential part of modern theoretical physics. Able to generate solutions to systems that are far too intricate to be solved analytically, this technique has given physicists the ability to model, examine, and better understand complex physical situations. From the study of microscopic systems to the modeling of the world's climate, finite-difference programs have opened up an entire field of research that has only been possible within the the past 40 years.

In the following paper we will examine a series of finite-difference programs, gaining a clearer understanding of their underlying physical principles and the techniques by which these are implemented. It is our intention to represent these physical systems so that they will be easily understood both by those who are dealing with them for the first time and those familiar with their partial-differential representations. Partial differential equations will be used but only as a result of a discussion of the basic principles from which they are derived. The mathematics will follow, as it should, from a clear set of physically meaningful observations.

It will be essential, however, for the reader to have a clear understanding of the FORTRAN computer language in which all programs will be written. While there are many finite-difference simulations that are written in other languages, FORTRAN has proved to be an efficient, easily understood, and widely accepted language for scientific computing. For these reasons we will limit our discussion to simulations written in this language, and for reasons of scope we will not discuss the meanings of each of its commands. We assume that the reader is already familiar with computers and desires to apply this knowledge to the study of physical systems. It is our intention to apply physical principles to the creation of computer simulations, not to discuss the syntax of the simulations themselves.

There are two ways to approach this work. It can be interpreted as a guide to writing one's own finite difference simulations, a type of handbook for creating one's own codes, or it can be read as a textbook, omitting actual programming by the reader. We hope that the reader will adopt the former approach. While it will take more time for the reader to create his own simulations, the extra time will prove to be time well spent.

The reasons for writing one's own code are twofold. First, the reader who is able to structure his own code will be sure to have a full understanding of the concepts involved. There is only so much that can be explained about the process; true understanding will result only from experience. Secondly, by writing his own programs, the reader gains the advantage of being able to examine the results obtained from various initial parameters. Only a limited number of results will be presented in this work, thereby leaving the reader with a vast set of cases which can be independently investigated.

As we move through this series of programs, we will examine a broad spectrum of physical systems. We will begin with the simulation of heat transfer in one dimension, examining various forms of numerical instabilities and explicit and implicit solution techniques. Our discussion will then move to compressible fluid flow in one dimension, examining both Eulerian and Lagrangian methods of simulation of a number of different systems. We will show a method for determining the accuracy of our finite-difference representations and use it to examine numerical instabilities. We will discuss the simulation of incompressible fluid flow in two dimensions, calculate incompressible fluid flow in conjunction with heat, discuss two-dimensional compressible fluid flow, and finally implement the equations of turbulence transport in an incompressible code.

Likewise, our discussion will cover an equally broad set of topics in a range of technical fields. We will discuss various physical equations and the systems that they represent, the mathematical properties of these equations and how these relate to solving them with a computer, and the structuring of the programs themselves. Although we will be dealing with these varying subjects, a single underlying thrust must remain clear in our minds. We must remember that what we are doing is taking observable physical phenomena and translating them into terms which can be dealt with by the computer. The form will be greatly changed, but the basic physical principles will always be faithfully represented.

In as much as we can simulate reality, we can use the computer to make predictions about what will occur in a certain set of circumstances. Finite-difference techniques can create an artificial laboratory for examining situations which would be impossible to observe otherwise, but we must always remain critical of our results. Finite-differencing can be an extremely powerful tool, but only when it is firmly set in a basis of physical meaning. In order for a finite-difference code to be successful, we must start from the beginning, dealing with simple cases and examining our logic each step of the way. Building further insights on what we have done in the past, we will start with the simplest case possible: heat transfer in a single dimension. The rest will follow logically.

II. ONE-DIMENSIONAL HEAT FLOW

A. Flux and Conservation

The first system that will be examined in this series of studies is that of heat conduction in a single dimension. In this chapter, we will write a program that numerically solves a single equation of heat transfer over a one-dimensional array. This program can most easily be pictured as the simulation of a metal rod that is initially at an even temperature and is insulated along its sides. As the program progresses, the simulated rod is heated from one end, and the resulting change in temperature along the rod is recorded as output. A diagram of this system appears in Fig. II-1.

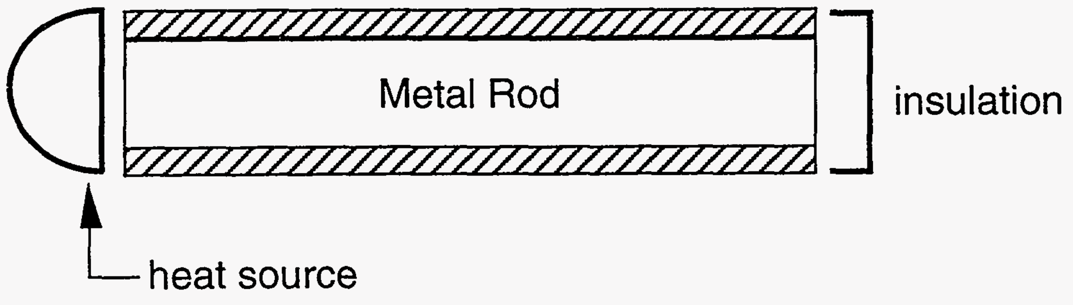


Fig. II-1

Each section of this rod is represented by an element in an array that corresponds to its position. These elements, called ZONES, record the temperature at finite distances along the rod and at finite time intervals, hence the name *finite difference*. This type of representation can be thought of as similar to a motion picture, where each frame exists for a small but finite time step. Motion is not fluid as in reality but is instead approximated by a series of small changes from one "frame" to the next.

Our simulation of heat flow in this manner will introduce two basic concepts that are essential to the understanding of the underlying principles on which many finite-difference codes are based: FLUX and CONSERVATION. Flux is the amount of something passing through a unit area in a unit time. In our current example, the flux that is of interest is HEAT FLUX, the transport of heat from one zone to another. But flux is by no means limited to only heat. It can represent the movement of mass, momentum, energy, or any other value that describes the amount of something that is present in a zone. No matter what is being fluxed, the concept remains essentially the same. Flux represents motion from one place to another, the rate at which something moves through a given area.

Conservation means that the total amount of something never changes regardless of its motion from one region to another. If this same concept is viewed in terms of the amount of something that exists in a finite region, conservation means that in any region of space the change in something equals the amount that goes in minus the amount that comes out plus the change of that amount within the region. Once again, this principle holds true for many different quantities. Mass, momentum, and energy, while different physically, are identical in the fact that they are conserved.

These two concepts of flux and conservation are critical to the way that our finite difference codes are structured. Their implementation and a simple equation obtained from experimental observation are all that is necessary to represent the transfer of heat numerically.

B. Numerical Representation

In order to represent this system in a manner that can be solved computationally, we must first examine the structure represented by each zone in our simulated metal rod. Looking at an individual zone (also called a CELL), we find a physical system as in Fig. II-2.

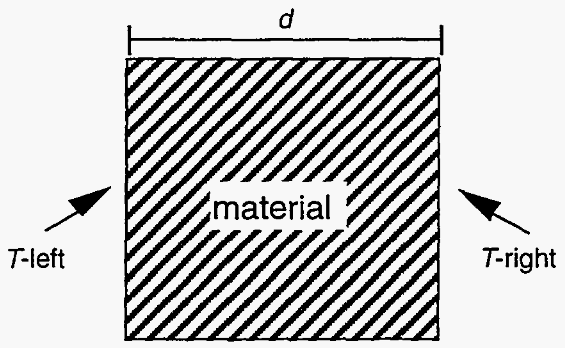


Fig. II-2

Here T-left and T-right represent the temperature along either edge of the zone, and d represents the thickness. The heat flux of this system is defined by an equation known as Fick's Law. A result of direct experimental observations, Fick's Law is as follows:

$$\text{flux of heat} = \frac{k\,(T_{\text{left}} - T_{\text{right}})}{d}\,. \tag{II-1}$$

In this equation, k is called THE COEFFICIENT OF HEAT CONDUCTIVITY and is proportional to the rate at which a given material conducts heat across a temperature gradient. k is an intrinsic property of the material being represented and must be chosen based on the conductive properties of that material. For example, silver, a very conductive metal, is represented by a high value of k, around 4 J/s·cm· °C. Wood on the other hand, is a poor heat conductor and is consequently associated with a low k value, 1.3×10^{-3} J/s·cm· °C. The conductivity of iron is somewhere between these two materials, yielding an intermediate value for k, 0.67 J/s·cm· °C.

Taking this equation as it applies to a single cell, we can now make a generalization as to how it can be implemented over an array. Given a rod of length D, this length can be divided into an array of size $\bar{\jmath}$. Each zone will now have a length dx, which is defined as $D/\bar{\jmath}$. Such a system is shown in Fig II-3.

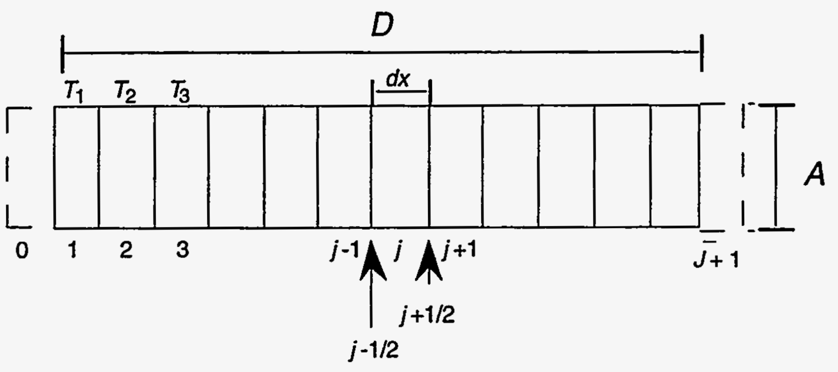


Fig. II-3

Each zone in this array can now be indexed with a counter j, with zones $j-1$ and $j+1$ being the zones at the left and right respectively. Note that the flux between zones will occur at the walls and will therefore occur at points such as $j+1/2$ and $j-1/2$ in the diagram. Note also that the diagram contains both a zone 0 and a zone $\bar{j}+1$, existing beyond the normal bounds of the rod. These zones are used to implement BOUNDARY CONDITIONS, equations that represent the external conditions that affect the values of the real zones. The heating source at the left of the pipe is represented by one such boundary condition. Temperatures for each cell are defined at the center of the cell, existing at positions 1, 2, 3, etc. Density of cells and cross-sectional area between cells are defined as ρ and A respectively.

These definitions can be used to write an expression for the heat energy contained in any given cell:

$$\text{Volume} = A\ dx \tag{II-2}$$

$$\text{Heat energy of cell } j = \text{Mass } b\ T_j\ , \tag{II-3}$$

where $b =$ the specific heat of the material. As

$$\text{Mass} = \text{Volume density} = A\ dx\ \rho\ , \tag{II-4}$$

the heat energy can also be written as

$$\text{Heat energy of cell } j = A\,dx\,\rho\,b\,T_j\,. \tag{II-5}$$

We now apply our conservation of energy principle to derive an equation for the change in energy. The letter n will be used as a TIME CYCLE COUNTER, an integer that represents the number of time cycles that have been calculated. These cycles, also called time steps, can be thought of as individual frames in our analogy of the motion picture. The time at a cycle n is represented by t^n, which is computed as follows:

$$t^n = n\,dt\,. \tag{II-6}$$

In this equation dt is equal to the time increment per cycle, i.e., the change in time between each "frame." The superscript n in t^n notates that the value being expressed occurs at time cycle n. It does *not* indicate t raised to the power n. We will continue to use superscipts in this manner, combining them with the subscripts used earlier to represent position. T_j^n will therefore be defined as the temperature in cell j at time step n, and similarly, T_j^{n+1} will represent the temperature in cell j at the time step $n+1$.

By using this notation and assuming that our heat source is always placed at the left, energy conservation can be expressed as

$$\begin{gathered}[\text{Heat Energy}]_j^{n+1} - [\text{Heat Energy}]_j^n = \\ [\text{Amount in}]_{j-1/2}^n - [\text{Amount out}]_{j+1/2}^n\,. \end{gathered} \tag{II-7}$$

Referring now to our principle of flux:

$$[\text{Amount in}]_{j-1/2}^n = [\text{Flux}]_{j-1/2}^n\,A\,dt \tag{II-8}$$

and

$$[\text{Amount out}]_{j+1/2}^n = [\text{Flux}]_{j+1/2}^n\,A\,dt \tag{II-9}$$

Using Fick's law to determine flux at $j + 1/2$ and $j - 1/2$, and using the equation for heat energy of a cell (II-5), we can express Eq. (II-7) as follows:

$$A\,\rho\,dx\,b\,T_j^{n+1} - A\,\rho\,dx\,b\,T_j^n = k\frac{(T_{j-1}^n - T_j^n)}{dx}\,A\,dt - k\frac{(T_j^n - T_{j+1}^n)}{dx}\,A\,dt\,. \qquad \text{(II-10)}$$

This equation can be algebraically manipulated to obtain

$$T_j^{n+1} - T_j^n = \frac{kdt}{b\rho\,dx^2}(T_{j-1}^n - T_j^n - T_j^n + T_{j+1}^n)\,. \qquad \text{(II-11)}$$

$\frac{k}{b\rho}$ is often called the THERMOMETRIC CONDUCTIVITY of a material and is represented by the Greek letter σ. Thus, our conservation equation in final form appears as follows:

$$T_j^{n+1} - T_j^n = (\sigma\frac{dt}{dx^2})(T_{j-1}^n - 2T_j^n + T_{j+1}^n) \qquad \text{(II-12)}$$

This equation expresses heat flow in a manner that can be computationally solved. Based upon our knowledge of the previous time step, this equation allows us to calculate the new temperatures for every zone along the rod. By carrying out this equation repeatedly, the overall flow of heat can be observed.

Based on our discussion so far, it is now possible to begin writing the finite-difference code itself; but before this process is begun, let us first examine the nature of our equation. Although this equation has been generated from basic principles, it is obtained more often through the manipulation of partial differential equations. While not necessary to the writing of simple finite-difference codes, these partial-differential equations (P.D.E.'s) give scientists greater insight into simple systems and allow for analysis of much more complicated physical phenomena. Because these equations are continually being applied to finite-difference codes, it is important that they be examined and related to the problem at hand.

C. Partial-Differential Equations

For those familiar with partial-differential equations and their use, the following discussion of heat flow in analytical terms will serve to provide a different viewpoint into the construction of our finite-difference codes. However, this section is not essential to the writing of this code and should, therefore, not deter the reader who is unfamiliar with these expressions. Such a reader should try to work through these concepts without intimidation; they are merely provided as an alternate method to examining this problem.

Going back to Eq. (II-11) and distributing the dx^2 term among the temperatures, we obtain the following equation:

$$\frac{T_j^{n+1} - T_j^n}{dt} = \sigma \left[\frac{\frac{T_{j+1}^n - T_j^n}{dx} - \frac{T_j^n - T_{j-1}^n}{dx}}{dx} \right] . \qquad \text{(II-13)}$$

By changing our nomenclature to more clearly represent T as a function of position and time, we can rewrite T_j^n as $T(x_j, t^n)$, T_{j+1}^n as $T(x_j + dx, t^n)$, and T_j^{n+1} as $T(x_j, t^n + dt)$. Our equation now takes the form

$$\frac{T(x_j, t^n + dt) - T(x_j, t^n)}{dt} = \sigma \left[\frac{\frac{T(x_j+dx,t^n) - T(x_y,t^n)}{dx} - \frac{T(x_y,t^n) - T(x_j-dx,t^n)}{dx}}{dx} \right] . \qquad \text{(II-14)}$$

Using the definition of the derivative of $f(x)$, namely

$$\frac{df}{dx} \equiv \lim_{dx \to 0} \frac{f(x + dx) - f(x)}{dx} ,$$

we take the limit as dt and $dx \to 0$ and obtain the following terms:

$$\lim_{dt \to 0} \frac{T(x_j, t^n + dt) - T(x_j, t^n)}{dt} = \frac{\partial T}{\partial t} \qquad \text{(II-15)}$$

$$\lim_{dx \to 0} \frac{T(x_j + dx, t^n) - T(x_j, t^n)}{dx} = \left(\frac{\partial T}{\partial x} \right)_{j+1/2} \qquad \text{(II-16)}$$

$$\lim_{dx \to 0} \frac{T(x_j + t^n) - T(x_j - dx, t^n)}{dx} = \left(\frac{\partial T}{\partial x} \right)_{j-1/2} . \qquad \text{(II-17)}$$

Thus Eq. (II-14) can be rewritten:

$$\frac{\partial T}{\partial t} = \sigma \left[\frac{\left(\frac{\partial T}{\partial x}\right)_{j+1/2} - \left(\frac{\partial T}{\partial x}\right)_{j+1/2}}{dx} \right] . \qquad \text{(II-18)}$$

Once again taking the limit as $dx \to 0$:

$$\frac{\partial T}{\partial t} = \sigma \frac{\partial^2 T}{\partial x^2} . \tag{II-19}$$

This is the heat-flow equation for a single direction. Starting with this equation, one would have been able to work backwards, choosing "finite differences" for each derivative and eventually generating Eq. (II-12). Derived from the same principles as our finite-difference equations, partial-differential equations provide a different outlook from which to approach computation.

D. Computational Implementation of Equations

Having derived an expression for heat flow in finite-difference form, the question still remains of how it will be computationally implemented. To complete this final stage in the writing of our code, three major issues must be examined: boundary conditions, redefinition of variables, and the structure of the program itself.

Our first major issue is the construction of boundary conditions. As was previously discussed, boundary conditions represent the external conditions that act to change a system. This representation is accomplished by the placing of zones beyond the normal boundaries of the array. The values of these FICTITIOUS ZONES or GHOST ZONES are chosen in such a way that they accurately express the external environment of the system in question. In this chapter, the conditions to be simulated will be a heat source at the left and an uninsulated area at the right of the rod.

While the temperature in each true zone within the rod will be determined by successive calculations of Eq. (II-12), the values at the fictitious zones will be calculated to represent fixed temperatures at either end of the array. In order to determine these values, consider the situation at either end of the array as represented in Fig. II-4.

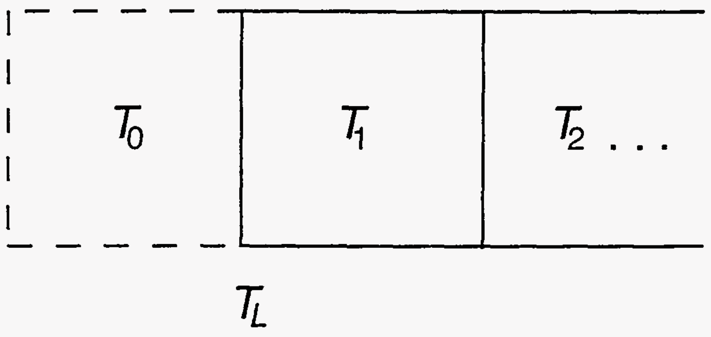


Fig. II-4

In this diagram, T_L represents the temperature along the left end of the rod, the value that should remain constant throughout the simulation. Although not present in the actual array, this constant temperature can be thought of as $T_{1/2}$, the average of the temperatures at zone T_1 and ghost zone T_0. Therefore T_L can be expressed as follows:

$$T_L = \frac{T_0 + T_1}{2} \, . \tag{II-20}$$

Solving this equation for T_0,

$$T_0 = 2T_L - T_1 \, . \tag{II-21}$$

Similarly, if T_R is defined as the temperature along the right end of the rod, $T_{\bar{j}+1}$ can be expressed as follows:

$$T_{\bar{j}+1} = 2T_R - T_{\bar{j}} \, . \tag{II-22}$$

By implementing these two equations, boundary conditions can be expressed for both the left and right of the system. Expressions for the other surfaces of the rod will not be needed, as they are assumed to be completely insulated, thus reducing the problem to one dimension.

The second major issue in solving of Eq. (II-12) computationally is the redefinition of variables. Thus far in this chapter, our equations have been represented in a manner that is not accepted by the FORTRAN programming language. Therefore certain modifications must be made in the way that various variables are represented; they must be redefined in terms of computationally accepted symbols:

$$\bar{j} \equiv \text{jbar}$$

$$\sigma \equiv \text{sig}$$

T_j^n, the temperature at zone j at time cycle n, will now be defined as T(j), an element in an array T defined from T(0) to T(jbar +1). Likewise, T_j^{n+1} will be defined as Tnew (j), an element in an array defined from Tnew (1) to Tnew (jbar).

The following new variables will also be defined:

T0 ≡ the intial temperature of the rod

stime ≡ the time at which the program ceases to run

ptime ≡ the time between successive displayings of the values of the zones in the array

pt ≡ a counter for ptime

st ≡ a counter for stime

Our finite-difference code will be divided into five sections, each with a clearly defined task. The first of these sections is the initialization procedure that dimensions the arrays and assigns initial values to all variables. This initialization is done in a subroutine that is called only once at the beginning of the program.

The second section of our code sets up a loop that repeats each time cycle. This section determines if the time counters pt and st have reached ptime and stime respectively and then increments the counters. If pt has reached ptime, it is reset to zero, and the current array of zones is sent to the output subroutine. If st reaches stime, the program terminates.

The third major section is the definition of boundary conditions, which occurs after the test for ptime and stime and before the actual computation of the next time cycle. In our particular program, this section should carry out Eq. (II-21) and Eq. (II-22) on the array T, updating values for the ghost zones at each time cycle.

The fourth section is the portion of the program that implements Eq. (II-12), which also occurs within the time-counting loop. This implementation is made up of two loops, the first of which assigns *Tnew* according to this equation, and the second of which transfers the values of *Tnew* back into *T*. The code for this section is as follows:

```
        do 100 j =1, jbar
        Tnew(j) = T(j) + sig*dt/dx**2 * (T(j+1) + T(j-1)-2*T(j))
  100  continue
        do 200 j=1, jbar
        T(j) = Tnew(j)
  200  continue
```

This two-loop structure is essential to the successful computation of T at the new time step. If one were to forego the computation of Tnew and directly compute T, the temperature terms at the right hand of the equation would not exist at the same time cycle. While T(j+1) and T(j) would still be at time n, T(j-1), having already been computed in the previous iteration of this do loop, would exist at time $n + 1$. By creating a second array and moving these values into T after they are all computed, we are able to avoid this problem.

The final section in our code is the output subroutine, which occurs when pt=ptime. This procedure could contain various graphics routines, write results to an output file, or simply display the various array values on the screen. A diagram of these sections and their interactions appears in Fig. II-5.

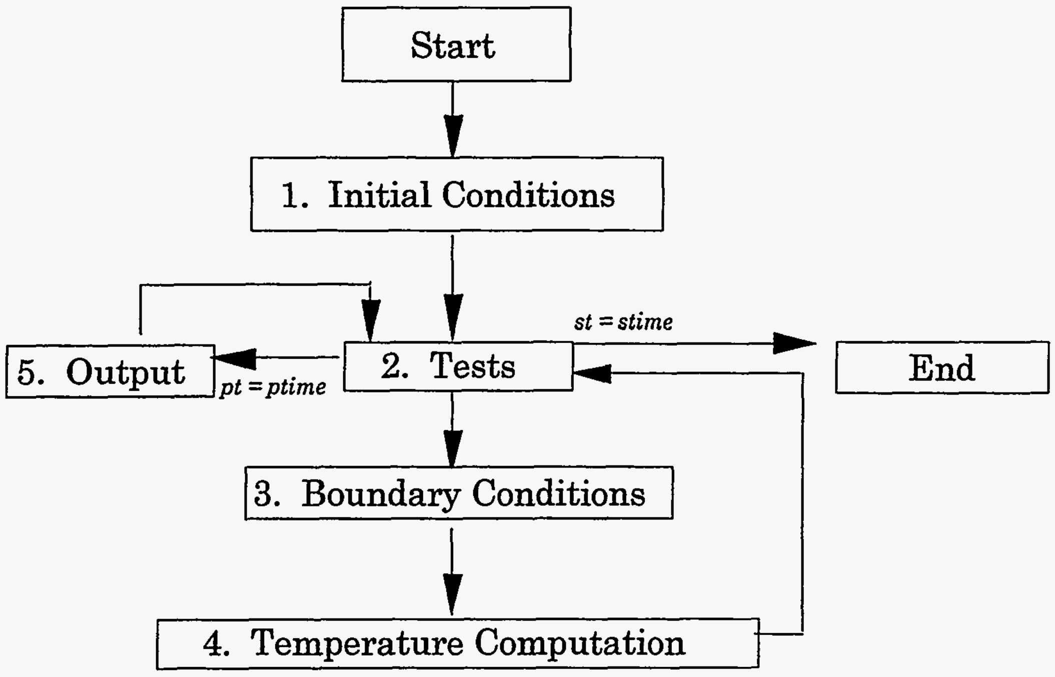


Fig. II-5

E. Programming and Results

We have now reached a point where the reader should be able to write his own finite-difference code for heat transfer. In this work a limited series of examples are examined in order to demonstrate the output of our code.

Figures II-6 through II-10 below show the results of a simulation of an insulated rod that is originally at 0°C, with a fixed temperature at the left (T_L) of 400°C, fixed temperature at the right (T_R) of 0°C, a thermometric conductivity (σ) of 1.0 m^2 per sec, a zone length (dx) of one meter, a $\bar{j}$ of 50, and a time step (dt) of 0.1 seconds. Results are shown at 10, 50, 100, 250, and 1000 seconds.

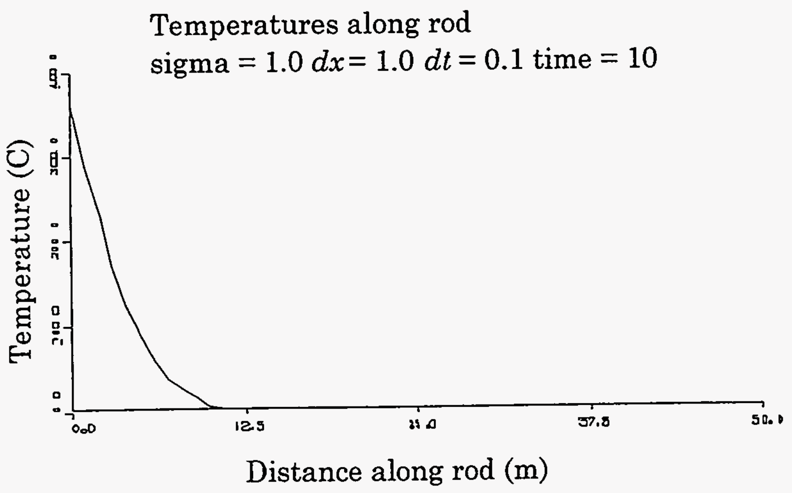


Figure II-6

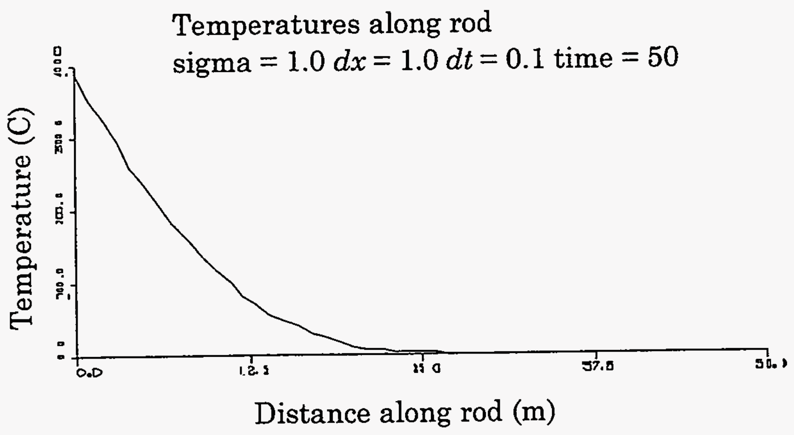


Figure II-7

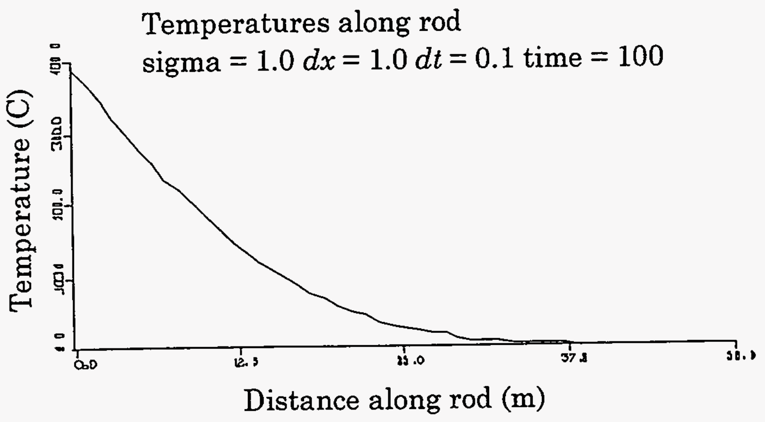


Figure II-8

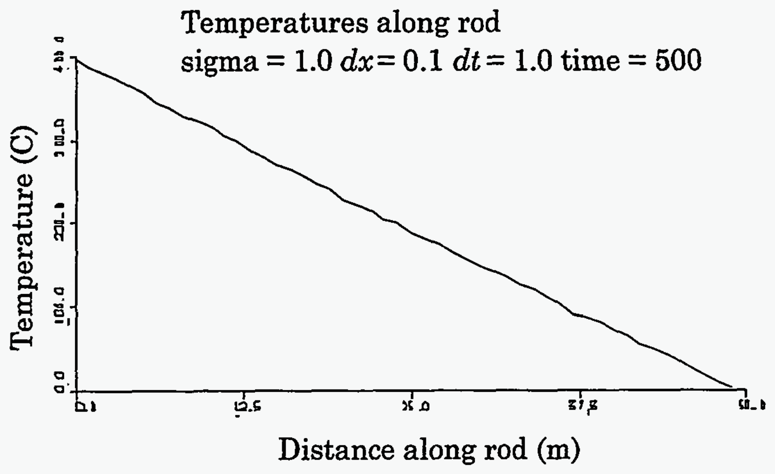


Figure II-9

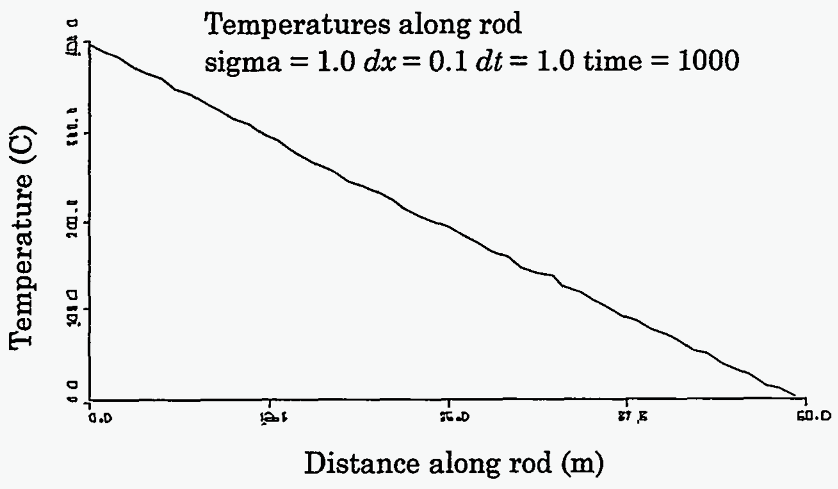


Figure II-10

Notice how the zone temperatures approach a straight line as the simulation progresses. Such a line is the final steady-state solution of this system, regardless of thermometric conductivity. Solutions at earlier time steps can be approximated using Eq. (III-55), which is derived at the end of Chapter III.

In the next three graphs, the effects of changes in time step are shown in relation to a simulation which is otherwise identical to the one above. Figure II-11 shows a temperature curve for a system at a time of 100 seconds with a time step of 0.495, just under 1/2.

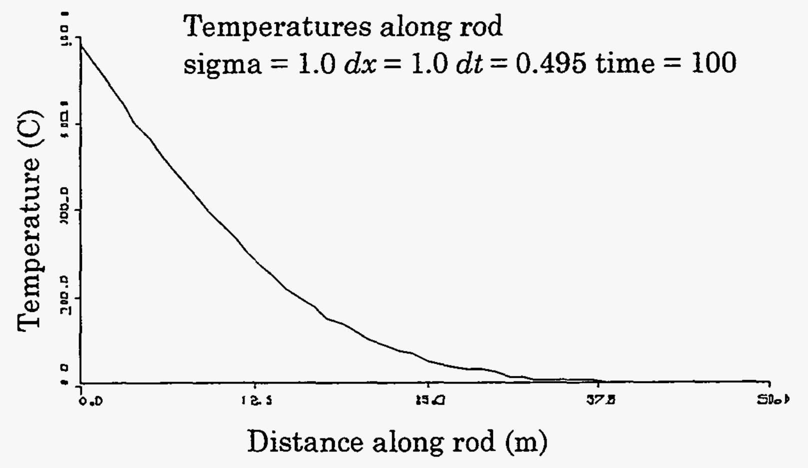


Figure II-11

This figure is almost identical to Fig. II-8, indicating that there is little difference between the results obtained with a time step of 0.1 and the results obtained with a time step of 0.495. Results are quite different, however, when a time step of 0.5 is used, as in Fig. II-12.

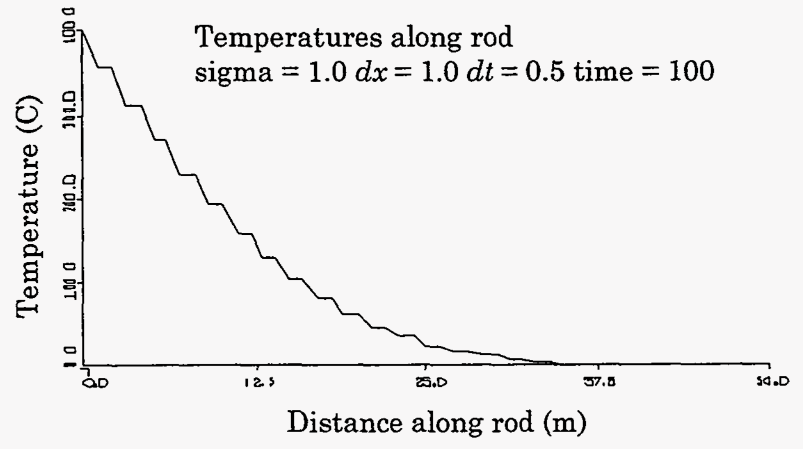


Figure II-12

The stair-step type temperatures that can be seen in this graph are a result of a numerical instability. This instability becomes even more violent when the time step is further increased to 0.505, as in Fig. II-13.

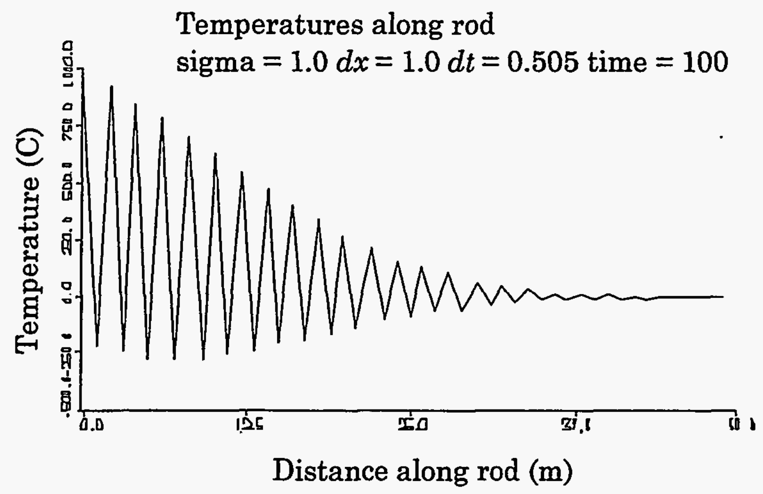


Figure II-13

Notice that in this figure the highest temperatures are much greater than 400°C, whereas the lowest temperatures are below −250°C. Obviously, this does not accurately represent the transfer of heat down the rod.

The numerical instability seen in Figs. II-12 and II-13 arises whenever the quantity $\frac{\sigma dt}{dx^2}$ is greater than 1/2. The presence of this instability means that the more accurately one wishes to resolve a set of circumstances, the shorter the time step that must be chosen. This problem highly limits the sorts of cases that can be simulated, yet there is a method by which it can be overcome. The following chapter examines this numerical instability and discusses the use of an implicit method of solution—a method that increases the speed, accuracy, and applicability of our finite-difference codes.

III. NUMERICAL INSTABILITY AND IMPLICIT CALCULATIONS

A. A Graphical Explanation of the Diffusional Stability Condition

In the cases presented at the end of the last chapter, we discovered that our finite-difference code is numerically unstable when the value of $\frac{\sigma dt}{dx^2}$ exceeds 1/2. This constraint is known as the diffusional stability condition, and it is one of two important conditions that we will examine in our series of finite-difference codes.

Consider the simplest case possible for our simulation: that of a rod at a constant temperature, T_0, with this same temperature at either end. Now consider the case in which this system of constant temperatures is perturbed by slightly increasing the temperatures of the odd-numbered zones by an amount ϵ and slightly decreasing the temperatures of the even-numbered zones by the same amount. The result is a system such as depicted in Fig. III-1.

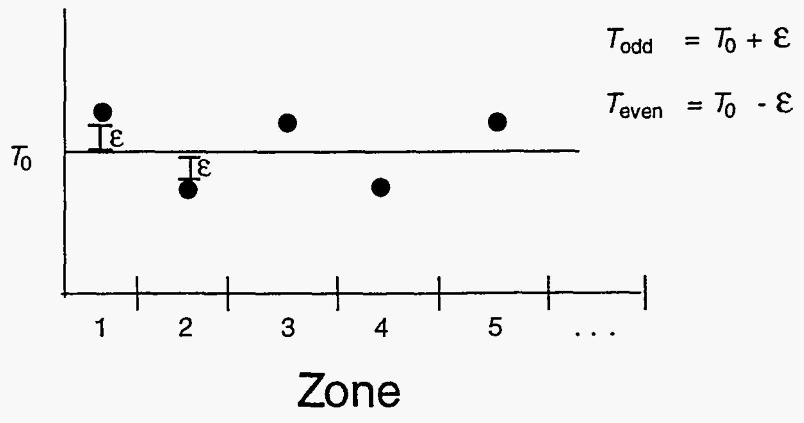


Fig. III-1

Let us now chose an odd numbered zone, j, and examine the calculation of its temperature at each time step. We begin with

$$T_j^{n+1} - T_j^n = \frac{\sigma dt}{dx^2}\left[T_{j+1}^n + T_{j-1}^n - 2T_j^n\right] \tag{II-12}$$

and substitute our new definition of T to obtain

$$T_j^{n+1} = T_0 + \epsilon + \frac{\sigma dt}{dx^2}\left[T_0 - \epsilon + T_0 - \epsilon - 2(T_0 + \epsilon)\right] , \tag{III-1}$$

which can be reduced to

$$T_j^{n+1} = T_0 + \epsilon - \frac{4\epsilon\sigma dt}{dx^2} \tag{III-2}$$

or

$$T_j^{n+1} = T_0 + \epsilon\left[1 - \frac{4\sigma dt}{dx^2}\right] . \tag{III-3}$$

If a constant ξ is defined such that

$$\xi \equiv \frac{4\sigma dt}{dx^2} ,$$

our equation becomes

$$T_j^{n+1} = T_0 + \epsilon[1 - \xi] . \tag{III-4}$$

Note that ξ in this equation is made up of all positive components; therefore, $\xi > 0$ and $(1 - \xi) < 1$ in all circumstances.

These constraints leave us with four cases to examine, the first of which occurs when $0 < \xi < 1$. In this case, $1 - \xi$ is a fraction between 0 and 1. T_j^{n+1} is therefore computed as $T_0 + \epsilon$ (fraction), yielding a value closer to T_0 than the previous time step. Subsequent iterations of this equation generate a series of temperatures such as shown in Fig. III-2.

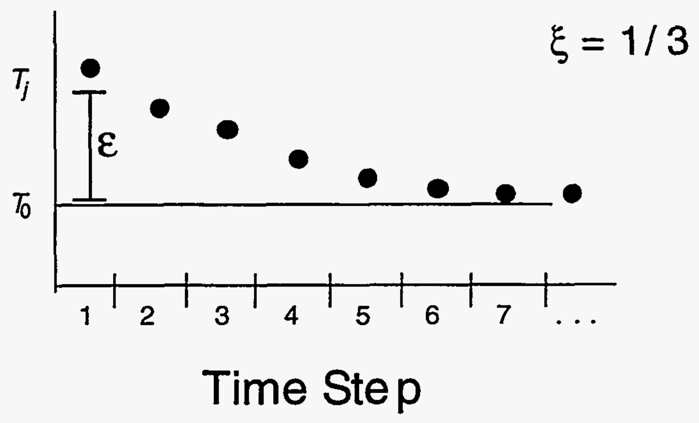


Fig. III-2

In this case, the temperature converges towards T_0, moving towards the array of constant temperatures that defines a correct solution.

In our second case $\xi = 1$, leaving us with the equation $T_j^{n+1} = T_0$. The graph of this case converges immediately, as illustrated in Fig. III-3.

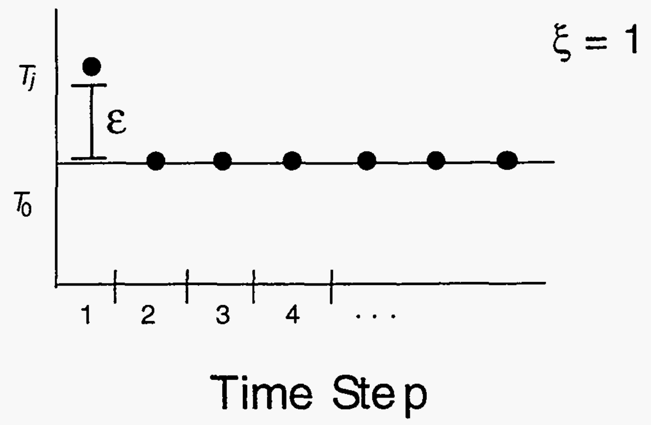


Fig. III-3

Our third curve is similar to the first and occurs when $1 < \xi < 2$. When this is true, $1 - \xi$ is again a fraction, but this time it is a number between 0 and -1. The result is a set of values which alternate above and below T_0 but converge toward that value as shown in Fig. III-4.

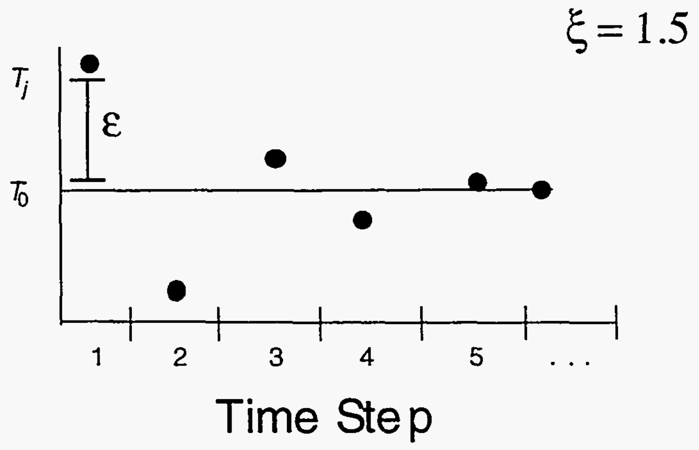


Fig. III-4

A corollary to this case occurs when $\xi = 2$. For this value the graph oscillates but does not converge, as in Fig. III-5. This case, while not convergent, is still considered to represent the bounds of numerical instability.

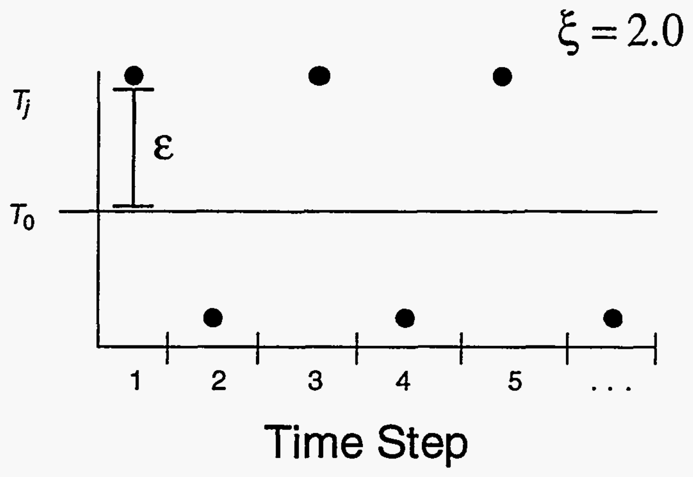


Fig. III-5

Our fourth and final case occurs as soon as this bound is crossed, when $\xi > 2$. In this set of circumstances $1-\xi < -1$, yielding values of T^{n+1} that not only oscillate but diverge from the correct solution. The graph of temperatures appears as in Fig. III-6, with values diverging until the program is terminated.

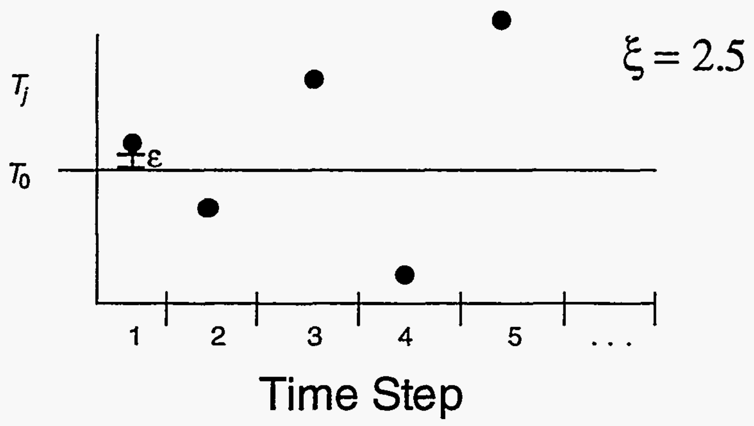


Fig. III-6

In order to avoid this condition, as well as the stable but nonconverging state pictured in Fig. III-5, we must choose ξ such that $\xi < 2$. Referring to our definition of ξ as $\frac{4\sigma dt}{dx^2}$, we obtain

$$\frac{4\sigma dt}{dx^2} < 2\,, \tag{III-5}$$

which is the diffusional stability condition

$$\frac{\sigma dt}{dx^2} < \frac{1}{2}\,. \tag{III-6}$$

We have therefore demonstrated graphically that this condition must be met for a solution to converge.

B. A Mathematical Derivation of the Diffusional Stability Condition

Once again, we will turn to a mathematical explanation to reinforce an argument that has been made graphically. This section, like section II-C, is not essential to the writing of our codes; it is simply another method of arriving at the diffusional stability condition and better explaining the manner in which it can be overcome. Again, one should follow as closely as possible, gaining familiarity with the application of various mathematical methods towards this problem.

We will begin with the heat-flow equation as expressed in Eq. (II-12), this time substituting our definition for ξ:

$$T_j^{n+1} = T_j^n + \frac{\xi}{4}\left(T_{j+1}^n + T_{j-1}^n - 2T_j^n\right)\,. \tag{III-7}$$

Let us examine the behavior of T_j^n with the following trial solution:

$$T_j^n = Ae^{i\,k\,j\,dx}e^{i\,w\,n\,dt}\,. \tag{III-8}$$

Here T_j^n represents T at a time step n and zone j, whereas $e^{i\,k\,j\,dx}$ and $e^{i\,w\,n\,dt}$ represent e raised to $i\,k\,j\,dx$ and $i\,w\,n\,dt$ respectively, where i is the imaginary number. If r is defined as

$$r \equiv e^{i\,w\,dt}\,,$$

Eq. III-8 becomes

$$T_j^n = Ae^{i\,k\,j\,dx} r^n \; . \qquad \text{(III-9)}$$

From this equation, we see that r must be between 1 and -1 for T_j^n to converge. If $r > 1$, the solution will diverge monotonically, moving farther and farther towards either positive or negative infinity. If $r < -1$, the solution will diverge in an oscillatory manner, alternating between positive and negative values but always moving away from convergence.

Keeping these restrictions on r in mind, let us now use our test definition of T_j^n to substitute for temperature terms in Eq. (III-7):

$$Ae^{i\,k\,j\,dx} r^{n+1} = Ae^{i\,k\,j\,dx} r^n + \frac{\xi}{4}\left[Ae^{ik(j+1)} r^n + Ae^{i\,k\,(j-1)} r^n - 2Ae^{i\,k\,j} r^n\right] \; . \qquad \text{(III-10)}$$

Dividing both sides by $Ae^{i\,k\,j\,dx} r^n$ gives

$$r = 1 + \frac{\xi}{4}\left[e^{i\,k\,dx} + e^{-i\,k\,dx} - 2\right] \; . \qquad \text{(III-11)}$$

Using the identity $e^{i\theta} = \cos\theta + i\sin\theta$ we can rewrite this equation as

$$r = 1 + \frac{\xi}{4}\left[2\cos k\,dx - 2\right] \qquad \text{(III-12)}$$

or

$$r = 1 - \frac{\xi}{2}\left[1 - \cos k\,dx\right] \; . \qquad \text{(III-13)}$$

Consider the extreme cases for the $\cos kdx$ term, namely $\cos k\,dx = +1$ and -1. If $\cos k\,dx = +1$, then Eq. (III-13) reduces to $r = 1$, a valid statement according to the restrictions that we have placed on r. This case poses no problems.

Taking the other extreme, $\cos kdx = -1$, we are left with

$$r = 1 - \xi \, . \qquad \text{(III-14)}$$

Because ξ is always positive, r can never exceed 1 in this case. It can, however, be less than -1, a problem which places the following condition on ξ:

$$-1 < 1 - \xi \tag{III-15}$$

or

$$\xi < 2 \,. \tag{III-16}$$

Using our definition of ξ, we find that Eq. (III-16) is simply another statement of the diffusional stability condition:

$$\frac{\sigma dt}{dx^2} < \frac{1}{2} \,. \tag{III-17}$$

Our analytical method arrives at precisely the same result as the pictorial analysis; the diffusional stability condition must be met in order to ensure numerical stability.

C. Implicit Calculations

We have now derived the diffusional stability condition mathematically as well as graphically and have demonstrated that it is essential to the numerical solving of Eq. (II-12). It is possible, however, to solve the heat-flow equation numerically without meeting this condition, by expressing the ξ terms of the heat-flow equation at time $n+1$ rather than at time n. In this method, heat flow is not represented by Eq. (II-12), but instead by the following:

$$T_j^{n+1} = T_j^n + \frac{\sigma dt}{dx^2}\left[T_{j+1}^{n+1} + T_{j-1}^{n+1} - 2T_j^{n+1}\right] \,. \tag{III-18}$$

Let us now examine this equation mathematically as we did in Section B. Inserting a similar trial solution and dividing by $Ae^{i\,k\,j\,dx}r^n$, we obtain

$$r = 1 + \frac{\xi}{4}\left[e^{ikdx}r + e^{ikdx}r - 2r\right] \,. \tag{III-19}$$

or

$$r = 1 + \frac{\xi}{4}\left[2r\cos kdx - 2r\right] \,, \tag{III-20}$$

which can be algebraically manipulated to obtain

$$r = \frac{1}{1 + \frac{\xi}{2}(1 - \cos kdx)} \,. \tag{III-21}$$

Examining the upper and lower limits for $\cos kdx$, we find that

$$r = 1 \tag{III-22}$$

or

$$r = \frac{1}{1+\xi} \,. \tag{III-23}$$

Because ξ is always positive, r will be between 0 and 1 in all cases, indicating that our solution will be numerically stable regardless of the value of $\frac{\sigma dt}{dx^2}$. Equation (III-18) will therefore remain stable at any resolution and time step; all that remains is to solve it numerically.

Equation (III-18) represents an IMPLICIT METHOD of calculation. In this method, values at the new time cycle are not directly calculated from old values as they were in the EXPLICIT METHOD used in the previous chapter. They are instead calculated using an iterative process that begins with a trial solution and modifies that solution with each iteration until it has converged to within a specified value. In our program this iteration will be done using Newton's Method.

Newton's Method is an iterative process that uses successive approximations to solve an equation in the form $f(x) = 0$. In Newton's method a trial value for x_1 is first chosen, then the following equation is applied iteratively:

$$x_{n+1} = x_n - \frac{f(x_n)}{\left(\frac{df}{dx}\right)_{x_n}} \,. \tag{III-24}$$

This equation generates successive approximations for x, each more accurate than the one before it. When x has converged to within a specified range, x_n is then taken as the final solution.

We can better understand how this method arrives at a solution by examining an example equation, $f(x) = x^2 - 2$. Choosing $x_1 = 2$ as a trial value, we will let the solution converge to within three decimal places.

$$x_{n+1} = x_n - \frac{x_n^2 - 2}{2x_n} \tag{III-25}$$

$$x_1 = 2.000$$

$$x_2 = 1.500$$

$$x_3 = 1.466$$

$$x_4 = 1.414$$

$$x_5 = 1.414$$

A plot of these values along the graph of the equation appears in Fig. III-7.

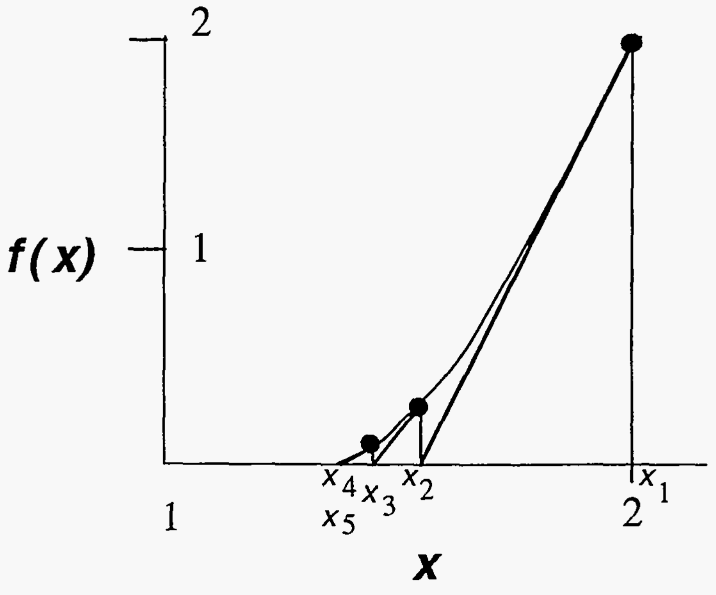


Fig. III-7

This figure illustrates how one solution is used to obtain the next. The tangent is used to approximate the graph of $f(x_n)$ at the value x_n, and x_{n+1} is given the value of

x for which the tangent crosses the x-axis. This value is then substituted for x and the process is repeated until it has converged.

Let us now apply this process to Eq. (III-18). Expressing this equation as a function of T_j^{n+1}, we obtain

$$f\left(T_j^{n+1}\right) = T_j^{n+1} - T_j^n - \frac{\sigma dt}{dx^2}\left[T_{j+1}^{n+1} + T_{j-1}^{n+1} - 2T_j^{n+1}\right] . \qquad \text{(III-26)}$$

Taking now the derivative with respect to T_j^{n+1}, we find that

$$f'\left(T_j^{n+1}\right) = 1 + \frac{2\sigma dt}{dx^2} . \qquad \text{(III-27)}$$

Using both of these definitions in Eq. (III-24), we are left with the final formula:

$$T_j^{n+1}(\text{new guess}) = T_j^{n+1}(\text{old guess}) - \left(\frac{f(T_{j(oldguess)}^{n+1})}{1 + \frac{2\sigma dt}{dx^2}}\right) . \qquad \text{(III-28)}$$

By using this formula iteratively, we can now compute values for T_j^{n+1} for any values of $\frac{\sigma\, dt}{dx^2}$.

D. Computational Implementation of the Implicit Method

Now that we have developed an implicit method for use in solving the heat-transfer equation, we can implement this method on the computer. We will do this by making small modifications to the heat-transfer code that has already been written.

We begin by defining a constant beta that is set during the initialization procedure. Beta is defined as

$$\beta \equiv \frac{1}{1 + \frac{2\sigma dt}{dx^2}}$$

and is used to avoid successive calculation of the denominator in Eq. (III-28) during iterations of Newton's method. Also in this procedure, we define a constant ftest that is equal to the margin of error to which our iterative procedure will converge. Typically ftest has a value of approximately 0.001 times some maximum value of T in the problem.

Besides these two definitions in the initialization procedure, most of the major modifications to the program occur in the computation section (referred to as Section 4 in the previous chapter). This section in its explicit form should be removed and replaced with an implicit section of code.

This implicit section should consist of a loop that makes the initial guesses for the temperatures at time $n+1$ and a loop that iterates until the values of T^{n+1} have converged to within ftest. The first loop is simply a do loop that defines the initial guesses for the new temperatures as the temperatures at the old time step. Thus,

$$\text{Tnew(j)} = \text{T(j)}\ . \tag{III-29}$$

This loop is then followed by an until loop that is constructed in the following manner. At the beginning of each iteration, a value fmax is set at zero. After this statement, the program moves into another loop that calculates $f(T)$ along every point along the rod and uses these values to calculate the next guess for the new temperatures. Also in this loop, the largest absolute value for $f(T)$ is stored in the variable fmax. After this loop, the program makes a check to see if fmax is less than ftest. If ftest is larger, the until loop ends; if fmax is larger, the loop is repeated. The code for this loop should be similar to the following:

```
100  fmax = 0.
       do 200 j= 1,jmax
          f = Tnew(j) - T(j) - (sig*dt/(dx*dx)) * (Tnew(j+1)+ T (j-1) - 2*T(j))
          fmax = amax1(abs(f),fmax)
          Tnew(j) = Tnew(j) - f * beta
200  continue
        if (fmax.gt.ftest) then goto 100
```

Notice that all the T terms on the right of the equation that sets f are actually temperature values at the present implicit iteration. T's and Tnew's are mixed due to the structure of the loop. Optionally, a counter for the number of iterations of the until loop can be added, terminating this iterative process when a maximum number of iterations is reached, regardless of the values of fmax.

When this loop has finally terminated, the T array is redefined with the values from the Tnew array, and the program moves on to the next time step. All other sections remain in the same form as in our original program. No other modifications are necessary to create a fully-implicit version of our one-dimensional heat-transfer code.

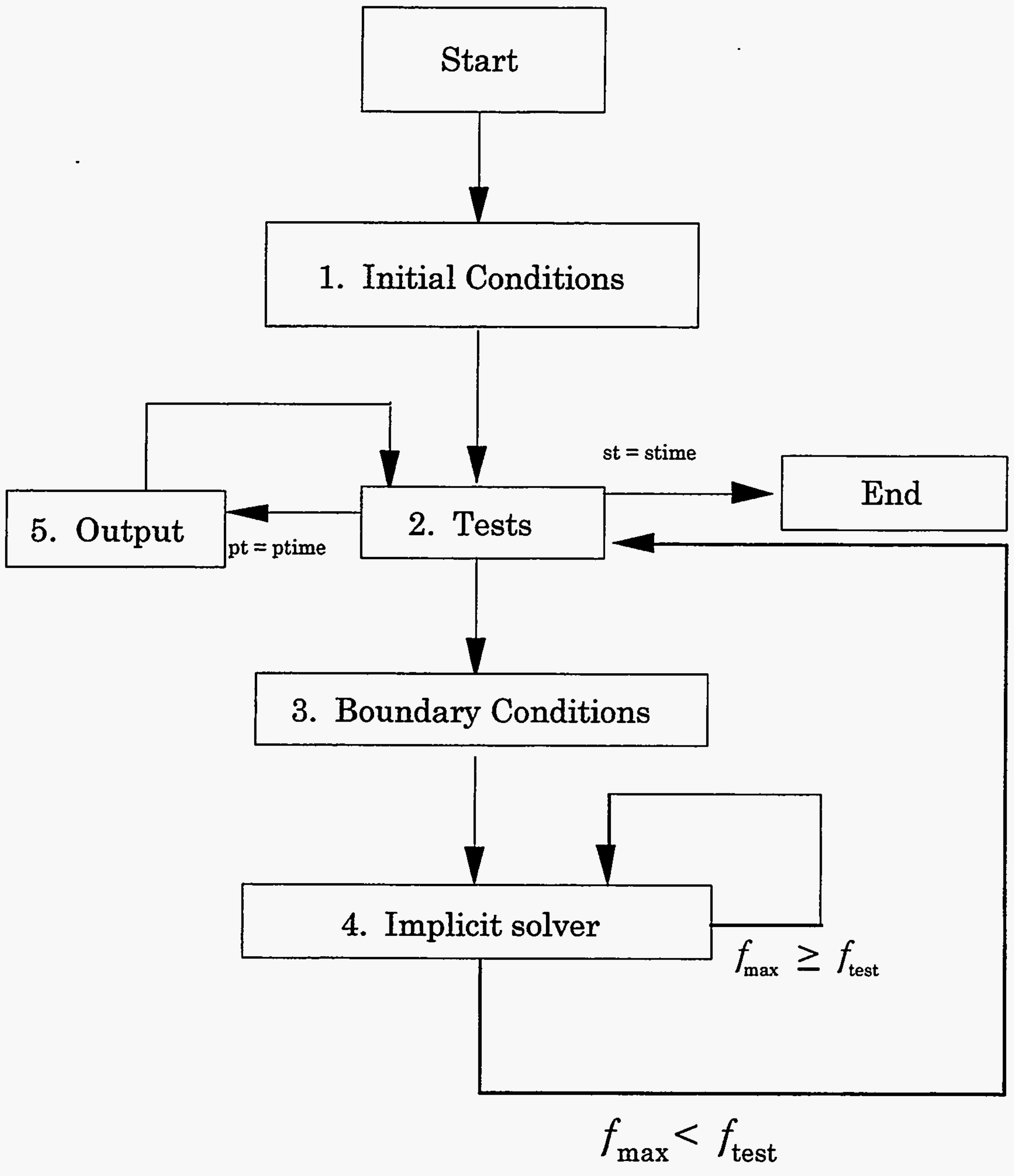


Fig. III-8

By using the implicit code with the same set of parameters as were present in Fig. II-12 ($T_0 = 0°C$, $T_1 = 400°C$, $T_r = 0°C$, $\sigma = 1\,\mathrm{m}^2/\mathrm{sec}$, $dx = 1$ m, $\bar{j} = 50$, $dt = 0.5$ sec, time $= 100$ sec, $\frac{\sigma dt}{\partial x^2} = 0.5$), the results shown in Fig. III-9 are obtained.

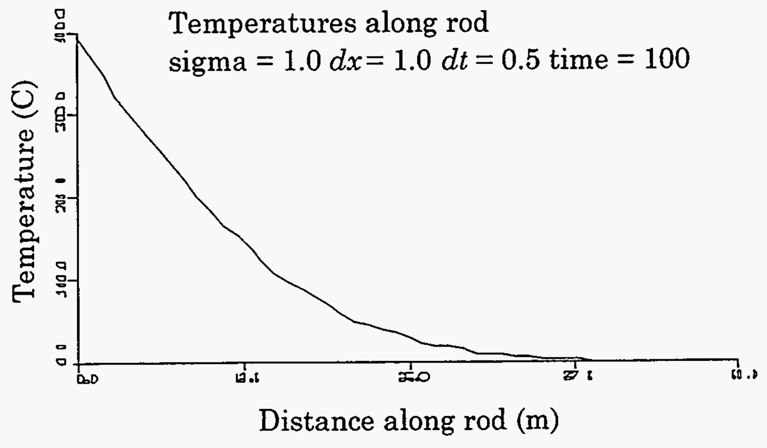


Fig. III-9

This figure helps to illustrate the numerical stability of this method. The system remains stable at a time step of 0.505 $\left(\frac{\sigma dt}{dx^2} = 0.505\right)$, as indicated in Fig. III-10.

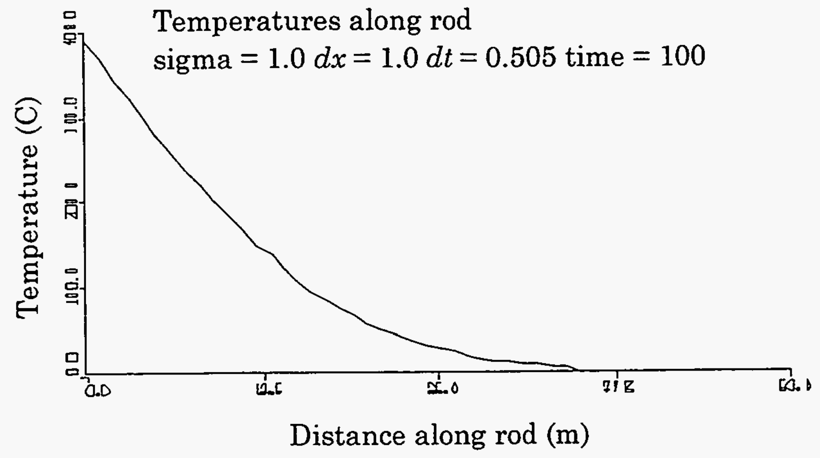


Fig. III-10

Even at a time step of 10 sec, where $\frac{\sigma dt}{dx^2}$ is equal to 10 and only 10 time cycles are computed up to time 100, the system remains numerically stable. The results in Fig. III-11 below appear almost identical to those calculated explicitly in Fig. II-11, yet the time step used is over twenty times as large.

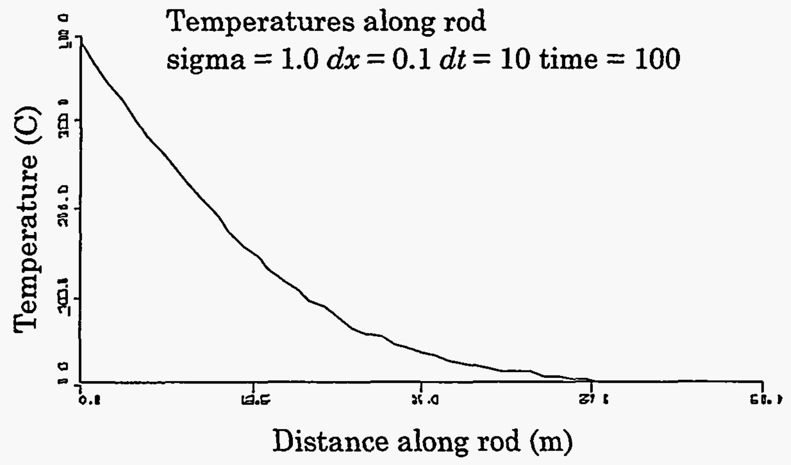


Fig. III-11

We see through example that implicit methods are able to generate results for sets of parameters that are numerically unstable when calculated explicitly. This technique will prove essential in later simulations, preventing the first of two major numerical instabilities that we will examine in our series of exercises.

E. Analytic Solution of the Heat-Flow Equation

In this section, we will be manipulating the heat-flow equation in order to generate an analytic solution that can be used to check the validity of our computational results. Once again, following the manipulation of this partial-differential equation is not essential to making use of the derived solution.

We begin with Eq. (II-19), the one-dimensional heat-flow equation in partial-differential form

$$\frac{\partial T}{\partial t} = \sigma \frac{\partial^2 T}{\partial x^2} \tag{II-19}$$

and make the assumption that T is a function of the single dimensionless quantity that includes σ, x, and t:

$$T = T(\xi) \tag{III-30}$$

where

$$\xi \equiv \frac{x}{\sqrt{\sigma t}} \, .$$

By making this definition, we are assuming that the rod is of infinite length, so that the length of the rod does not enter into these parameters. This assumption is made because the derivation of a solution for a finite rod is a much more involved process than a solution for the infinite case. For our purposes an analytic solution to the infinite rod case will prove to be sufficient.

Using Eq. (III-30), we can obtain expressions for its partial-derivatives. Differentiating with respect to t we obtain

$$\frac{\partial T}{\partial t} = \frac{dT}{d\xi}\frac{\partial \xi}{\partial t} = \frac{x}{2\sqrt{\sigma}}\left(-t^{3/2}\right)\frac{dT}{d\xi} \, . \tag{III-31}$$

Differentiating to obtain the second derivative with respect to x gives us

$$\frac{\partial T}{\partial x} = \frac{dT}{d\xi}\frac{\partial \xi}{\partial x} = \frac{1}{\sqrt{\sigma t}}\frac{dT}{d\xi} \tag{III-32}$$

and

$$\frac{\partial^2 T}{\partial x^2} = \frac{1}{\left(\sqrt{\sigma t}\right)^2}\frac{d^2 T}{d\xi^2} \, . \tag{III-33}$$

Equations (III-31) and (III-33) are substituted into Eq. (II-19) to obtain

$$\frac{x}{2\sqrt{\sigma}}\left(-t^{3/2}\right)\frac{dT}{d\xi} = \frac{\sigma}{\sigma t}\frac{d^2 T}{d\xi^2} \tag{III-34}$$

or

$$-\frac{\xi}{2}\frac{dT}{d\xi} = \frac{d}{d\xi}\left(\frac{dT}{d\xi}\right) \, . \tag{III-35}$$

If we define a variable y such that

$$y \equiv \frac{dT}{d\xi} \, ,$$

Eq. (III-35) becomes

$$-\frac{\xi}{2}y = \frac{dy}{d\xi} \tag{III-36}$$

or

$$-\frac{\xi}{2}d\xi = \frac{1}{y}dy \, , \tag{III-37}$$

which can be integrated to obtain

$$-\frac{\xi^2}{4} = \ln y + C \, , \tag{III-38}$$

where C is a constant.

Exponentiating both sides of this equation gives us:

$$e^{\frac{-\xi^2}{4}} = e^{\ln y + C} \tag{III-39}$$

or

$$e^{\frac{-\xi^2}{4}} = yK_1 \, , \tag{III-40}$$

where K_1 is a constant. We then use our definition of y and multiply both sides of the equation by $d\xi$ to obtain

$$e^{\frac{-\xi^2}{4}} d\xi = K_1 dT \, . \tag{III-41}$$

This can be integrated to obtain

$$\int_{\xi_1}^{\xi_2} e^{\frac{-\xi^2}{4}} d\xi = K_1 T + K_2 \, , \tag{III-42}$$

where K_2 is another constant. If we choose ξ_1 to represent ξ at a distance of zero from the end of the rod this equation becomes

$$\int_{0}^{\frac{x}{\sqrt{\sigma t}}} e^{\frac{-\xi^2}{4}} d\xi = K_1 T + K_2 \, . \tag{III-43}$$

which can be simplified by defining a variable z such that

$$z \equiv \frac{\xi}{2} \, .$$

Equation (III-43) then becomes

$$2\int_{0}^{\frac{x}{2\sqrt{\sigma t}}} e^{-z^2}dz = K_1T + K_2 \,. \tag{III-44}$$

To determine the values K_1 and K_2, we examine two test cases. In the case where x is 0, the temperature is equal to that of the wall at the left end of the rod. We then have:

$$2\int_{0}^{0} e^{-z^2}dz = K_1T_L + K_2 \tag{III-45}$$

or

$$K_2 = -(K_1T_L) \,. \tag{III-46}$$

In the case where we are at an infinite distance from the heat source at one end of the rod, the temperature is equal to the initial temperature specified for the rod:

$$2\int_{0}^{\infty} e^{-z^2}dz = K_1T_0 + K_2 \,, \tag{III-47}$$

which reduces to

$$2\left(\frac{\sqrt{\pi}}{2}\right) = K_1T_0 + K_2 \tag{III-48}$$

or

$$\sqrt{\pi} - K_1T_0 = K_2 \,. \tag{III-49}$$

Setting Eqs. (III-46) and (III-49) equal to each other, we obtain

$$-K_1T_L = \sqrt{\pi} - K_1T_0 \tag{III-50}$$

or

$$K_1 = \frac{\sqrt{\pi}}{(T_0 - T_L)} \,. \tag{III-51}$$

K_2 can then be obtain by substituting into Eq. (III-46):

$$K_2 = -\frac{T_L\sqrt{\pi}}{(T_0 - T_L)} \,. \tag{III-52}$$

Substituting both of these values into Eq. (III-44) we obtain

$$2\int_0^{\frac{x}{2\sqrt{\sigma t}}} e^{-z^2}dz = (T - T_L)\left(\frac{\sqrt{\pi}}{(T_0 - T_L)}\right) \qquad \text{(III-53)}$$

or

$$T = T_L + (T_0 - T_L)\left(\frac{2}{\sqrt{\pi}}\right)\int_0^{\frac{x}{2\sqrt{\sigma t}}} e^{-z^2}dz\ . \qquad \text{(III-54)}$$

$\left(\frac{2}{\sqrt{\pi}}\int_0^{\frac{x}{2\sqrt{\sigma t}}} e^{-z^2}dz\right)$ in this equation is a form of the PROBABILITY INTEGRAL, also called the ERROR FUNCTION. This term is not integrable in terms of simple polynomials, but it can be "solved" by defining

$$erf(a) \equiv \frac{2}{\sqrt{\pi}}\int_0^a e^{-z^2}dz\ ,$$

where $erf(a)$ can be determined as in the following table.

$erf(a)$

a	$erf(a)$	a	$erf(a)$
0.00	0.0000	0.9	0.7969
0.05	0.0564	1.0	0.8427
0.10	0.1125	1.1	0.8801
0.15	0.1680	1.2	0.9103
0.20	0.2227	1.3	0.9340
0.25	0.2763	1.4	0.9523
0.30	0.3286	1.5	0.9661
0.35	0.3794	1.6	0.9764
0.40	0.4283	1.7	0.9838
0.45	0.4755	1.8	0.9891
0.50	0.5205	1.9	0.9928
0.55	0.5633	2.0	0.9953
0.60	0.6039	2.1	0.9970
0.65	0.6420	2.2	0.9981
0.70	0.6778	2.3	0.9987
0.75	0.7112	2.4	0.9994
0.80	0.7421	2.5	0.9996

Our final solution to the heat-flow equation is then

$$T = T_L + (T_0 - T_L)\ erf\left(\frac{x}{2\sqrt{\sigma t}}\right) . \tag{III-55}$$

This equation can be used to check the accuracy of our numerical results. It represents the infinite rod case in which the temperature wave is not affected by the conditions at the right end of the rod. At early time steps, the temperatures in our finite-rod simulation should approximate those generated by this equation. Solutions at late time steps, as we saw in Chapter II, should approach a straight line. By examining the results generated by our code in both these circumstances, we can verify the validity of our finite-difference calculations.

IV. LAGRANGIAN FLUID DYNAMICS

A. Fluid Flow and Lagrangian Methods

Up to this point our finite-difference codes have dealt strictly with the equation of heat transfer [Eq. (II-19)], but heat flow is only one of many phenomena that can be modeled using the finite-difference method. In the following several chapters, we will be looking at another physical phenomenon that can be simulated in this manner: the motion of FLUIDS.

For our purposes, we will define a fluid as anything that is infinitely deformable or malleable. This means that, while a fluid may resist moving from one shape to another, it resists the same amount in all directions and in all shapes. Fluids can be either COMPRESSIBLE or INCOMPRESSIBLE. An incompressible fluid is one that does not change its density much when pressure is applied to it, meaning that the fluid is moving at a velocity much less than its sound speed. A compressible fluid is one that undergoes a large change in its density as pressure is applied to it, meaning that the fluid is moving at a speed that is comparable to its sound speed. We will be dealing with compressible fluids in this chapter.

Our simulation will be of a system that can be reduced to one dimension: a piston moving in a long cylinder that is filled with gas. The compression of gas in this manner can by dealt with in one of two ways: through LAGRANGIAN or EULERIAN methods.

In an Eulerian code, zones remain fixed in space throughout the simulation. Fluids move in and out of the zone at various rates, causing the mass contained in a particular zone to change as the simulation progresses. All physical quantities are fluxed between cells, but the position of the cells at all time steps remains the same. We will examine this method in Chapter V.

Another method for simulating this situation is the Lagrangian technique. In a Lagrangian code the positions of zones vary between each time step. As fluids are compressed and decompressed, the zones move accordingly, maintaining an equal mass throughout the simulation. In a Lagrangian calculation, the energy, momentum, and position of a given zone vary from time step to time step; only the mass contained by the zone is held fixed. The Lagrangian technique is the one that is used in this chapter.

B. Description of Equations Used in Lagrangian Fluid Flow

In order to derive the equations that are used in a one-dimensional Lagrangian code, we must first define a group of variables and coordinates similar to those used in our first two simulations. We again have a one-dimensional system of zones, each zone representing a certain section of the system being simulated. The system appears as in Fig. II-3, with a series of $\bar{j}$ true zones and ghost zones appearing at 0 and $\bar{j}+1$

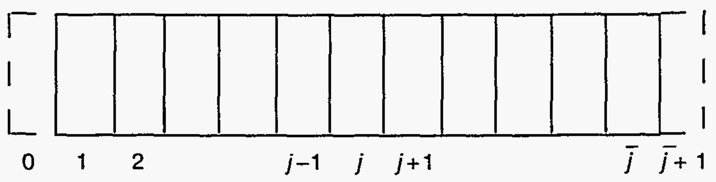


Figure IV-1

The variables that will be applied to this system, however, are quite different from those of the heat-transfer problem. In the fluid-flow case there is no longer a single array of temperatures, but instead a group of arrays that represent position, pressure, velocity, density, internal energy, and viscous pressure. The definition of these variables over a zone j is shown in Fig. IV-2.

In this figure:

$x_{j+1/2} \equiv$ position of cell wall to the right of zone j

$u_{j+1/2} \equiv$ velocity at cell wall to the right of zone j

$p_j \equiv$ pressure of zone j

$I_j \equiv$ internal energy per unit mass of zone j

$q_j \equiv$ viscous pressure of zone j

$\rho \equiv$ density of zone j

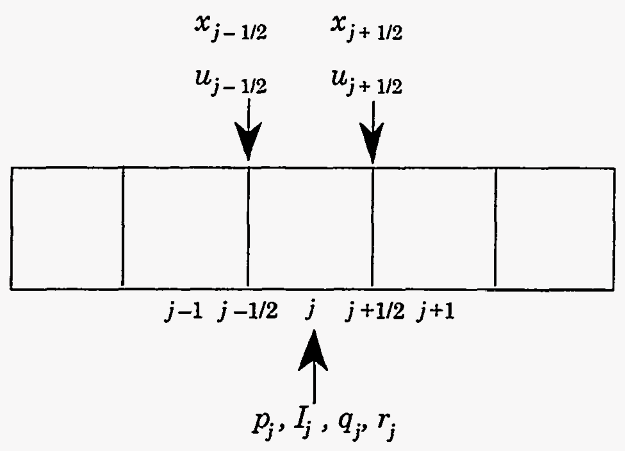


Figure IV-2

Note that u and x are located at the walls of the cells while the rest of the variables are located at the centers. These positions will be important in determining the relationship among these various quantities.

With our variables defined as in Fig. IV-2, the equations that relate them to one another can be derived. Consider the relationship between x and u: x is the array of wall positions and u is the array of the time rate of change of those wall positions, i.e., velocity. From these definitions, we see that

$$\frac{\partial x}{\partial t} = u \, . \tag{IV-1}$$

Finite-differencing this equation gives us

$$\frac{x^{n+1}_{j+1/2} - x^{n}_{j+1/2}}{dt} = u^{n}_{j+1/2} \, , \tag{IV-2}$$

which can be rewritten as an equation for position in terms of velocity:

$$x^{n+1}_{j+1/2} = x^{n}_{j-1/2} + u^{n}_{j+1/2} dt \, . \tag{IV-3}$$

This is the first important equation of our Lagrangian fluid flow code.

The next equation follows from Newton's second law of motion, (Force = Mass × Acceleration), and the definition of pressure, (pressure $\equiv \frac{\text{Force}}{\text{area}}$). In our code we define a

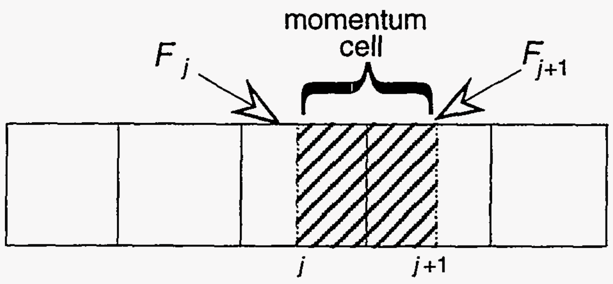


Figure IV-3

momentum cell whose center lies at the boundary between two normal cells and define F_j and F_{j+1} as the force at the right and the left of the j momentum cell. A momentum cell is depicted in Fig. IV-3.

By the definition of pressure, F_j and F_{j+1} are rewritten in terms of variables defined in Fig. IV-2:

$$F_j = (p_j + q_j)\,A \tag{IV-4}$$

$$F_{j+1} = (p_{j+1} + q_{j+1})\,A\ , \tag{IV-5}$$

where A is the surface area of a cell wall. Note that here we use the sum of the physical pressure and the viscous pressure, an additional pressure that is necessary to achieve numerical stability. The viscous pressure will be discussed in more detail in section C.

Using Newton's second law and noting that acceleration is the time rate of change of velocity, we obtain

$$F_{j+1/2} = m\left(\frac{u^{n+1}_{j+1/2} - u^{n}_{j+1/2}}{dt}\right)\ , \tag{IV-6}$$

where $F_{j+1/2}$ is equal to the net force at the cell wall $j + 1/2$. In the case where there are no outside forces such as gravity in the x-direction, the net force at $j + 1/2$ is equal to the force pushing the momentum cell from the left (F_j) minus the force by which it pushes the next momentum cell on the right (F_{j+1}):

$$F_{j+1/2} = F_j - F_{j+1} \; . \tag{IV-7}$$

Combining Eq. (IV-6) and Eq. (IV-7) gives us

$$F_j - F_{j+1} = m\left(\frac{u^{n+1}_{j+1/2} - u^n_{j+1/2}}{dt}\right) , \tag{IV-8}$$

or, by substituting for F_j and F_{j+1},

$$\left(q_j + p_j - q_{j+1} - p_{j+1}\right) A = m\left(\frac{u^{n+1}_{j+1/2} - u^n_{j+1/2}}{dt}\right) . \tag{IV-9}$$

Defining the quantity M as $\frac{m}{A}$ and solving for $u^{n+1}_{j+1/2}$, we are left with the expression

$$u^{n+1}_{j+1/2} = u^n_{j+1/2} + \frac{dt}{M}\left(p^n_j + q^n_j - p^n_{j+1} - q^n_{j+1}\right) \tag{IV-10}$$

which gives change in u in terms of variables used in Fig. IV-2.

An expression for ρ can be obtained by again using our definition of M. Because density is equal to M divided by the width of a zone, it follows that

$$\rho^n_j = \frac{M}{x^n_{j+1/2} - x^n_{j-1/2}} , \tag{IV-11}$$

which is the Lagrangian density equation.

An expression for I, the internal energy per unit mass of a cell, can be derived by appealing to the definition of internal energy. I can be defined as the difference between the total energy per unit mass and the kinetic energy per unit mass contained in a cell, which is the same as the heat energy per unit mass. We can therefore use the first law of thermodynamics,

$$\Delta I = \Delta E = Q - p\Delta V , \tag{IV-12}$$

where ΔE is the change in heat energy, Q is the heat received from both conduction from an outside source (which will not be important in this simulation) and dissipation of mean flow kinetic energy, p is the pressure, and ΔV is the change in volume.

We now make use of the variable q, which we called the viscous pressure earlier in this chapter. One way of thinking of this pressure is as $-Q/\Delta V$, or the increase in heat energy over the compressive (or negative) change in volume. Using q, Eq. (IV-12) can be written as

$$\Delta I = -(q+p)\Delta V \; . \tag{IV-13}$$

Because the total internal energy is equal to $m \times I$, this equation can be rewritten in the following differential form:

$$m\frac{\partial I}{\partial t} = -(q+p)\frac{\partial V}{\partial t} \; . \tag{IV-14}$$

In finite-difference form, Eq. (IV-14) becomes

$$m\left(\frac{I_j^{n+1} - I_j^n}{dt}\right) = -(q+p)A\left[\frac{\partial x_{j+1/2}}{\partial t} - \frac{\partial x_{j-1/2}}{\partial t}\right] \; . \tag{IV-15}$$

Using our definition of M as $\frac{m}{A}$ and $u_{j+1/2}$ as $\frac{\partial x_{j+1/2}}{\partial t}$, we solve for I_j^{n+1} and obtain the Lagrangian internal energy equation in finite-difference form:

$$I_j^{n+1} = I_j^n + \frac{dt}{M}\left(q_j^n + p_j^n\right)\left(u_{j-1/2}^n - u_{j+1/2}^n\right) \; . \tag{IV-16}$$

To obtain an equation for pressure (p), we employ the ideal gas law, namely

$$pV = n\mathrm{R}T \; , \tag{IV-17}$$

where p is the pressure, V is volume, n the number of moles, T the temperature, and R is the universal gas constant. In SI units R = 8.3145 J/°K mol

In an ideal gas it can be shown that when C_p and C_v are the molar heat capacities at constant pressure and constant volume,

$$C_p - C_v = R \; . \tag{IV-18}$$

Combining this equation with Eq. (IV-17) and solving for p gives

$$p = \frac{n}{V}(C_p - C_v)T \; , \tag{IV-19}$$

which can be rewritten as

$$p = \frac{1}{V}\left(\frac{C_p}{C_v} - 1\right) nC_vT . \tag{IV-20}$$

Because C_v is the molar specific heat and I is the internal energy per unit mass,

$$nC_vT = mI . \tag{IV-21}$$

By use of this equation, Eq. (IV-20) becomes

$$p = \frac{1}{V}\left(\frac{C_p}{C_v} - 1\right) mI . \tag{IV-22}$$

We now define a constant γ such that

$$\gamma \equiv \frac{C_p}{C_v} .$$

This variable is called the POLYTROPIC GAS CONSTANT. This constant is always greater than one and represents the ratio of specific heats and is a property of the gas being simulated. Some typical values of γ are

air, $\gamma = 1.4$

helium, $\gamma = 1.66$

CO_2, $\gamma = 1.34$

SF_6, $\gamma = 1.08$

γ and ρ are used in Eq. (IV-22) to obtain the Lagrangian finite-difference equation for pressure:

$$p_j^n = (\gamma - 1)\rho_j^n I_j^n . \tag{IV-23}$$

This equation, also known as the POLYTROPIC EQUATION OF STATE, is used to calculate values of p at each time step.

With pressure defined, let us take a closer look at q, called the viscous pressure. This variable accounts for loss of kinetic energy in addition to what is used to compress the gas. It serves as a means by which kinetic energy is dissipated in irreversible processes in the fluid such as the creation of heat through friction. The equation for artificial viscous pressure appears in the following form:

$$q_j^n = q_o \rho_j^n c \left(u_{j-1/2}^n - u_{j+1/2}^n \right) \qquad \text{if positive}$$
$$\text{or if negative} \qquad q_j^n = 0 \, , \tag{IV-24}$$

where q_0 is a constant between 0.1 and 0.25, and c is a characteristic velocity of the system.

This equation will not be derived in this work, but it is important to understand why it appears in this form. The irreversible processes that are modeled through the use of the q equation occur when there is a rapid change in the volume occupied by a gas in a system. In our system this change occurs when there is a large differential between the velocities at the left and right of a given cell. Hence we make q proportional to $u_{j-1/2}^n - u_{j+1/2}^n$, indicating a large amount of kinetic energy dissipated when there is a large velocity differential and a small amount dissipated when there is a lesser difference in velocities. Because dissipated kinetic energy is never returned to the system, this term is said to have a value of zero when its computed value is negative.

In order to be dimensionally correct $u_{j-1/2}^n - u_{j+1/2}^n$ is multiplied by the density of the zone and by c, which is called the characteristic velocity. As c's purpose is simply to make our equation dimensionally correct, we have some leeway in choosing this velocity. Typically it is chosen in one of three ways. The simplest method is to define c as equal to the value of some other major velocity in the simulation. In our simulation this would be the velocity of the piston that compresses the gas in the cylinder. Another way that this velocity can be defined is by using the sound speed of the fluid in question. Namely

$$c = \sqrt{\frac{\gamma p}{\rho}} \, , \tag{IV-25}$$

which can be rewritten using our equation for pressure as

$$c = \sqrt{\gamma(\gamma - 1) I} \, . \tag{IV-26}$$

The third way in which c can be determined is by combining these two approaches by adding the sound speed to a prevalent velocity in the problem. In this case

$$c = \text{piston speed} + \sqrt{\gamma(\gamma - 1)I} \,. \tag{IV-27}$$

This third method is the one used in the model presented in this work.

In this section transport equations for position, velocity, density, internal energy, pressure, and viscous pressure were derived—all the equations necessary to construct a one-dimensional compressible fluid-flow simulation. The equations of fluid flow, particularly those of density (or continuity), internal energy, and velocity (or momentum), are often called the NAVIER-STOKES EQUATIONS. This is a general term that can be used to represent any set of fluid-flow equations in 1, 2, or 3 dimensions. Having derived these equations, we are ready to move our discussion to the construction of the computer code itself; but before we take this step, let us first take a closer look at the artificial viscous pressure. Its relationship with the diffusion equation will help us derive a stability requirement that will be important in this simulation.

C. Viscous Pressure and Diffusion

Let us examine the effect of q on the momentum equation [Eq. (IV-10)]. Substituting our definition of q [Eq. (IV-24)] into the momentum equation, while dropping the p's and the subscripts on ρ, gives us

$$u^{n+1}_{j+1/2} = u^{n}_{j+1/2} + \frac{dt}{M}\left(q_o\, \rho\, c \left(u^{n}_{j-1/2} - u^{n}_{j+1/2}\right) - q_o\, \rho\, c\left(u^{n}_{j+1/2} - u^{n}_{j+3/2}\right)\right) \,, \tag{IV-28}$$

which can be rewritten

$$u^{n+1}_{j+1/2} = u^{n}_{j+1/2} + \frac{q_o\, \rho\, c\, dt}{M}\left(u^{n}_{j-1/2} + u^{n}_{j+3/2} - 2u^{n}_{j+1/2}\right) \,. \tag{IV-29}$$

This equation is in the form of a diffusion equation [Eq. (III-18)] where

$$\sigma = \frac{q_0\, \rho\, c\, dx^2}{M} \,. \tag{IV-30}$$

We can use our definition of ρ

$$\rho = \frac{m}{Adx} = \frac{M}{dx} , \qquad \text{(IV-31)}$$

to rewrite Eq. (IV-30) as

$$\sigma = q_0 \, c \, dx \, . \qquad \text{(IV-32)}$$

Using the diffusional stability condition on σ [Eq. (III-6)],

$$\frac{\sigma \, dt}{dx^2} < \frac{1}{2} , \qquad \text{(III-6)}$$

we find that

$$\frac{q_0 \, c \, dt}{dx} < \frac{1}{2} \, . \qquad \text{(IV-33)}$$

This important stability requirement arises from the parallelism between the effect of q on the momentum equation and the equation of heat diffusion. It is the first of two major stability conditions that are found in this code.

The second condition, the COURANT STABILITY CONDITION, is a stability condition that occurs as a result of a numerical instability that will be discussed in Chapter VI. Its presence in a Lagrangian simulation can be explained by using a simple example.

Consider the Fig. IV-4, where the fluid at the left wall of a zone of length dx is moving to the right with velocity v, while the fluid at the right wall is stationary.

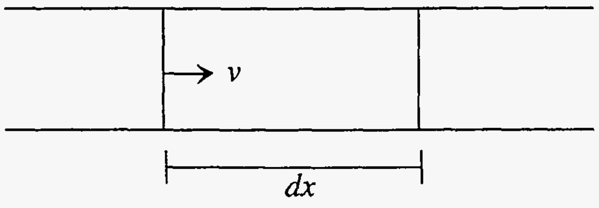


Figure IV-4

In this case, we see that for a given time step (dt), the left wall will move toward the right wall a distance $v \, dt$. In order for our Lagrangian simulation to remain valid, the left wall cannot be allowed to move past the right wall. Mathematically the system must satisfy the following equation:

$$v\,dt < dx \,. \tag{IV-34}$$

Because dx is always positive, this equation can also be written as

$$v\frac{dt}{dx} < 1 \,. \tag{IV-35}$$

Since cross-overs can also occur when a right wall moves leftward past a left wall, our final stability condition is

$$|v|\,\frac{dt}{dx} < 1 \,. \tag{IV-36}$$

In our code, v is equal to the maximum velocity in the system $|u| + c$. Equation (IV-36) is the Courant stability condition. It will be discussed in greater detail later on in this work. In this chapter, we need only note its restrictions when we represent our system of equations computationally.

D. Computational Lagrangian Fluid Flow

Using the equations derived in Section B, we can begin writing our one-dimensional Lagrangian compressible fluid-flow code. As was true in the heat code, we must define our variables in FORTRAN terms before we discuss their use in the code itself. Our variable names appear in code form as follows: $p_j \equiv$ p(j), $q_j \equiv$ q(j), $I_j \equiv$ sie(j), $\rho_j \equiv$ rho(j), $u_{j+1/2} \equiv$ u (j), $x_{j+1/2} \equiv$ x(j), $u_{j-1/2} \equiv$ u(j-1), and $x_{j-1/2} \equiv$ x(j-1). Note that x and u, while still defined at $j + 1/2$ and $j - 1/2$, are written as x(j), u(j), x(j-1), and u(j-1). For both of these variables an array index of j indicates a value at position $j + 1/2$, the wall directly to the right of cell j.

Our program is structured similarly to the heat flow problem illustrated in Fig. II-5. The code exists in five main sections: an initialization routine, a section for time checks and incrementation of counters, the definition of boundary conditions, the updating of variable values, and an output procedure.

First, let us examine our initialization procedure, which defines all the initial values necessary for the problem. Just as in the last simulation, this procedure is used to initialize time counters and define the length of the system being simulated, but this procedure must also set values which were not present in our last case.

It defines the constants:

q0	as occurs in Eq. (IV-24)
gamma	the ratio of specific heats
M	mass/area
ul	the velocity at the left end of the cylinder
ur	the velocity at the right end of the cylinder

Also initial values must be assigned to the constants:

rho0	the initial zone density
sie0	the initial zone internal energy
u0	the initial zone velocity

A loop such as the one that assigned temperatures in the previous problem must be constructed to initialize all the real elements of arrays rho as rho0, sie as sie0, and u as u0, and to compute initial values for x, p, and q using Eqs. (IV-3), (IV-23), and (IV-24), respectively.

After the initialization procedure, the program moves into the same sort of loop as did the heat program: making checks, incrementing time counters, updating boundary conditions, and updating the arrays. The time check portion of the loop is exactly the same as in our last two codes and can be written by repeating what was discussed in Chapter II. The boundary conditions and array assignments are also quite similar to those of our first program but must now be modified to deal with a group of arrays as opposed to a single array of temperatures.

The boundary conditions must be defined for each variable that is referenced at the $j = 0$ or $j = \text{jbar}+1$ positions. We can determine which variables are referenced in these positions by referring to the equations that define our variable values: Eqs. (IV-3), (IV-10), (IV-11), (IV-16), (IV-23), and (IV-24).

From Eq. (IV-11), which appears in code form as

$$\text{rho(j)} = \text{M}/(\text{x(j)} - \text{x(j} - 1)) \ , \tag{IV-37}$$

we see that x(0) will be referenced, indicating the need for the position at the left of the system to be prescribed in our boundary conditions. By similar analysis of Eqs. (IV-16) and (IV-24), we see that there is also a need for values to be determined for $u_{-1/2}$ and $u_{\bar{j}+1/2}$, and for this reason boundary conditions must be assigned to u(0) and u(jbar), representing $u_{-1/2}$ and $u_{\bar{j}+1/2}$, respectively.

An examination of Eq. (IV-10) might lead the reader to believe that boundary conditions are also required for p and q at position jbar+1. This requirement would be true if the wall at the right of the cell were not prescribed, indicating a u(jbar) that is determined independently of p and q. The only three variables that must be modified to establish our boundary conditions are x(0), u(0), and u(jbar).

These variables should be assigned according to the system we wish to represent. For the piston problem, the wall at the right is stationary, indicating that u(jbar) = ur. The velocity at the leftmost cell wall in this problem is equal to the velocity of our simulated piston, u(0) = ul. The position of the leftmost cell wall is equal to its position at the old time step plus the distance that it moves to the right during the new time step, x(0) = x(0) + dt * ul.

In our boundary conditions, we set a value of u(jbar) and not u(jbar+1). This value may seem strange to the student, as it does not make use of a ghost zone but rather modifies a real value in the array. It is allowed because u(jbar) itself exists at a boundary, representing the velocity at the wall directly to the left of zone jbar. In effect, a ghost zone is being modified in which u(jbar) defines the rightmost wall.

With the boundary conditions updated, the code then moves into the updating portion of the program. This is an explicit procedure, which does not use the nested loop structure that was employed in the previous chapter. We are once again dealing with a single loop that assigns new array values based on the values at the previous time step. However, this time we are not dealing with a single array of temperatures but a series of interdependent arrays.

This change presents a problem that was not present in the previous simulation, namely that of updating the values of the variables in an order such that all terms defined at time n in an equation exist at the same time step. This problem is a more complicated version of the one that caused us to create a *Tnew* array in Chapter II. Now we are not only concerned that the terms of a single array exist at the same time step, but that the values of a group of arrays be updated in an order such that each variable is calculated using values from appropriate time steps.

To understand how this order is determined, let us first list our six equations in pseudo-code format. All variables are expressed as they would be in FORTRAN with the exception of the superscripts which are used to remind the reader of the time step at which each of these terms exists. In this form, Eq. (IV-3) becomes

$$\mathrm{x}^{n+1}(\mathrm{j}) = \mathrm{x}^{n}(\mathrm{j}) + \mathrm{u}^{n}(\mathrm{j})\ \mathrm{dt}. \tag{IV-38}$$

Equation (IV-10) becomes

$$\mathrm{u}^{n+1}(\mathrm{j}) = \mathrm{u}^{n}(\mathrm{j}) + (\mathrm{dt/M}) * (\mathrm{p}^{n}(\mathrm{j}) + \mathrm{q}^{n}(\mathrm{j}) - \mathrm{p}^{n}\ (\mathrm{j+1}) - \mathrm{q}^{n}(\mathrm{j+1})). \tag{IV-39}$$

Equation (IV-11), written at the new time step, is

$$\mathrm{rho}^{n+1}\ (\mathrm{j}) = \mathrm{M}\ (\mathrm{x}^{n+1}(\mathrm{j}) - \mathrm{x}^{n+1}(\mathrm{j-1})\ ). \tag{IV-40}$$

Equation (IV-16) becomes

$$\mathrm{sie}^{n+1} = \mathrm{sie}^{n}(\mathrm{j}) + (\mathrm{dt/M}) * (\mathrm{q}^{n}(\mathrm{j}) + \mathrm{p}^{n}(\mathrm{j})) * (\mathrm{u}^{n}(\mathrm{j-1}) - \mathrm{u}^{n}(\mathrm{j})). \tag{IV-41}$$

Equation (IV-23) at the new time step is

$$\mathrm{p}^{n+1}(\mathrm{j}) = (\mathrm{gamma} - 1) * \mathrm{rho}^{n+1}\ (\mathrm{j}) * \mathrm{sie}^{n+1}(\mathrm{j}). \tag{IV-42}$$

Equation (IV-24) also at the new time step is

$q^{n+1}(j) = q0 * rho^{n+1}(j) * c^{n+1} * (u^{n+1}(j-1) - u^{n+1}(j))$

$$\text{if } (q^{n+1}(j) \text{ . lt. } 0)\ q^{n+1}(j) = 0.0, \qquad \text{(IV-43)}$$

where c^{n+1} is computed at cell j as in Eq. (IV-27).

The equations are placed in an order such that each variable exists at an appropriate time step when it is used. For example, both x and u must exist at time step n when Eq. (IV-38) is implemented, so this equation must appear before Eq. (IV-39). By similar argument, Eq. (IV-39), which includes a p term at time n, must appear before Eq. (IV-42), which updates p. Further examination of equations in this manner leaves us with a final order in which these equations must be placed, namely, Eq. (IV-38), Eq. (IV-41), Eq. (IV-39), Eq. (IV-40), Eq. (IV-42), and Eq. (IV-43), where the third and fourth are interchangeable, as well as the fifth and sixth. The variable updating portion of the program is a loop that implements these transport equations in an appropriate order.

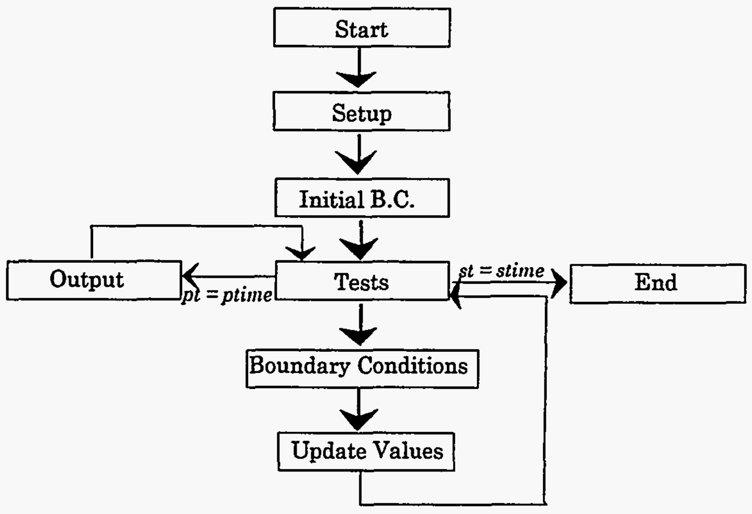


Figure IV-5

With these four sections completed, all that remains is to construct an output procedure desirable to the user, and the one-dimensional Lagrangian fluid code is complete. A graphical representation of this code appears in Figure IV-5.

E. Shocks and Shock Tubes

The following five figures are plots of the density of the fluid in the cylinder as the piston moves in from the left. The parameters chosen for this simulation are: length = 10.(cm), ul = 0.5(cm/s), ur = 0.0(cm/s), gamma = 5/3, jbar=20, rho0 = 1.0 (g/cm^3), sie0 = 0 (cm^2/s^2), q0= 0.3, and dt = 0.05(s). Plots appear at times of 2, 4, 6, 8, and 10 seconds respectively.

Densities at Time 2 (s)

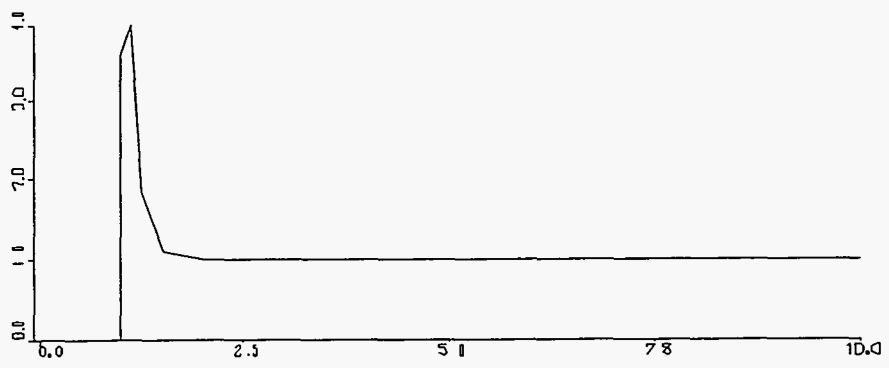


Figure IV-6

Densities at Time 4 (s)

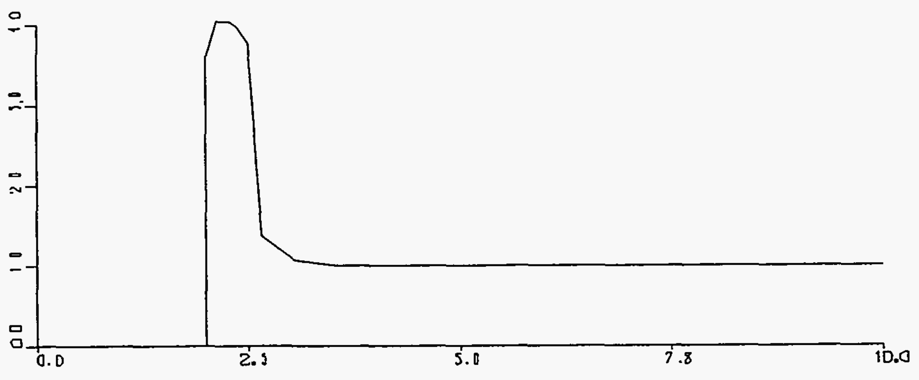


Figure IV-7

Densities at Time 6 (s)

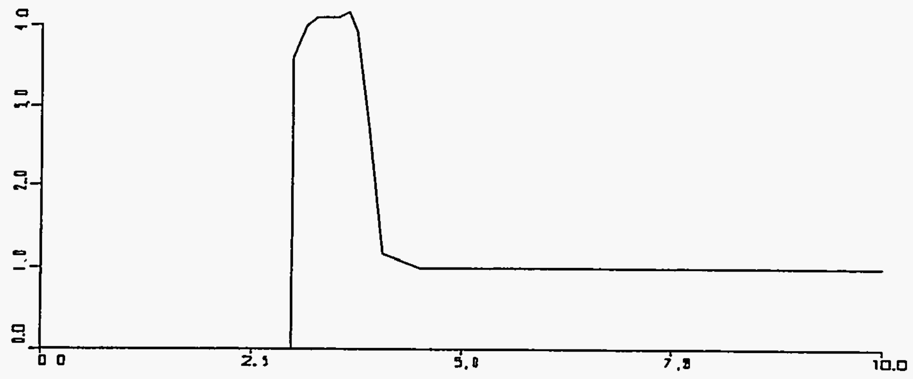


Figure IV-8

Densities at Time 8 (s)

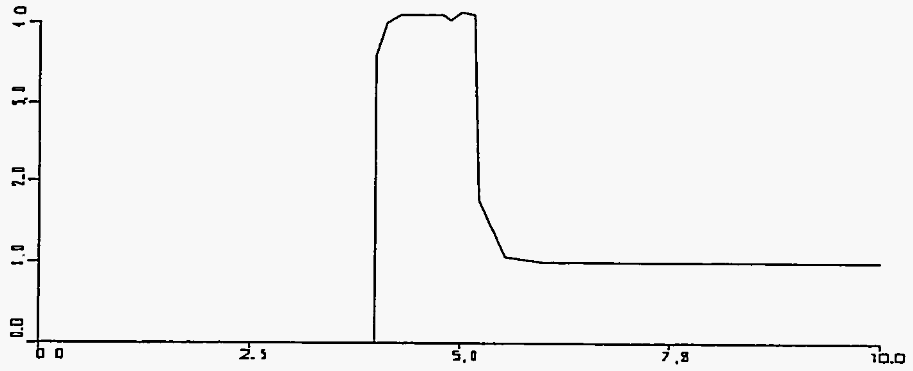


Figure IV-9

Densities at Time 10 (s)

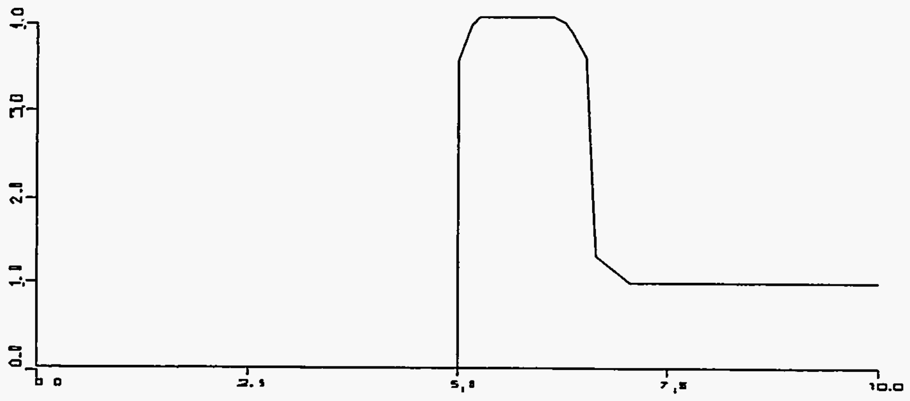


Figure IV-10

The phenomenon that we are examining in these plots is known as a SHOCK, a rapid transition between two states that moves relative to the fluid. Weak shocks occur when fluid is moved at low speeds, but the effects of shocks are most notable when a fluid is moved at a velocity that is near to or greater than the sound speed of the fluid. A shock can be visualized by using the analogy of an evenly spaced line of billiard balls.

Consider the case in which a narrow channel has a piston at one end and is filled with evenly spaced billiard balls, as depicted in Fig. IV-11.

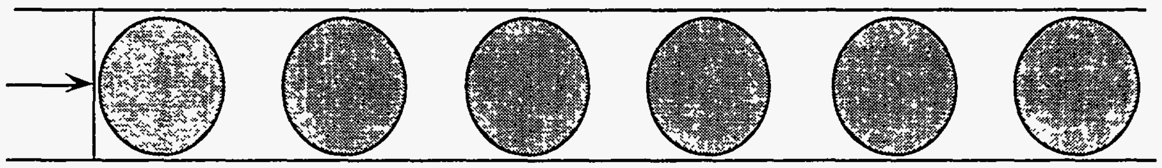


Figure IV-11

The leftmost ball is pushed to the right by the piston and begins to pile up balls in front of it as it moves down the passage. This movement creates a region in which billiard balls exist at a much higher density than in the rest of the passage, because the billiard balls that are moving are touching each other whereas the stationary ones are still evenly spread apart. The front of this compressed region (called the shock or SHOCK FRONT) moves forward faster than the piston itself because billiard balls are constantly piling up in front of the piston as it moves to the right. This system is illustrated in Fig. IV-12. Note that in this figure the transition from the compressed region to the undisturbed surroundings is virtually instantaneous, and that the shock front is not a gradual change in density but rather takes place over a very narrow span.

Our Lagrangian plots demonstrate this sharp contrast between compressed and uncompressed fluid that occurs in a shock. In these plots we can also see the compressed region expanding and the shock front moving at a velocity that is greater than that of the piston. The velocity of the shock front can be predicted, as can other properties,

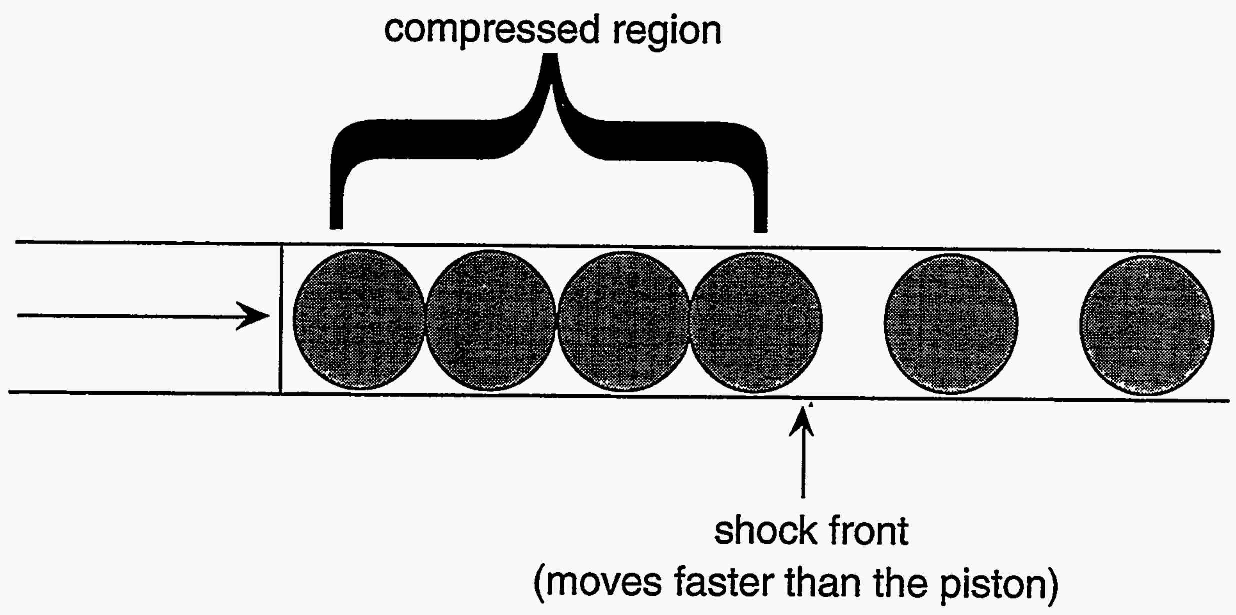


Figure IV-12

by appealing to the equations that describe the theory of shocks. In particular, we will be using the equations of INFINITE STRENGTH SHOCKS, shocks that occur when the shock speed is large compared to the sound speed ahead.

These equations will not be derived in this work, but such derivations are available in various textbooks and monographs, specifically in "Fluid Dynamics—A LASL Monograph" by Francis Harlow and Anthony Amsden, LA-4700. In an infinite strength shock, these equations are

$$u_s = \frac{\gamma+1}{2} u_p \; , \tag{IV-44}$$

where u_s is the velocity of the shock, and u_p is the velocity of the piston, and

$$\rho_s = \frac{\gamma+1}{\gamma-1} \rho_0 \; , \tag{IV-45}$$

where ρ_s the density behind the shock, and ρ_0 is the initial density of the fluid.

Applying these equations to the parameters used in our simulation, we predict that the shock will move forward at a speed of 0.66 (cm/s), and produce a compressed region of density 4 (g/cm^3). These two values can be used to verify the results presented in Figs. IV-6 through IV-10.

The next two graphs illustrate the effect of $q0$ on the accuracy of our numerical simulations. If $q0$ is chosen too low, the answer becomes numerically unstable, as is illustrated in Fig. IV-13.

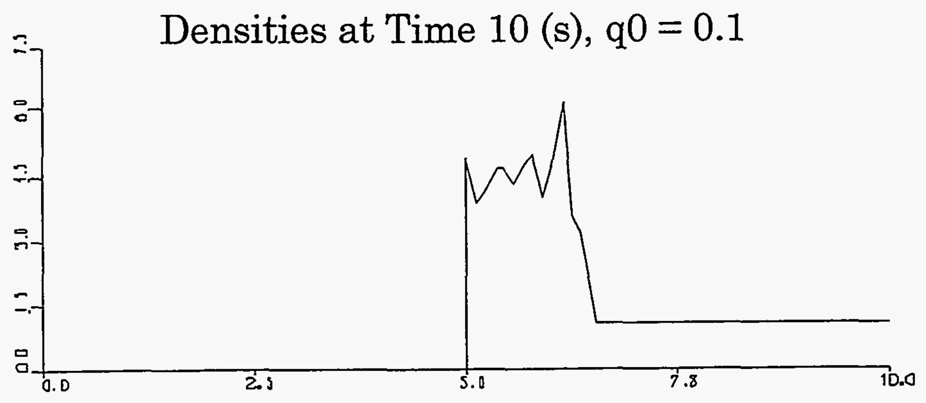


Figure IV-13

If $q0$ is too high, on the other hand, the answer is stable but inaccurate, losing the degree of clarity that was present in the $q0 = 0.3$ graphs.

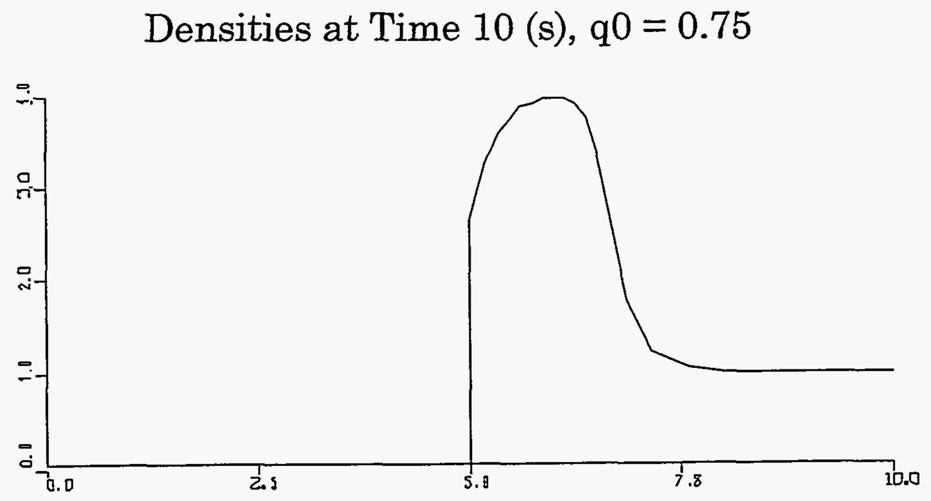


Figure IV-14

At even higher $q0$ values, the diffusional stability condition is violated, resulting in the program being terminated by errors.

A second problem that can be modeled using a one-dimensional Lagrangian code is that of a SHOCK TUBE, a tube that contains two fluids, usually gases, of different densities. Computationally, this problem is set up by setting all velocities to zero and creating an array that is made up of one set of zones at a density ρ_{left} and another set of zones at a different density ρ_{right}. Because our equations assume a constant M, these zones must be MASS MATCHED such that the mass of every zone is a constant. This matching is done by decreasing the initial length of the denser zones relative to the initial length of the less dense zones such that $dx\,\rho$ is a constant.

In the example presented in this work, ρ_{left} is chosen to be 1 (g/cm^3) while ρ_{right} is 4 (g/cm^3). Mass matching is achieved by multiplying the length of the left zones by 8/5 and multiplying the length of the right zones by 2/5 so that $8/5 \times 1 = 8/5 = 2/5 \times 4$. The resulting code appears as the following:

```
      do 100 j = 1,jbar/2
        rho(j) = rho0
        x(j) = x(j-1)+(8./5.)(length/float(jbar))
 100  continue
      do 200 j = (jbar/2)+1,jbar+1
        rho(j) = rho0*4
        x(j) = x(j-1)+(2./5.)*(length/float(jbar))
 200  continue
```

For the results shown in this simulation, the following parameters are used: length = 10.0 (cm), ul = 0.0 (cm/s), ur = 0.0 (cm/s), jbar = 20, rhol = 1.0 (g/cm^3), rhor = 4.0 (g/cm^3), sie0 = 1.0(cm^2/s^2), q0= 0.3, gamma = 5/3, and dt = 0.05 (s). Note that sie0 is not equal to zero in this simulation; there would be no motion of fluids in the shock tube without some initial internal energy being present. Figures IV-15 through IV-20 are graphs of this system at times of 0, 1, 2, 3, 4, and 5 seconds respectively.

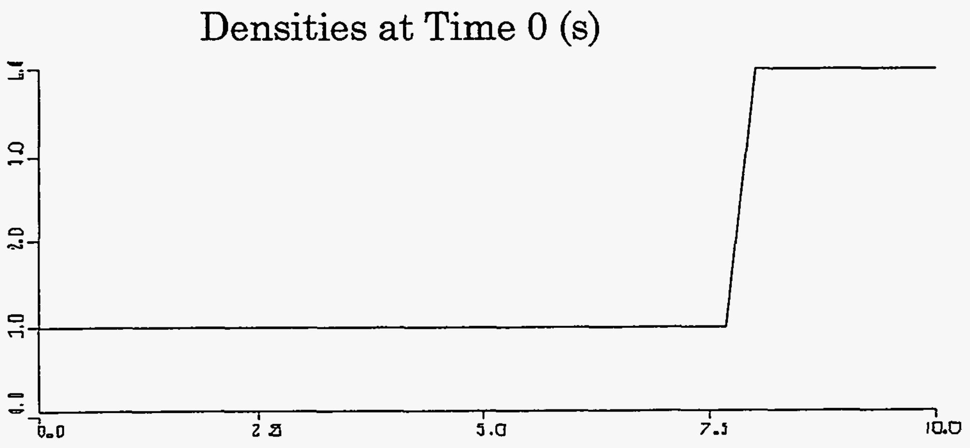


Figure IV-15

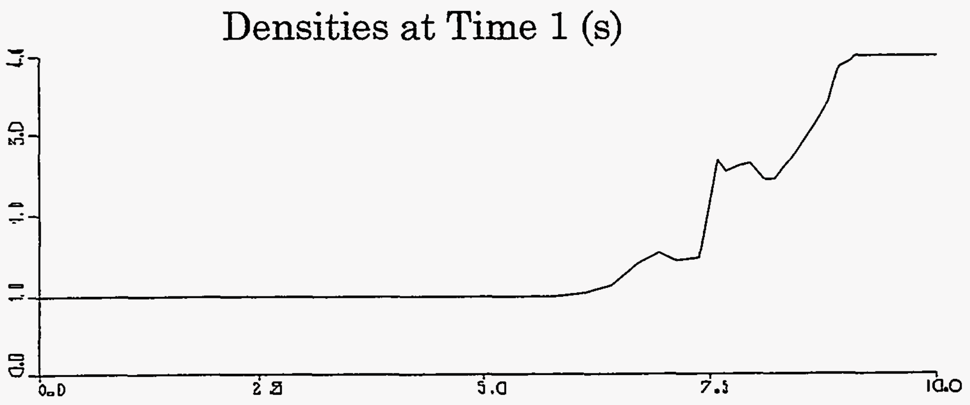


Figure IV-16

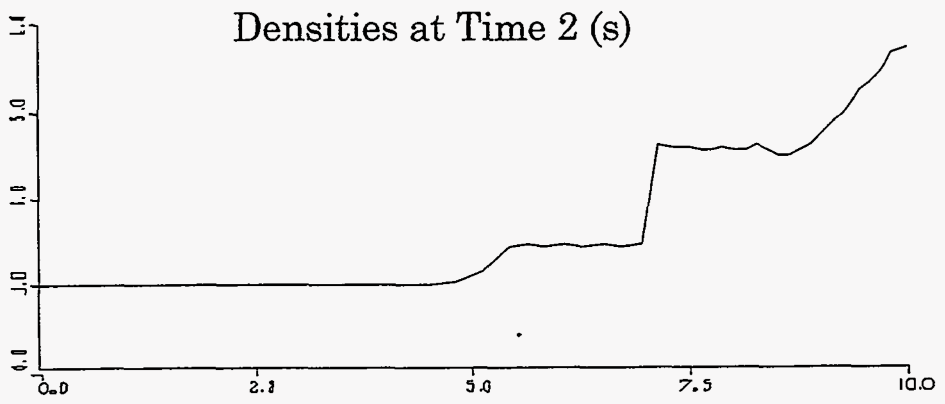


Figure IV-17

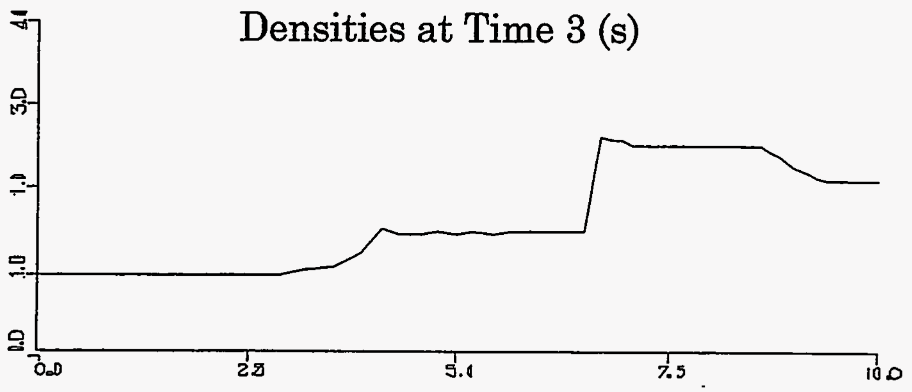


Figure IV-18

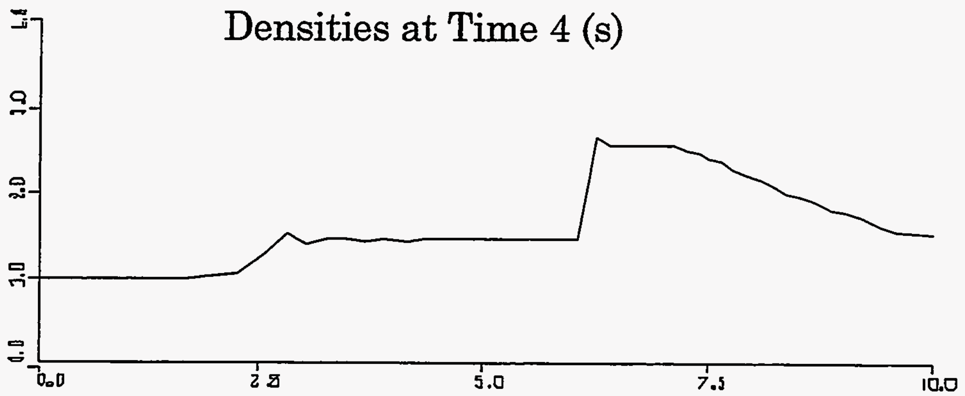


Figure IV-19

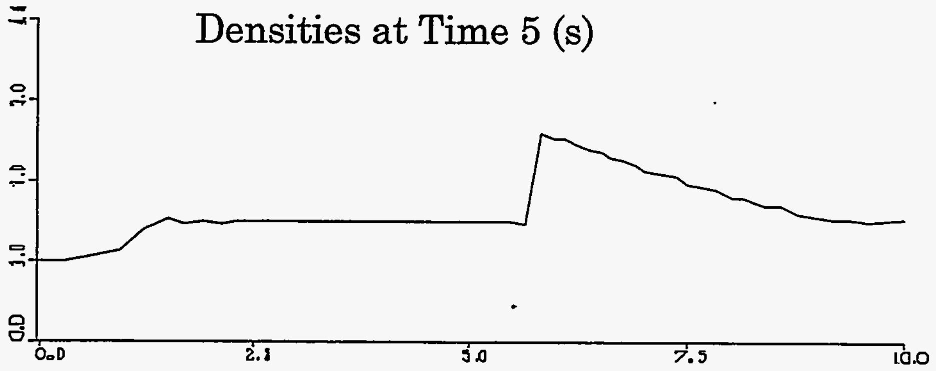


Figure IV-20

In these graphs we see three major features: a shock wave moving to the left, a CONTACT DISCONTINUITY between the two fluids that is also moving to the left, and a RAREFACTION WAVE that is moving to the right and bouncing off of the wall. Each of these elements has been labeled in Fig. IV-21, below.

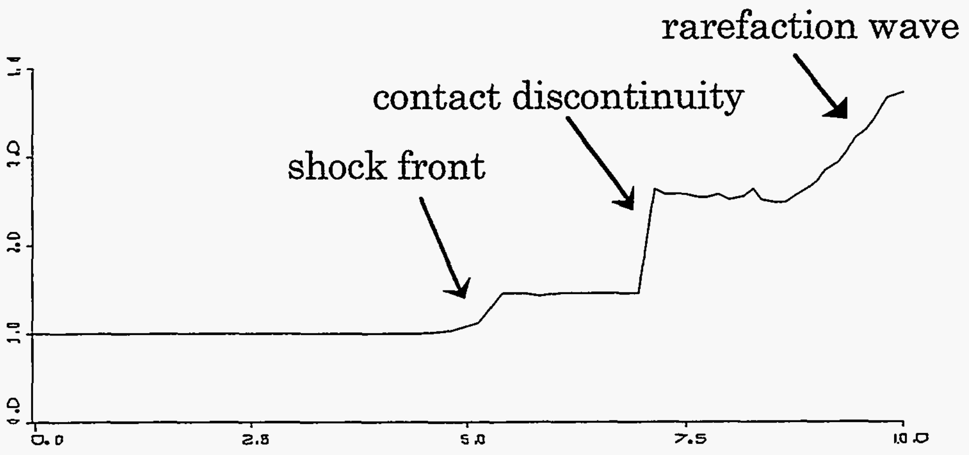


Figure IV-21

The equations that describe the properties of each of these three features of the shock tube problem will not be included in this work. Once again, the reader interested in these equations should refer to LA-4700 or a similar work.

The same sort of instabilities that were present in the piston problem can also be induced in the shock tube problem, as is illustrated by the following plots of density at a time of 2 seconds, each generated by the same parameters as the previous graphs except for $q0$, which is 0.1 in the first graph and 0.75 in the second.

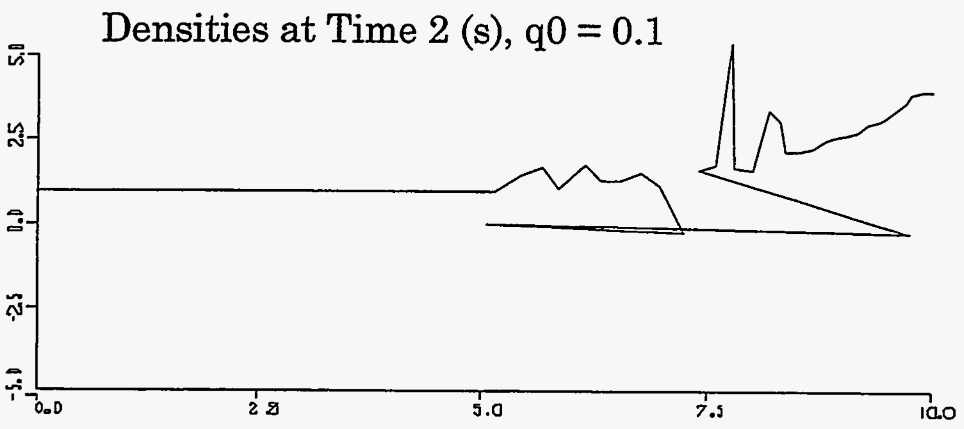


Figure IV-22

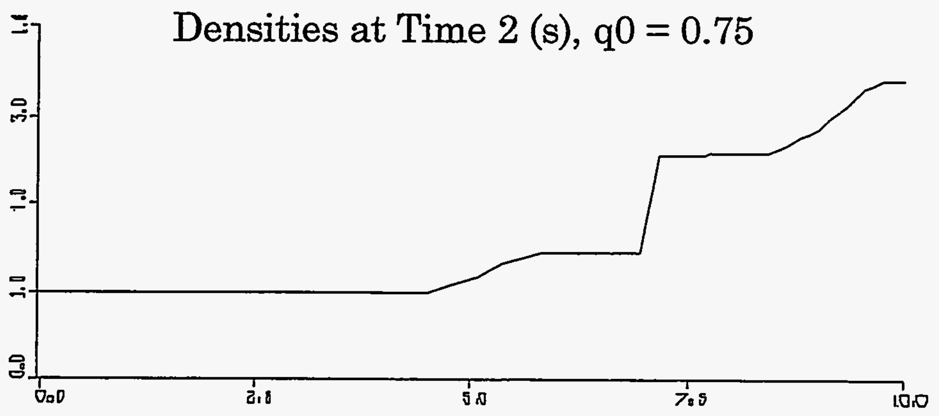


Figure IV-23

Again, if $q0$ is increased to an even higher level, the code will become numerically unstable.

In this chapter we have seen a number of simulations that can be created using the Lagrangian equations for one-dimensional compressible fluid flow. In the following chapter, we will solve the same sorts of problems using an Eulerian method, learning a different technique that can be used to solve the equations of fluid motion computationally.

V. EULERIAN FLUID DYNAMICS

A. Eulerian Methods and Advective Flux

In the previous chapter we examined the use of Lagrangian methods in solving the equations of one-dimensional compressible fluid flow. We are now going to approach the same problem from a different perspective, using an Eulerian technique. In this method the zone positions are held completely fixed, while all quantities are allowed to move between zones. Cell masses are not constant in time, but instead fluid moves between cells; while only the spatial coordinates of the zones remain constant.

This constancy of spatial coordinates is maintained by the calculation of ADVECTIVE FLUXES, fluxes that occur as a result of the motion of fluid from one region to another. An example of this type of flux is the transfer of heat by convection, where heat energy is moved from one region to another by the transfer of the material that contains that energy. The new region is heated not because the material in that region has absorbed the energy from another region, but because a new, hotter material has been moved in to replace the old.

This type of flux is in contrast to the NONADVECTIVE FLUXES that were present in our Lagrangian calculations. Those fluxes occur when the quantities themselves move from one region to another without any motion of material. An example of a nonadvective flux is heat conduction.

While our previous simulation dealt only with nonadvective fluxes, our Eulerian one-dimensional fluid code will include both advective and nonadvective fluxes. In order to accomplish this, we must return to our six equations that describe the interaction of the various physical quantities and add to each a term that describes the advective fluxes that are intrinsic to the Eulerian method.

B. The Equations of Eulerian Fluid Flow

In order to understand the manner that advective flux can be mathematically represented, we must first take a closer look at the situation that it represents. Consider a system such as in Fig. V-1, in which a portion of the material in one zone is being moved into the zone that is adjacent to the right.

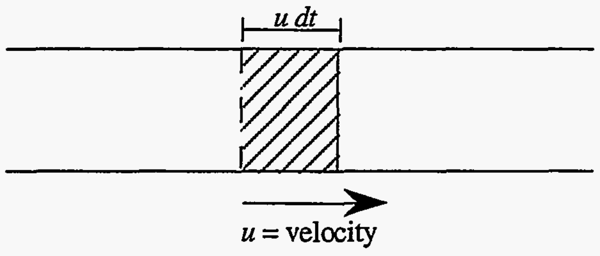


Figure V-1

In this picture we see that when the material in a zone is moving at a velocity u, the material contained in a length $u\,dt$ will be moved into the adjacent cell. Because each zone has an area A, the volume moved from one cell to another is $A\,u\,dt$.

This transfer of volume can be multiplied by ρ, the mass per unit volume, to obtain the following equation for total mass crossing the cell boundary in a given time step:

$$\text{Total Mass Crossing Boundary in a Time Step } (dt) = A\,\rho\,u\,dt \tag{V-1}$$

This equation can be used to find the mass flux, the total mass crossing per unit time per unit area:

$$\text{Mass Flux } = \frac{A\,\rho\,u\,dt}{dt\,A} = \rho\,u \tag{V-2}$$

Equation (V-2) is a statement of the advective mass flux between two cells. It illustrates a much more general principle that can be shown by replacing ρ by a value Q, the density of any quantity that is being advected. In this general case

$$\text{Advective Flux } = Qu\,. \tag{V-3}$$

The density of each of the various physical variables is computed by simply dividing the desired quantity by the volume of a cell. Consider the case of momentum, for example. As was stated in the previous chapter, momentum is mass times velocity:

$$m\,u = \text{momentum}\ . \tag{V-4}$$

Dividing both sides by the volume of a cell, we obtain

$$\frac{m\,u}{\text{volume}} = \frac{\text{momentum}}{\text{volume}}\ . \tag{V-5}$$

Because m/volume is ρ and momentum/volume is the momentum density, this equation can be rewritten:

$$\rho\,u = \text{momentum density}\ . \tag{V-6}$$

By a similar process, we find that

$$\rho\,I = \text{internal energy density}\ , \tag{V-7}$$

and

$$\frac{\rho\,u^2}{2} = \text{kinetic energy density}\ . \tag{V-8}$$

Note that I in Eq. (V-7) is internal energy per unit mass.

Substituting these three density terms into Eq. (V-3), we obtain the following equations of advective flux:

$$\text{Advective flux of mass} = \rho\,u \tag{V-9}$$

$$\text{Advective flux of momentum} = \rho\,u^2 \tag{V-10}$$

$$\text{Advective flux of internal energy} = \rho\,I\,u \tag{V-11}$$

$$\text{Advective flux of kinetic energy} = \frac{\rho u^2}{2}u \tag{V-12}$$

We will use these expressions in deriving the equations of Eulerian fluid flow.

We begin with the expression for density, which was described in our Lagrangian calculations as

$$\rho_j^n = \frac{M}{x_{j+1/2}^n - x_{j-1/2}^n} \, . \tag{IV-11}$$

This expression needs to be modified to reflect the fact that mass is no longer a constant and that distance between cell walls is no longer a variable. To modify this equation, we first substitute dx, the fixed distance between the cell walls, for $x_{j+1/2}^n - x_{j-1/2}^n$:

$$\rho = \frac{M}{dx} \, . \tag{V-13}$$

We must now derive an expression for changes in M, the mass of a cell divided by the area. This derivation is similar to that of the expression for heat in Chapter II. From mass conservation,

$$\text{mass}_j^{n+1} - \text{mass}_j^n = \text{amount in - amount out} \, . \tag{V-14}$$

Because amount in and amount out are simply flux × area × time step, and flux has been defined by Eq. (V-2), numerical expressions for both these terms can be calculated:

$$\text{amount in} = \text{flux}_{\text{left}} \, A \, dt = (\rho u)_{j-1/2} \, A \, dt \tag{V-15}$$

$$\text{amount out} = \text{flux}_{\text{right}} \, A \, dt = (\rho u)_{j+1/2} \, A \, dt \tag{V-16}$$

By substituting these two values into Eq. (V-14) the change in mass, $\text{mass}_j^{n+1} - \text{mass}_j^n$, can be expressed as follows:

$$\Delta \, \text{mass} = (\rho u)_{j-1/2} \, A \, dt - (\rho u)_{j+1/2} \, A \, dt \, . \tag{V-17}$$

Using our definition of M as mass divided by area and factoring out like terms, we obtain an equation for change in M:

$$\Delta M = dt \left((\rho u)_{j-1/2} - (\rho u)_{j+1/2} \right) \, . \tag{V-18}$$

Combining this equation with Eq. (V-13), we find the following:

$$\Delta \rho = \left(\frac{dt}{dx} \right) \left((\rho u)_{j-1/2} - (\rho u)_{j+1/2} \right) \, . \tag{V-19}$$

Because the new density is equal to the old density plus the change in density, $\rho + \Delta\rho$, we are left with a final equation for the updating of densities that is made up of two parts: an expression for the density at the old time step and an expression for the change due to advective flux:

$$\rho_j^{n+1} = \rho_j^n + \left(\frac{dt}{dx}\right) \left((\rho u)_{j-1/2} - (\rho u)_{j+1/2}\right) . \qquad \text{(V-20)}$$

This analysis leaves us with an equation that expresses density at the new time step, but also presents us with a problem. Equation (V-20) makes use of the advected $\rho_{j-1/2}$ and $\rho_{j+1/2}$ densities expressed at the left and right wall of cell j. These quantities cannot be referenced directly but instead must be computed using one of two methods: CENTERED or DONOR CELL.

Centered expressions for advected quantities are computed by averaging the values at the cell centers to the right and left of the wall across which fluid is being advected. In our case, centering would lead to an expression for density in the form of Eq. (V-15):

$$\rho_{j-1/2} = \frac{1}{2}(\rho_j + \rho_{j-1}) . \qquad \text{(V-21)}$$

This value is not acceptable for $\rho_{j-1/2}$, however, because it is UNCONDITIONALLY UNSTABLE, meaning unstable no matter how small we choose our time step. The reason for this instability will be discussed in Chapter VI.

A better method is the donor-cell technique, which uses the upstream value as the value at the advection cell wall. In this technique, the value of a quantity at the cell wall is equal to the value at the left cell center if the flow is from the left or equal to the value at the right cell center if the flow is from the right. This choice of values is mathematically expressed as

$$(u\rho)_{j-1/2} = \rho_{j-1} u_{j-1/2} \quad \text{if} \quad u_{j-1/2} > 0$$

$$\text{or} \qquad \text{(V-22)}$$

$$(u\rho)_{j-1/2} = \rho_j u_{j-1/2} \quad \text{if} \quad u_{j-1/2} < 0$$

and is illustrated visually in Fig. V-2.

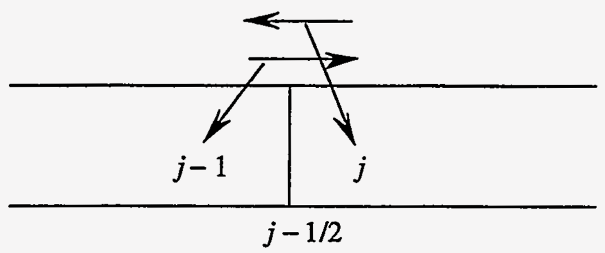


Figure V-2

The donor technique should be employed wherever a quantity is being advected across a cell wall, as is the case with internal energy.

The Eulerian internal energy equation can be calculated beginning with the Lagrangian equation:

$$I_j^{n+1} = I_j^n + \frac{dt}{M}\left(q_j^n + p_j^n\right)\left(u_{j-1/2}^n - u_{j+1/2}^n\right) \tag{IV-16}$$

This transport equation for I is made up of two major terms: the internal energy at the last time step (I_j^n) and the change because of nonadvective flux $\frac{dt}{M}\left(q_j^n + p_j^n\right)$ $\left(u_{j-1/2}^n - u_{j+1/2}^n\right)$. To write this equation in an Eulerian manner, we must add a third term to represent the advective flux. Before this term is added, however, this equation must first be modified. Multiplying by M, we obtain

$$(M\,I)_j^{n+1} = (M\,I)_j^n + dt(p+q)_j^n\left(u_{j-1/2}^n - u_{j+1/2}^n\right) \ . \tag{V-23}$$

This equation represents the total change in $M\,I$ due to the nonadvective pressure terms.

We saw in Eq. (V-20) that the transport equation for a variable whose value is changed only by advective flux appears in the form

$$\left(\frac{Q}{V}\right)_j^{n+1} = \left(\frac{Q}{V}\right)_j^n + (dt/dx)\left((\text{flux of } Q)_{j-1/2} - (\text{flux of} Q)_{j+1/2}\right) \ , \tag{V-24}$$

where Q is any variable property of the cells and V is the volume of a single zone. Using this equation to express change due to advective flux in terms of energy density, we obtain

$$(MI)_j^{n+1} = (MI)_j^n - dt\left((\rho u I)_{j+1/2}^n - (\rho u I)_{j-1/2}^n\right) \ . \tag{V-25}$$

Combining this equation with Eq. (V-23) gives us an expression for change in internal energy that accounts for both advective and nonadvective fluxes:

$$(MI)_j^{n+1} = (MI)_j^n - dt\left[(\rho uI)_{j+1/2} - (\rho uI)_{j-1/2}^n + (p+q)_j^n(u_{j+1/2}^n - u_{j-1/2}^n)\right] \ . \quad \text{(V-26)}$$

Because zones are stationary in an Eulerian simulation, $M = \rho\, dx$. Therefore, this equation can be rewritten as

$$(\rho I)_j^{n+1} = (\rho I)_j^n - \frac{dt}{dx}\left[(\rho uI)_{j+1/2}^n - (\rho uI)_{j-1/2}^n + (p+q)_j^n(u_{j+1/2}^n - u_{j-1/2}^n)\right] \ . \quad \text{(V-27)}$$

This equation is computed using the donor-cell technique for the ρuI terms:

$$(\rho uI)_{j-1/2} = u_{j-1/2}(\rho I)_{j-1} \quad \text{if} \quad u_{j-1/2} > 0$$

$$\text{or} \qquad \text{(V-28)}$$

$$(\rho uI)_{j-1/2} = u_{j-1/2}(\rho I)_j \quad \text{if} \quad u_{j-1/2} < 0 \ .$$

By a process similar to the derivation of Eq. (V-26), Eq. (IV-10) is rewritten as

$$(Mu)_{j+1/2}^{n+1} = (Mu)_{j+1/2}^n - dt\left((p+q)_{j+1}^n - (p+q)_j^n\right) \ . \quad \text{(V-29)}$$

This equation is combined with an advective equation in the form of Eq. (V-25), namely

$$(Mu)_{j+1/2}^{n+1} = (Mu)_{j+1/2}^n + dt\left((\rho u^2)_j^n - (\rho u^2)_{j+1}^n\right) \ , \quad \text{(V-30)}$$

to obtain an equation that accounts for both the advective and nonadvective fluxes that affect momentum:

$$(Mu)_{j+1/2}^{n+1} = (Mu)_{j+1/2}^n - dt\left((\rho u^2)_{j+1}^n - (\rho u^2)_j^n + (p+q)_{j+1}^n - (p+q)_j^n\right) \ . \quad \text{(V-31)}$$

Once again referring to our equation for M given a fixed distance between cells ($M = \rho\, dx$) we obtain

$$(\rho u)_{j+1/2}^{n+1} = (\rho u)_{j+1/2}^n - \frac{dt}{dx}\left((\rho u^2)_{j+1}^n - (\rho u^2)_j^n + (p+q)_{j+1}^n - (p+q)_j^n\right) \ . \quad \text{(V-32)}$$

In this equation, values for u at the cell centers must be computed using the donor-cell technique. These appear in the form

$$(\rho u^2)_j = u_j(\rho u)_{j-1/2} \quad \text{if} \quad u_j > 0$$

or

$$(\rho u^2)_j = u_j(\rho u)_{j+1/2} \quad \text{if} \quad u_j < 0\,, \tag{V-33}$$

where u_j, $\rho_{j-1/2}$, and $\rho_{j+1/2}$ are computed as averages of the values half a cell to the left and half a cell to the right of the point at which these quantities are defined:

$$u_j = \left(\frac{u_{j-1/2} + u_{j+1/2}}{2}\right) \tag{V-34}$$

$$\rho_{j-1/2} = \left(\frac{\rho_{j-1} + \rho_j}{2}\right) \tag{V-35}$$

$$\rho_{j+1/2} = \left(\frac{\rho_j + \rho_{j+1}}{2}\right)\,. \tag{V-36}$$

The student may pose the question of why these averages are used in donor-cell calculations, as they seem to indicate a centered approach that is unconditionally unstable. To explain why these averages are employed, we return to our momentum cell diagram, noting where these various variables are located. In the following figure, the letters in bold indicate quantities at positions where their values are not specified.

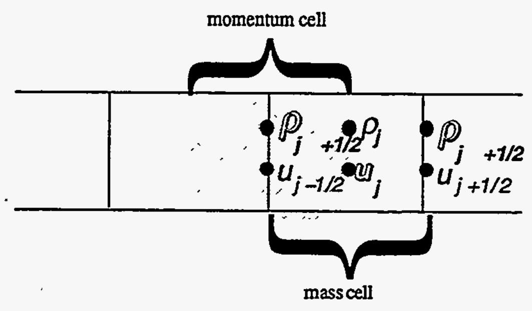


Figure V-3

The terms that employ the donor cell technique are made up of two portions, the donoring velocity and the quantity that is donored. In our original case of density flux, these quantities are $u_{j-1/2}$ and $\rho_{j-1/2}$, respectively. The donoring velocity is always taken at the position at which the flux is taking place, whereas the donored quantity is taken at the center of the cell to the upstream side of the cell wall at which a flux is taking place. For density flux and internal energy flux, all of these values can be taken from positions at which these quantities were directly defined: density and internal energy at the cell centers and velocity at the cell walls. For momentum flux, however, the situation is different.

From Fig. V-3, we see that the donoring velocity at the wall of the momentum cell is u_j, whereas the donored quantities at the center of the momentum cell exist at positions $j - 1/2$ or $j + 1/2$. This configuration forces us to use values that are not directly present in our arrays. These values: u_j, $\rho_{j-1/2}$, and $\rho_{j+1/2}$, are obtained by averaging as in Eqs. (V-34) through (V-36).

We have now derived Eulerian equations for density (ρ), internal energy (I), and momentum (mass $\times\, u$). Values for pressure and viscous pressure (p and q) are determined directly from the values of the other three quantities at each new time step. Thus, the equations for p and q from Chapter IV can be used in our Eulerian simulation. We have completed all the derivation necessary to obtain a set of equations for the simulation of one-dimensional fluid flow in an Eulerian manner and can now begin to implement these equations on the computer. Before we begin this implementation, however, let us first take a look at how our equations appear in partial-differential form and make some observations as to the way that Lagrangian and Eulerian calculations are related.

C. The Partial-Differential Equations of Fluid Flow

Once again, we are going to examine the partial-differential equations that relate to our finite-difference equations. As was the case when we previously examined these equations, this section is not necessary in the writing of our finite-difference code. It is provided only as an additional method of looking at this system.

By a process similar to that used in Section C of Chapter 2, we can rewrite our equations of fluid flow by taking the limits as dx and dt approach zero and generating equations in partial-differential form. In this form Eqs. (V-20) (mass), (V-27) (momentum), and (V-31) (heat energy) appear as follows:

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u}{\partial x} = 0 \tag{V-37}$$

$$\frac{\partial \rho u}{\partial t} + \frac{\partial \rho u^2}{\partial x} = -\frac{\partial (p+q)}{\partial x} \tag{V-38}$$

$$\frac{\partial \rho I}{\partial t} + \frac{\partial \rho u I}{\partial x} = -(p+q)\frac{\partial u}{\partial x} . \tag{V-39}$$

These equations represent the Eulerian form of the transport equations for mass, momentum, and heat energy respectively. They are another form of the Navier-Stokes Equations.

We are going to take a look at these Eulerian equations and relate them to the equations used in the Lagrangian code, trying to gain a better understanding of why these two seemingly dissimilar methods yield computationally similar results.

We will begin with the mass equation, Eq. (V-37). By the chain rule, the second term can be expanded to obtain

$$\frac{\partial \rho}{\partial t} + u\frac{\partial \rho}{\partial x} + \rho\frac{\partial u}{\partial x} = 0 . \tag{V-40}$$

We now employ the mathematical identity for the total differential of a function of two variables, $f(x,t)$:

$$df = \frac{\partial f}{\partial t}dt + \frac{\partial f}{\partial x}dx . \tag{V-41}$$

This equation states that for arbitrarily slight changes in t and x (denoted by dt and dx) the function f changes by an amount df, as given by the formula. Dividing by dt gives us

$$\frac{df}{dt} = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial x}\frac{dx}{dt} . \tag{V-42}$$

In the special case when $\frac{dx}{dt}$ follows the motion of a fluid, as in a Lagrangian calculation, then $\frac{dx}{dt} = u$ and

$$\frac{df}{dt} = \frac{\partial f}{\partial t} + u\frac{\partial f}{\partial x} \, . \tag{V-43}$$

This is an expression for the rate of change of f along the motion of a fluid, also known as the LAGRANGIAN DERIVATIVE. It will be denoted in this work as $\frac{df}{dt}$ as opposed to $\frac{\partial f}{\partial t}$. Elsewhere in the literature, the notation $\frac{Df}{Dt}$ is often used to further emphasize the difference between the partial and Lagrangian derivatives. The meaning, however, is equivalent.

Using the Lagrangian derivative to rewrite Eq. (V-40), we obtain

$$\frac{d\rho}{dt} + \rho\frac{\partial u}{\partial x} = 0 \, . \tag{V-44}$$

This equation is the Lagrangian partial-differential equation for fluid flow; its finite-difference approximation is equivalent Eq. (IV-11). To show this equivalence, we begin with Eq. (V-44) and divide by ρ^2 to obtain

$$\frac{1}{\rho^2}\frac{d\rho}{dt} + \frac{1}{\rho}\frac{\partial u}{\partial x} = 0 \tag{V-45}$$

or

$$-\frac{d\frac{1}{\rho}}{dt} + \frac{1}{\rho}\frac{\partial u}{\partial x} = 0 \, . \tag{V-46}$$

Finite differencing the second term gives us

$$-\frac{d\frac{1}{\rho}}{dt} + \frac{1}{\rho dx}\left(u_{j+1/2} - u_{j-1/2}\right) = 0 \, . \tag{V-47}$$

Note that in this equation dx is no longer part of a partial derivative but a finite distance between zones.

From Eq. (V-13) we have $M = \rho\, dx$, and from Eq. (IV-1) we have $u = dx/dt$, so we can write this equation as

$$-\frac{d\frac{1}{\rho}}{dt} + \frac{1}{M}\left(\frac{dx_{j+1/2}}{dt} - \frac{dx_{j-1/2}}{dt}\right) = 0 \, . \tag{V-48}$$

In this equation, all dt terms are Lagrangian derivatives and can thus be treated in the same manner. We can therefore integrate this equation with respect to dt to obtain

$$\frac{1}{\rho} = \left(\frac{x_{j+1/2} - x_{j-1/2}}{M} \right) . \tag{V-49}$$

This equation is equivalent to our Lagrangian density equation,

$$\rho_j^n = \frac{M}{x_{j+1/2}^n - x_{j-1/2}^n} . \tag{IV-11}$$

We see then, through the use of partial-differential equations, that the Eulerian and the Lagrangian mass equations are equivalent in the properties that they represent.

This equivalence is also true for the momentum equation, which appears in Eulerian form as Eq. (V-38). This equation can be expanded to obtain

$$\rho \frac{\partial u}{\partial t} + u \frac{\partial \rho}{\partial t} + \rho u \frac{\partial u}{\partial x} + u \frac{\partial \rho u}{\partial x} = -\frac{\partial P}{\partial x} , \tag{V-50}$$

where P signifies the total pressure ($P = p + q$).

Returning to the mass equation (V-37), we see that the sum of the second and fourth terms of Eq. (V-50) is equal to zero; this gives us

$$\rho \frac{\partial u}{\partial t} + \rho u \frac{\partial u}{\partial x} = -\frac{\partial P}{\partial x} \tag{V-51}$$

or

$$\rho \left(\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} \right) = -\frac{\partial P}{\partial x} . \tag{V-52}$$

Employing the Lagrangian derivative, we obtain

$$\rho \frac{du}{dt} = -\frac{\partial P}{\partial x} . \tag{V-53}$$

This equation is the Lagrangian partial-differential equation for momentum. Dividing both sides by ρ and finite-differencing it gives us

$$\frac{u_j^{n+1} - u_j^n}{dt} = -\frac{1}{\rho dx} \left(P_{j+1/2}^n - P_{j-1/2}^n \right) , \tag{V-54}$$

which, with a shift of indices, is equal to Eq. (IV-10):

$$u_{j+1/2}^{n+1} = u_{j+1/2}^{n} + \frac{dt}{M}\left(p_j^n + q_j^n - p_{j+1}^n - q_{j+1}^n\right) \ . \tag{IV-10}$$

For internal energy, the process is similar. Eq. (IV-39) is expanded

$$\rho\frac{\partial I}{\partial t} + I\frac{\partial \rho}{dt} + \rho u\frac{\partial I}{\partial x} + I\frac{\partial \rho u}{\partial x} = -\frac{P\partial u}{\partial x} \ , \tag{V-55}$$

the second and fourth terms are dropped using the mass equation

$$\rho\left(\frac{\partial I}{\partial t} + u\frac{\partial I}{\partial x}\right) = -P\frac{\partial u}{\partial x} \ , \tag{V-56}$$

and finally the Lagrangian derivative is used to get the Lagrangian equation for change in internal energy:

$$\rho\frac{dI}{dx} = -P\frac{\partial u}{\partial x} \ . \tag{V-57}$$

Through finite differencing, this equation can be shown to be a partial-differential representation of Eq. (IV-16):

$$I_j^{n+1} = I_j^n + \frac{dt}{M}\left(q_j^n + p_j^n\right)\left(u_{j-1/2}^n - u_{j+1/2}^n\right) \ . \tag{IV-16}$$

So we see that for an Eulerian simulation, our equations appear as

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u}{\partial x} = 0 \ , \tag{V-37}$$

$$\frac{\partial \rho u}{\partial t} + \frac{\partial \rho u^2}{\partial x} + \frac{\partial P}{\partial x} = 0 \ , \tag{V-58}$$

$$\frac{\partial \rho I}{\partial t} + \frac{\partial \rho u I}{\partial x} + \frac{P\partial u}{\partial x} = 0 \ ; \tag{V-59}$$

whereas, in a Lagrangian simulation, our equations are

$$\frac{d\rho}{dt} + \rho\frac{\partial u}{\partial x} = 0 \ , \tag{V-44}$$

$$\rho\frac{du}{dt} + \frac{\partial P}{\partial x} = 0 \ , \tag{V-60}$$

$$\rho\frac{dI}{dt} + P\frac{\partial u}{\partial x} = 0 \ . \tag{V-61}$$

These partial-differential equations provide another way of looking at our one-dimensional fluid-flow equations. They help to explain the Lagrangian and Eulerian finite-difference equations and demonstrate that, although these are seemingly different, their underlying principles are the same.

D. Computational Implementation of Equations

The structure of our Eulerian one-dimensional fluid code is similar to that of the Lagrangian code: it contains the same five sections, its variable declarations are almost the same, and the output procedure is of the same type.

There are, however, some major differences between these two codes. These differences are found in the initialization procedure, in the boundary conditions, and in the equations that are used to update the variable values.

The order in which our variables are given new values is again rho, u, I, p, and q; but rho, u, and I must now be calculated using quantities calculated before the program enters the loop that assigns new values to these arrays. This loop must generate values for rho, u, and I; but the transport equations that were derived in Section B are written in terms of ρ_j, $(\rho u)_{j+1/2}$, and $(\rho I)_j$. We have to obtain array values for the following quantities before calculating the other physical variables:

$$\text{rhon(j)}—\rho_j^{n+1}$$

$$\text{rhoun(j)}—\rho u_{j+1/2}^{n+1}$$

$$\text{rhoin(j)}—\rho I_j^{n+1} \ .$$

Each of these arrays is calculated using the Eulerian equations of transport. The calculations are done for all array values before any updating of rho, u, I, p, or q is done.

Density is computed by simply setting the rho array equal to the rhon array:

$$\text{rho(j)} = \text{rhon(j)} \ .$$

Velocity, u is computed by dividing the density times velocity array by the density array at position j+1/2:

```
u(j) = rhou(j)/(.5 * (rhon(j) + rhon(j+1))) .
```

Internal energy is computed by dividing the internal energy times density array by the density array:

```
sie(j) = rhoin(j)/rhon(j)
```

The p and q equations remain unchanged from the Lagrangian case. This situation leaves us with a loop that assigns values for rho, u, I, p, and q that appears in the following form:

```
      do 300 j =1,jbar
        rho(j) = rhoun(j)
        u(j) = rhoun(j)/(.5*(rhon(j)+rhon(j+1))
        sie(j) = rhoin(j) / rhon(j)
        p(j) = (gamma-1) * rho(j) * sie(j)
        asie = gamma * (gamma-1) * abs(sie(j))
        c = abs(ul) + sqrt(asie)
        q(j) = q0 * rho(j) * c * (u(j-1)-u(j))
        if (q(j).lt.(0.0)) q(j) = 0.0
 300  continue
```

This loop is preceeded by another loop that computes rhon, rhoun, and rhoin arrays using the Eulerian equations for the transport of mass, energy, and momentum:

$$\rho_j^{n+1} = \rho_j^n + \frac{dt}{dx}\left((\rho\, u)_{j-1/2} - (\rho\, u)_{j+1/2}\right) \tag{V-20}$$

$$(\rho I)^{n+1} = (\rho I)_j^n - \frac{dt}{dx}\left[(\rho u I)_{j+1/2}^n - (\rho u I)_{j-1/2}^n + (p+q)_j^n (u_{j+1/2}^n - u_{j-1/2}^n)\right] \tag{V-27}$$

$$(\rho u)_{j+1/2}^{n+1} = (\rho u)_{j+1/2}^n - \frac{dt}{dx}\left((\rho u^2)_{j+1}^n - (\rho u^2)_j^n + (p+q)_{j+1}^n - (p+q)_j^n\right) \tag{V-32}$$

Each of these equations requires the use of the donor-cell technique, meaning that donor-cell values must be computed for

$$\rho_{j-1/2} \quad \text{and} \quad \rho_{j+1/2}$$

$$(\rho u\, I)_{j-1/2} \quad \text{and} \quad (\rho\, u\, I)_{j+1/2}$$

$$(\rho u^2)_j \quad \text{and} \quad (\rho u^2)_{j+1} .$$

The problem of having to write our equations in a manner that allows the computation of donor-cell for each of these terms can be approached in at least two ways: with a series of if/then checks or with a double look-up technique.

The first of these methods involves writing a separate if/then check for each of these six terms. This approach uses six different variables, each with values determined according to the direction of the motion of the fluid, with six separate checks being made for the direction of fluid motion at every loop iteration. This method is viable, but it triples the number of if/then checks, calls for the use of additional scalar variables, and unnecessarily complicates our code.

A much easier technique is to carry out all the flow direction checks before any of the arrays are computed. To do this, we create two arrays of variables: idnr and jdnr. In a loop at the beginning of the variable updating portion of the program, all the elements in these arrays are assigned values of either 0, if the flow is from the left to the right, or 1, if the flow is from the right to the left. idnr represents the motion of fluid at the cell walls ($j + 1/2$), while jdnr represents the flow of the fluid at the cell centers (j). The loop in which they are computed is the following:

```
      do 100 j =1,jbar
         idnr(j) = 0
         jdnr(j) = 0
         if (u(j).lt.0.0) idnr(j)=1
         if ((u(j-1)+u(j)).lt.0.0) jdnr(j)=1
100   continue
```

We can use these integer arrays to determine the positions at which the donor-cell terms are computed. By indexing our variables with j plus an appropriate value of idnr or jdnr, we can rewrite the donor cell terms in the following manner:

$(\rho u)_{j-1/2}$—rho(j-1+idnr(j-1)) * u(j-1)

$(\rho u)_{j+1/2}$—rho(j+idnr(j)) * u(j)

$(\rho u I)_{j-1/2}$—rho(j-1+idnr(j-1)) * u(j-1) * sie(j-1+idnr(j-1))

$(\rho u I)_{j+1/2}$—rho(j+idnr(j)) * u(j) * sie(j+idnr(j))

$(\rho u^2)_j$—(u(j-1)+u(j)) * .5 * u(j-1+jdnr(j-1))

(rho(j-1+jdnr(j-1))+rho(j+jdnr(j))) * .5

$(\rho u^2)_{j+1}$—(u(j)+u(j+1)) * .5 * u(j+jdnr(j)) *

(rho(j+jdnr(j))+rho(j+1+jdnr(j+1))) * .5

The terms on the right of this table are simply computational translations of Eqs. (V-22), (V-28), and (V-33), using a double look-up technique rather than carrying out an if/then statement for each of the equations.

We can compute rhon(j), rhoun(j), and rhoin(j) by constructing a loop that follows the computation of the donor-cell arrays but comes before the computation of the rho-u-sie loop. This loop should appear similar to the following, with the values from the above table being used whenever one of the bold (donor-cell) terms is used.

```
      do 200 j = 1,jbar
c..   density
        rhon(j) = rho(j) + ((dt/dx) * ( $\mathbf{(rho\ u)_{j-1/2}}$ −$\mathbf{(rhou)_{j+1/2}}$))
c..   internal energy
        rhoin(j) = (rho(j) * sie(j)) − ((dt/dx) * ( $\mathbf{(rho\ u\ sie)_{j+1/2}}$−
      &$\mathbf{(rho\ u\ sie)_{j-1/2}}$ + (p(j)+q(j)) * (u(j)−u(j−1))))
c..   momentum
        rhoun(j) = (((rho(j)+rho(j+1)) * .5) * u(j)) − ( (dt/dx) *
      &($\mathbf{(rho\ u^2)_{j+1}}$ − $\mathbf{(rho\ u^2)_j}$) * (p(j+1)+q(j+1)−p(j)−q(j)))
 200  continue
```

To summarize, the variable updating portion of our program consists of first a donor-cell loop; then a loop to compute the mass, momentum, and internal energy densities; and finally a loop that changes the values of the rho, sie, u, p, and q arrays. The return designations of these loops have been numbered in this order.

Returning to the issue of the boundary conditions: An analysis of our equations indicates a need for specified values of rho(0) and sie(0) in addition to u(0) and u(jbar) as in the Lagrangian case. Positions need no longer be updated because they remain fixed throughout the simulation; instead, the conditions must be added that sie at position 0 is equal to a variable siel, and rho at position 0 is equal to a variable rhol. These boundary conditions are of a different nature than those in the heat-transfer problem. There, a constant temperature was maintained at the wall by the recalculation of the value of T(0) at every time step. The equation was

$$T_0 = 2T_L - T_L \, . \tag{II-21}$$

This averaging is not necessary in the present code, because the value used at the wall is computed using the donor-cell technique. If flow is from the left to the right, as is the case with our piston, the values assigned to sie and rho at position j = 0 will effectively exist at the rightmost wall, $j = 1/2$. We are now faced with the question of what values should be assigned to the variables at these positions.

In the Eulerian case, we do not represent the piston itself but rather a shock that is created by the motion of a piston somewhere upstream. We can therefore appeal to the equations for the fluid dynamics of shocks to determine our boundary conditions at the left: $\rho = \frac{\gamma+1}{\gamma-1}\rho_0$, and $I = \frac{u^2}{2}\rho_0$. By substituting our "gamma" and "ul" for the γ's and u's in these equations, we can generate quantities for rhol and siel that will help to maintain a shock wave. Finally, these two variables are added to the initialization procedure, completing our Eulerian code. A graphical representation of this code appears in Figure V-4.

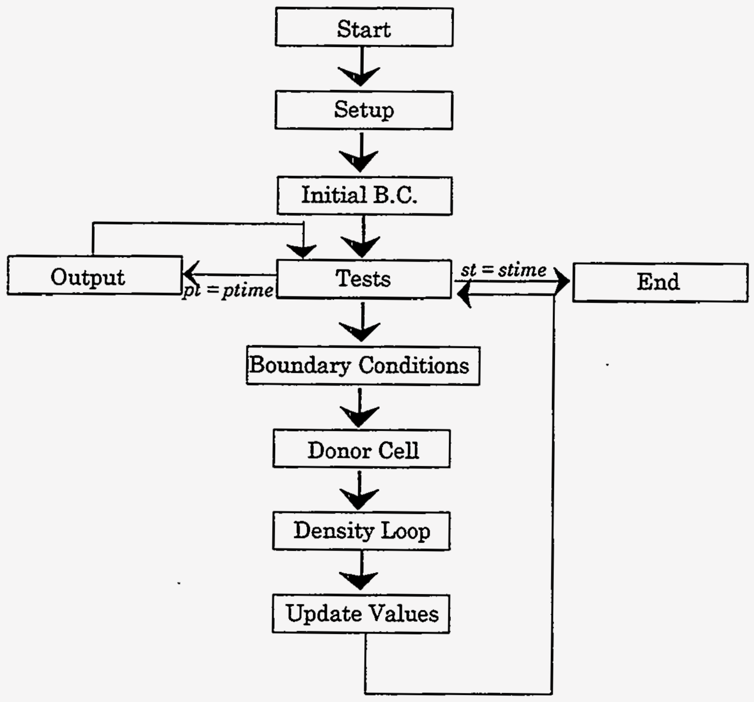


Figure V-4

E. Eulerian Results and Comparison of Eulerian and Lagrangian Simulations

The following five figures are plots of the density of the fluid in the cylinder as the shock moves in from the left. The parameters chosen for this simulation are: length = 10.0 (cm), ul = 0.5 (cm/s), ur = 0.0 (cm/s), jbar=20, rho0 = 1.0 (g/cm^3), sie0 = 0 (cm^2/s^2), q0 = 0.25, gamma = 5/3, and dt = 0.05 (s). Note that these parameters are precisely those used to run the Lagrangian piston problem, with the exception of $q0$, which is lowered from 0.3 to 0.25 for the Eulerian simulation. Plots appear at times of 2, 4, 6, 8, and 10 seconds respectively.

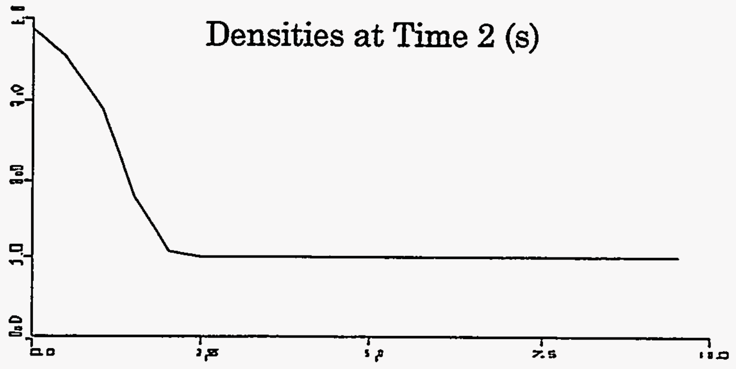


Figure V-5

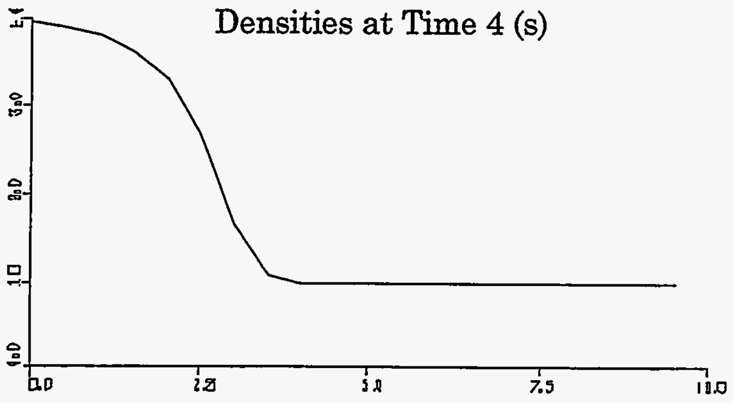


Figure V-6

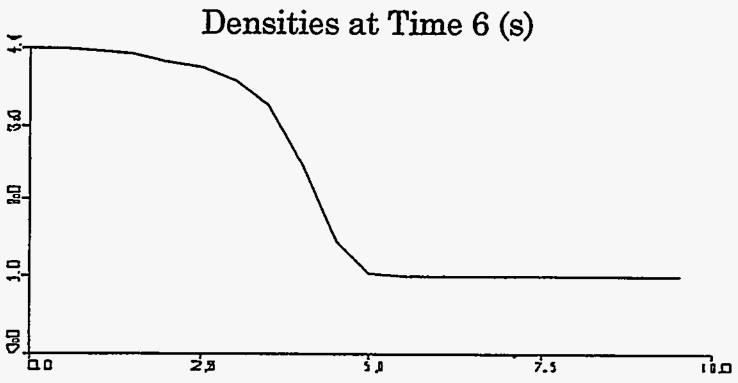


Figure V-7

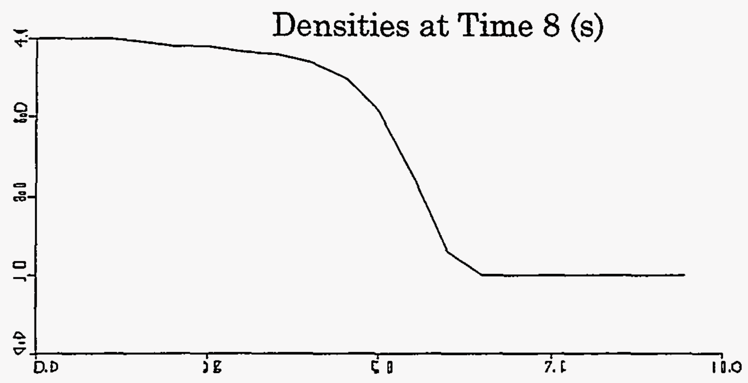


Figure V-8

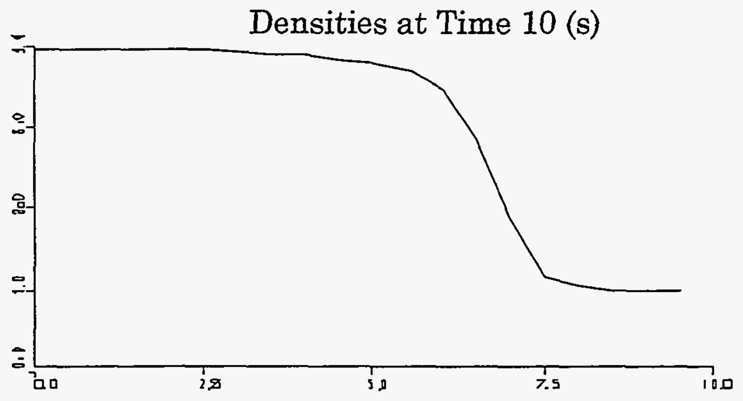


Figure V-9

Note that the Eulerian shock is not as sharp as the Lagrangian shock, even at this lower value of q0. This difference is due to an artificial diffusion that results as an effect of the donor-cell technique. This effect will be discussed in Chapter VI.

Once again applying the equations of shocks [Eqs. (IV-44) and (IV-45)] to the parameters used in our simulation, we predict that our shock will move forward at a speed of 0.66 (cm/s) and produce a compressed region with a density of 4 (g/m^{2}s) . These values verify the results presented in Figs. V-5 through V-9.

The next set of plots demonstrate the results that can be obtained by applying an Eulerian code to the shock tube problem. The following parameters are used: length = 10. (cm), ul = 0.0 (cm/s), ur = 0.0 (cm/s), jbar=20, rhol = 1.0 (g/cm^3), rhor = 4.0

(g/cm^3), sie0 = 1.0 (m^2/s^2), q0= 0.3 and dt = 0.025 (s). Our simulation is set up such that 0.8 of the tube is filled with the less dense fluid and 0.2 is filled with the denser fluid. This set up is necessary to parallel the situation simulated in Chapter IV. Unlike the Lagrangian simulation, however, no mass matching is necessary in the Eulerian case. As was previously stated, masses of zones in an Eulerian simulation are variable; only the positions of zones are constant. Figures V-10 through V-15 are graphs of density within the shock tube system at times of 0, 1, 2, 3, 4, and 5 seconds respectively.

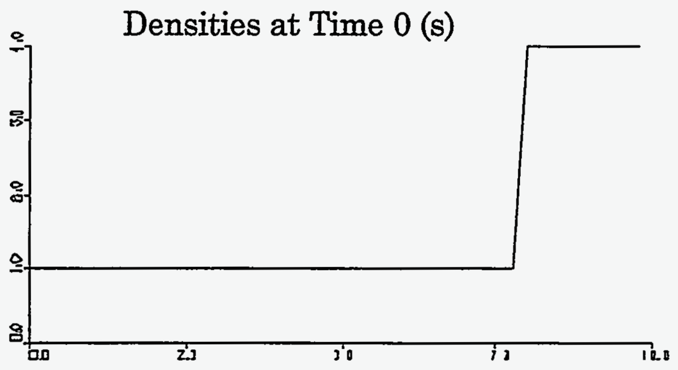


Figure V-10

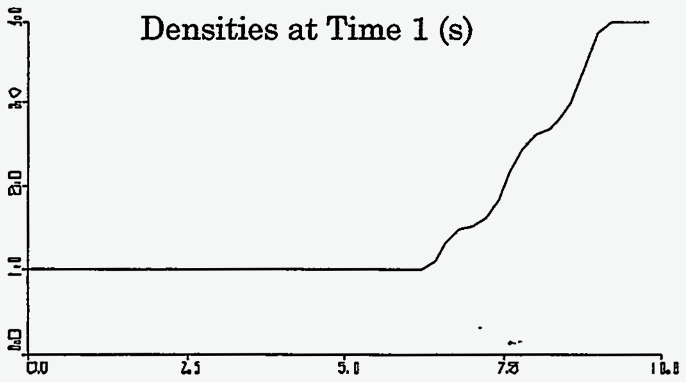


Figure V-11

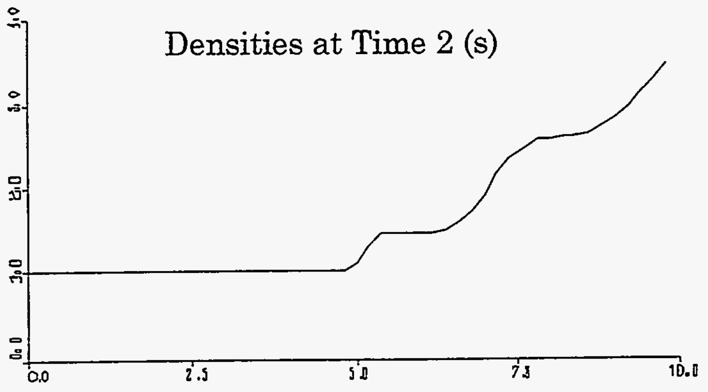


Figure V-12

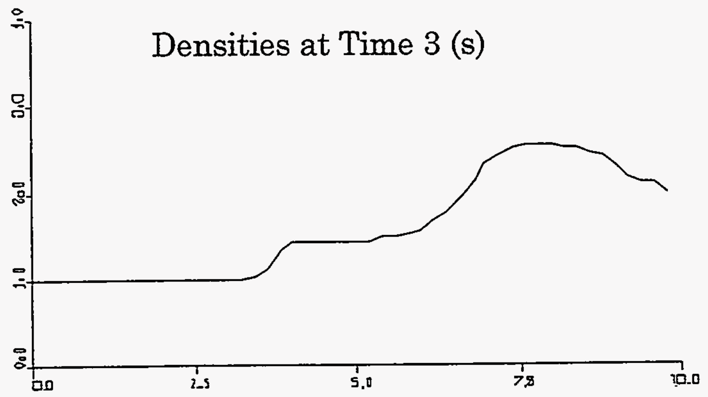


Figure V-13

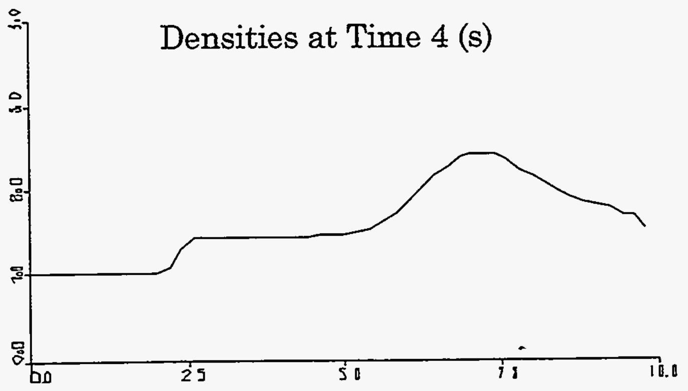


Figure V-14

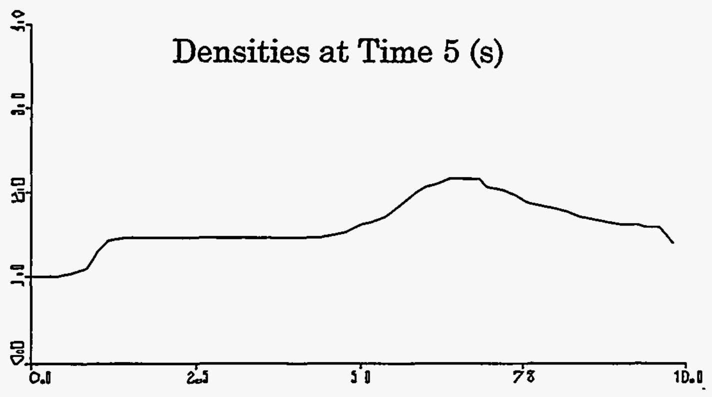


Figure V-15

These graphs contain the same features as the Lagrangian graphs: a shock wave moving to the left, a contact discontinuity between the two fluids that is also moving to the left, and a rarefaction wave that is moving to the right. These features have the same properties as those of the Lagrangian graph and are described by the same set of fluid-dynamics equations.

Lagrangian and Eulerian simulations are also subject to the same stability conditions. At low $q0$ values, the Courant condition is violated, as illustrated in Fig. V-16.

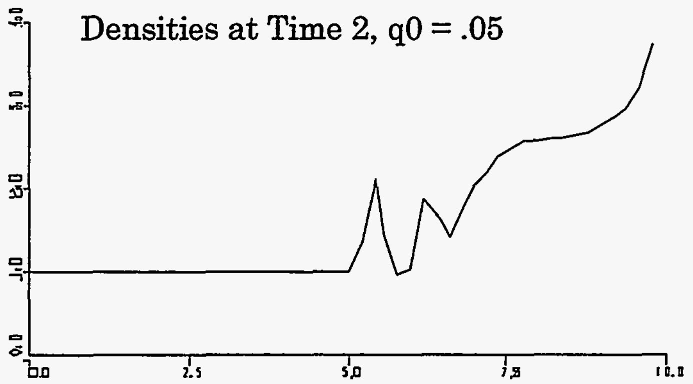


Figure V-16

At high $q0$ values, the features are smeared out:

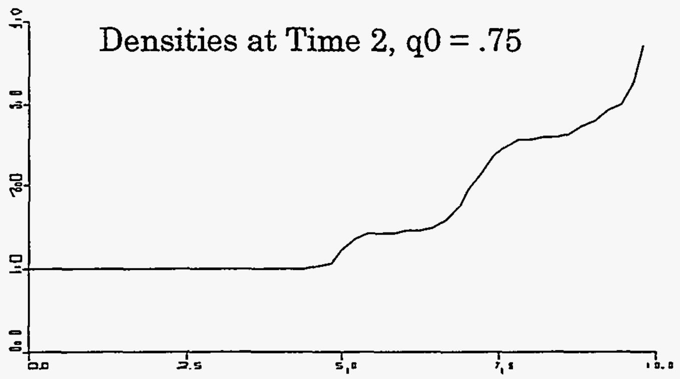


Figure V-17

If $q0$ is raised even higher, the diffusional stability condition is violated and the program terminates.

Although the Lagrangian and Eulerian simulations share the same stability conditions, they are quite different in the sharpness with which they resolve features at a given set of parameters. We see this by comparing graphs of both these simulations at a time step of 2 (s) and a $q0$ of 0.3.

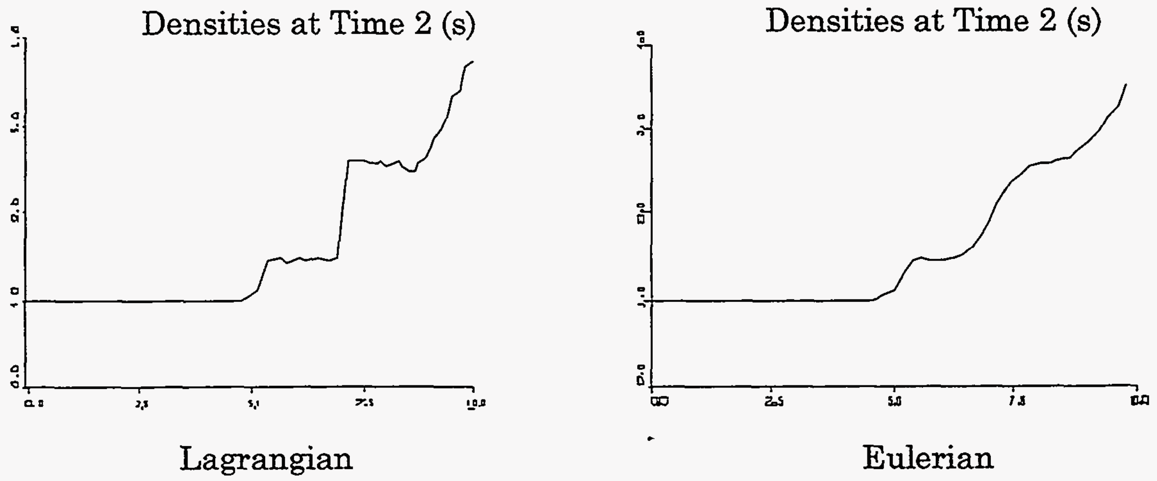


Figure V-18

In this figure, we see that the features of the Lagrangian graph are much sharper than those of the Eulerian simulation. The difference in sharpness is particularly noticeable for the contact discontinuity, which is clear in the Lagrangian simulation but smeared out over several zones in the Eulerian simulation.

The smearing of features in the Eulerian case is a result of the artificial diffusion that is intrinsic to the donor-cell technique. In order to understand why the donor-cell technique causes diffusion in this manner, as well as to understand why the Courant condition is present in an Eulerian simulation, we will have to make use of a method known as truncation error analysis. This method will be discussed in Chapter VI.

VI. TRUNCATION ERROR ANALYSIS AND THE COURANT CONDITION

A. Introduction

This chapter is unique in this book in that it is the only chapter in which we will not write code. We will instead examine a technique known as TRUNCATION ERROR ANALYSIS, which can be used to analyze the error of finite-difference approximations, and we will apply this method to determine a condition that must be met to ensure numerical stability of our fluid-flow model.

This chapter is also unique in that it is almost exclusively based on the manipulation of partial-differential equations. As was the case before, the use of these equations means that this analysis is not essential to the construction of simulations. As was stated in the introduction, mastery of these equations is not a prerequisite to writing finite-difference codes. This chapter does not present the reader with any additional stability conditions or methods of representing finite-difference equations; its purpose is merely to clarify the ones that have already been discussed.

Although not crucial to the writing of our programs, this discussion presents a method of analysis that is important for a person dealing with finite-difference codes. It serves to introduce a new method for examining the validity of our finite-difference approximations and determining the constraints that must be met for our equations to be numerically stable.

We will apply this method to a number of cases, but first let us return to a previously discussed method for determining numerical stability. Using the test solution method introduced in section III-B, we will address the question of the numerical stability of the cell-centered finite-difference wave equation. This discussion will give us a familiar approach to which we can compare our truncation error analysis calculations.

B. Numerical Instability of the Cell-Centered Approach

Consider the general form of the first order wave equation:

$$\frac{\partial T}{\partial t} + a\frac{\partial T}{\partial x} = 0\,. \tag{VI-1}$$

Finite-differencing this equation gives

$$\frac{T_j^{n+1} - T_j^n}{dt} + a\frac{T_{j+1/2}^n - T_{j-1/2}^n}{dx} = 0\,. \tag{VI-2}$$

In the case where a cell-centered flux is used, where

$$T_{j+1/2}^n = \frac{(T_j^n + T_{j+1}^n)}{2}\,, \tag{VI-3}$$

Eq. (VI-2) becomes the cell-centered finite-difference wave equation:

$$\frac{T_j^{n+1} - T_j^n}{dt} + a\frac{T_{j+1}^n - T_{j-1}^n}{2dx} = 0\,. \tag{VI-4}$$

If we now use a test solution for T, choosing

$$T_j^n = Ae^{ikjdx}r^n\,, \tag{VI-5}$$

Eq. (VI-4) becomes

$$Ae^{ikjdx}\frac{\left(r^{n+1} - r^n\right)}{dt} + a\frac{Ar^n\left(e^{ik(j+1)dx} - e^{ik(j-1)dx}\right)}{2dx} \tag{VI-6}$$

or

$$\frac{r-1}{dt} + a\frac{e^{ikdx} - e^{-ikdx}}{2dx} = 0\,. \tag{VI-7}$$

We now define a constant z such that

$$z \equiv \frac{adt}{dx}\,.$$

which allows us to rewrite Eq. (VI-7) as:

$$r = 1 - z\frac{\left(e^{ikdx} - e^{-ikdx}\right)}{2}\,. \tag{VI-8}$$

Using the mathematical identity, $e^{i\theta} = \cos\theta + i\sin\theta$, we obtain

$$r = 1 - iz\sin kdx \ . \tag{VI-9}$$

The magnitude of r is then

$$|r| = \sqrt{1 + 2z^2 \sin^2 kdx} \ . \tag{VI-10}$$

This value is always greater than one, indicating that the solution will always diverge. Hence, the cell-centered wave equation is unconditionally unstable.

C. Truncation Error Analysis

We will now approach the same problem of determining the numerical stability of a wave equation that uses cell-centered differencing with another method of analysis. Instead of using a test case, the numerical stability of our finite-difference equations will be determine by truncation error analysis. In this method, partial differential equations are finite-differenced, and a TAYLOR SERIES EXPANSION is used to determine the accuracy of the finite-difference approximations.

A Taylor series expansion is based on Taylor's theorem which states that for any differentiable function $f(x)$:

$$f(x+dx) = f(x) + \frac{dx}{1!}f'(x) + \frac{dx^2}{2!}f''(x) + \frac{dx^3}{3!}f'''(x) + \dots \ , \tag{VI-11}$$

where f', f'', f''', etc., are the first, second, third, etc., derivatives of the function f.

Because T is simply a function of j and n, our variables can be represented as follows:

$$\begin{aligned} T_j^n &= T(j\,dx, n\,dt) \\ T_j^{n+1} &= T(j\,dx, (n+1)\,dt) \\ T_{j+1}^n &= T\left((j+1)dx, n\,dt\right) \\ T_{j-1}^n &= T\left((j-1)dx, n\,dt\right) \ . \end{aligned}$$

Expanding these values using a Taylor series and substituting the resulting terms into our cell-centered wave equation, Eq. (VI-4), gives us

$$\frac{T_j^n + dt\frac{\partial T}{\partial t} + \frac{dt^2}{2}\frac{\partial^2 T}{\partial t^2} - T_j^n}{dt} + a\left[\frac{\left(T_j^n + dx\frac{\partial T}{\partial x} + \frac{dx^2}{2}\frac{\partial^2 T}{\partial x^2}\right) - \left(T_j^n - dx\frac{\partial T}{\partial x} + \frac{dx^2}{2}\frac{\partial^2 T}{\partial x^2}\right)}{2dx}\right] = 0\,. \qquad \text{(VI-12)}$$

In this equation, all terms of order dx^3 or higher have been ignored. This equation can be reduced to

$$\frac{\partial T}{\partial t} + \frac{dt}{2}\frac{\partial^2 T}{\partial t^2} + a\frac{\partial T}{\partial x} = 0 \qquad \text{(VI-13)}$$

or

$$\frac{\partial T}{\partial t} + a\frac{\partial T}{\partial x} = -\frac{dt}{2}\frac{\partial^2 T}{\partial t^2}\,. \qquad \text{(VI-14)}$$

We will ignore our $-\frac{dt}{2}\frac{\partial^2 T}{dt^2}$ term for a moment, replacing it with an $O(dt)$ to indicate a term of order dt. Our equation becomes

$$\frac{\partial T}{\partial t} = -a\frac{\partial T}{\partial x} + O(dt)\,. \qquad \text{(VI-15)}$$

Differentiating this equation with respect to time gives

$$\frac{\partial^2 T}{\partial t^2} = -a\frac{\partial^2 T}{\partial x \partial t} + O(dt)\,, \qquad \text{(VI-16)}$$

while differentiating with respect to x yields

$$\frac{\partial^2 T}{\partial x \partial t} = -a\frac{\partial^2 T}{\partial x^2} + O(dt)\,. \qquad \text{(VI-17)}$$

We can now use the value for $\frac{\partial^2 T}{\partial x \partial t}$ from Eq. (VI-17) to rewrite Eq. (VI-16):

$$\frac{\partial^2 T}{\partial t^2} = -a\left[-a\frac{\partial^2 T}{\partial x^2} + O(dt)\right] + O(dt) \qquad \text{(VI-18)}$$

or

$$\frac{\partial^2 T}{\partial t^2} = a^2\frac{\partial^2 T}{\partial x^2} + O(dt)\,, \qquad \text{(VI-19)}$$

which can be substituted into Eq. (VI-14) to obtain

$$\frac{\partial T}{\partial t} + a\frac{\partial T}{\partial x} = -\frac{1}{2}a^2 dt \frac{\partial^2 T}{\partial x^2} + O(dt^2) \,. \tag{VI-20}$$

Ignoring the $O(dt^2)$ term, we see that we are left with a wave equation that includes a diffusion term with a negative coefficient of conduction. In other words, if $\sigma = -\frac{1}{2}a^2 dt$ and we ignore our $O(dt^2)$ term and the propagation term $\left(a\frac{\partial T}{\partial x}\right)$, we have

$$\frac{\partial T}{\partial t} = \sigma \frac{\partial^2 T}{\partial x^2} \,. \tag{VI-21}$$

Returning to our discussion of the stability of diffusion equations in Chapter III we can now employ Eq. (III-14), rewritten as

$$r = 1 - 4\sigma \frac{dt}{dx^2} \,. \tag{VI-22}$$

Using the σ from our wave equation, we see that

$$r = 1 + 2a^2 \frac{dt}{dx^2} \,. \tag{VI-23}$$

From this expression we see that r will always be greater than 1, indicating the unconditional instability of the centered approach. The error in this type of finite-difference approximation causes a negative diffusion that causes the system to become numerically unstable.

D. Truncation Error Analysis of the Donor-Cell Technique

Now let us apply this same form of analysis to a wave equation that is differenced using the donor-cell technique. Consider a flow that moves from the left to the right, where

$$a > 0 \,, \tag{VI-24}$$

then

$$T_{j-1/2} = T_{j-1} \tag{VI-25}$$

$$T_{j+1/2} = T_j \,. \tag{VI-26}$$

Using these values of $T_{j-1/2}$ and $T_{j+1/2}$ in Eq. (VI-2) gives

$$\frac{T_j^{n+1} - T_j^n}{dt} + a\frac{T_j^n - T_{j-1}^n}{dx} = 0 \,. \tag{VI-27}$$

Expanding this equation using a Taylor series up to the second order terms yields

$$\frac{T_j^n + dt\frac{\partial T}{\partial t} + \frac{dt^2}{2}\frac{\partial^2 T}{\partial t^2} - T_j^n}{dt} + a\left(\frac{T_j^n - \left(T_j^n - dx\frac{\partial T}{\partial x} + \frac{dx^2}{2}\frac{\partial^2 T}{\partial x^2}\right)}{dx}\right) = 0 \,, \tag{VI-28}$$

which can be reduced to

$$\frac{\partial T}{\partial t} + \frac{dt}{2}\frac{\partial^2 T}{\partial t^2} + a\frac{\partial T}{\partial x} - a\frac{dx}{2}\frac{\partial^2 T}{dx^2} = 0 \,. \tag{VI-29}$$

Analysis similar to that of of Eqs. (VI-15) through (VI-18) gives the expression

$$\frac{\partial^2 T}{\partial t^2} = a^2\frac{d^2 T}{\partial x^2} + O(dx) + O(dt) \,. \tag{VI-30}$$

This value can be substituted into Eq. (VI-29) to obtain

$$\frac{\partial T}{\partial t} + a\frac{\partial T}{\partial x} = \left(-\frac{1}{2}a^2 dt + \frac{1}{2}a dx\right)\frac{d^2 T}{dx^2} + O(dt^2) + O(dxdt) \,. \tag{VI-31}$$

By dropping our $a\frac{\partial T}{\partial x}$, $O(dt^2)$, and $O(dtdx)$ terms, we are once again left with a diffusion equation where

$$\sigma = \left(\frac{1}{2}a\,dx - \frac{1}{2}a^2 dt\right) \,, \tag{VI-32}$$

but this equation assumes that a is greater than zero; a more general σ is

$$\sigma = \left(\frac{1}{2}|a|dx - \frac{1}{2}a^2 dt\right) \,. \tag{VI-33}$$

In order for our equation to remain numerically stable,

$$r = 1 - 4\sigma\frac{dt}{dx} < 1 \,. \tag{VI-34}$$

or

$$-4\sigma\frac{dt}{dx} < 0\,; \tag{VI-35}$$

which, assuming positive dt and dx, becomes

$$\sigma > 0\,. \tag{VI-36}$$

Substituting our σ from Eq. (VI-33) into this equation gives us

$$\sigma = \frac{1}{2}\,|a|\,dx - \frac{1}{2}a^2dt > 0 \tag{VI-37}$$

or

$$1 - \frac{|a|\,dt}{dx} > 0\,. \tag{VI-38}$$

This equation is simply a statement of the Courant condition,

$$\frac{|v|\,dt}{dx} < 1\,. \tag{IV-36}$$

We see then that an Eulerian calculation that uses the donor-cell technique has the same Courant stability condition found in Lagrangian codes.

E. Summary of Numerical Instabilities and Artificial Viscosity

Through the use of truncation error analysis to examine the accuracy of our finite-difference approximation of the wave equation, we have shown that this approximation represents not only the motion of a wave but a form of diffusion. In the cell-centered case, the conduction coefficient of this artificial diffusion is negative, indicating a system that is unconditionally unstable. This coefficient is

$$\sigma = -\frac{1}{2}a^2dt\,. \tag{VI-39}$$

This instability can be avoided by using a donor-cell method, which yields a diffusion coefficient

$$\sigma = -\frac{1}{2}a^2dt + \frac{1}{2}\,|a|\,dx\,. \tag{VI-40}$$

This method remains stable as long as the Courant condition is satisfied. This method is flawed, however; for when we choose a case in which dt is very small compared to dx, σ becomes large, creating a large amount of artificial diffusion in the simulation. While this diffusion does not create a numerical instability, it results in a less accurate simulation, causing sharp boundary layers to become smooth.

In our simulations we will not attempt to avoid this lack of accuracy, but several methods exist that avoid this problem. One of the more useful of these methods is the ARTIFICIAL VISCOSITY technique, in which a cell-centered approach is used with an additional diffusional term added to counteract the negative diffusion intrinsic to the finite-difference approximations. Truncation error analysis of this method gives an equation in the form

$$\frac{\partial T}{\partial t} + a\frac{\partial T}{\partial x} = -\frac{1}{2}a^2 dt\frac{\partial^2 T}{\partial x^2} + \nu_a\frac{\partial^2 T}{\partial x^2}\,. \tag{VI-41}$$

If the coefficient of artificial viscosity ν_a is chosen, such that

$$\nu_a > \left(\frac{1}{2}\right) a^2 dt\,, \tag{VI-42}$$

the numerical stability of the system can be maintained while increasing the accuracy of solutions. This method of improving finite-difference codes is a direct result of truncation error analysis.

Truncation error analysis can be used to determine the validity of finite-difference approximations, indicate the conditions necessary to maintain numerical stability, and describe methods by which the accuracy of solutions can be improved. Although it does not have a major effect on the codes that we are writing in this series of exercises, it helps to explain some of the reasoning that lies behind these programs. A versatile and powerful tool, truncation error analysis is essential to the person who wishes to examine the basic foundations on which finite-difference codes are based.

VII. TWO-DIMENSIONAL INCOMPRESSIBLE FLUID FLOW

A. Calculations in Two-Dimensions

Up to this point in this series of exercises, all problems have been one-dimensional., thus simplifying our simulations in a number of ways: their equations had only to take into account changes in a single direction, their boundary conditions have existed at only two points, and their arrays of variable values have been of a small, one-dimensional sort. For the systems that we have been dealing with up to this point, a one-dimensional approach has allowed us to simplify our problems while still generating results that were accurate.

However, few problems can be represented in a single dimension. Our one-dimensional models assumed both cylindrical symmetry and radial uniformity, two qualities that are rarely found in the same system. Many more systems can be represented by using two-dimensional models. These models can represent any system that has uniformity in a single dimension, including the azimuthal direction used in cylindrically symmetric situations.

The type of two-dimensional code that assumes cylindrical symmetry employs an r-z set of coordinates. In this type of simulation, cells are defined by two numbers, r and z. r represents the distance of a cell center from the axis of the cylinder, and z represents the position of a cell center along that axis. The higher the r, the farther away from the center of the cylinder; the higher the z, the farther down along it.

A second sort of two-dimensional code assumes translational symmetry in one direction. While changes between cells in such a code may take place in two directions, all quantities are assumed to be invariant in the third direction. It is this type of simulation that we will examine in this chapter.

In this sort of code, an i-j set of cell counters are employed, with i representing the cell number in the horizontal direction and j representing cell number in the vertical direction. The resulting two-dimensional array of zones, also known as a MESH, is made up of $i \times j$ individual rectangles of length dx and height dy. This mesh appears in Fig. VII-1.

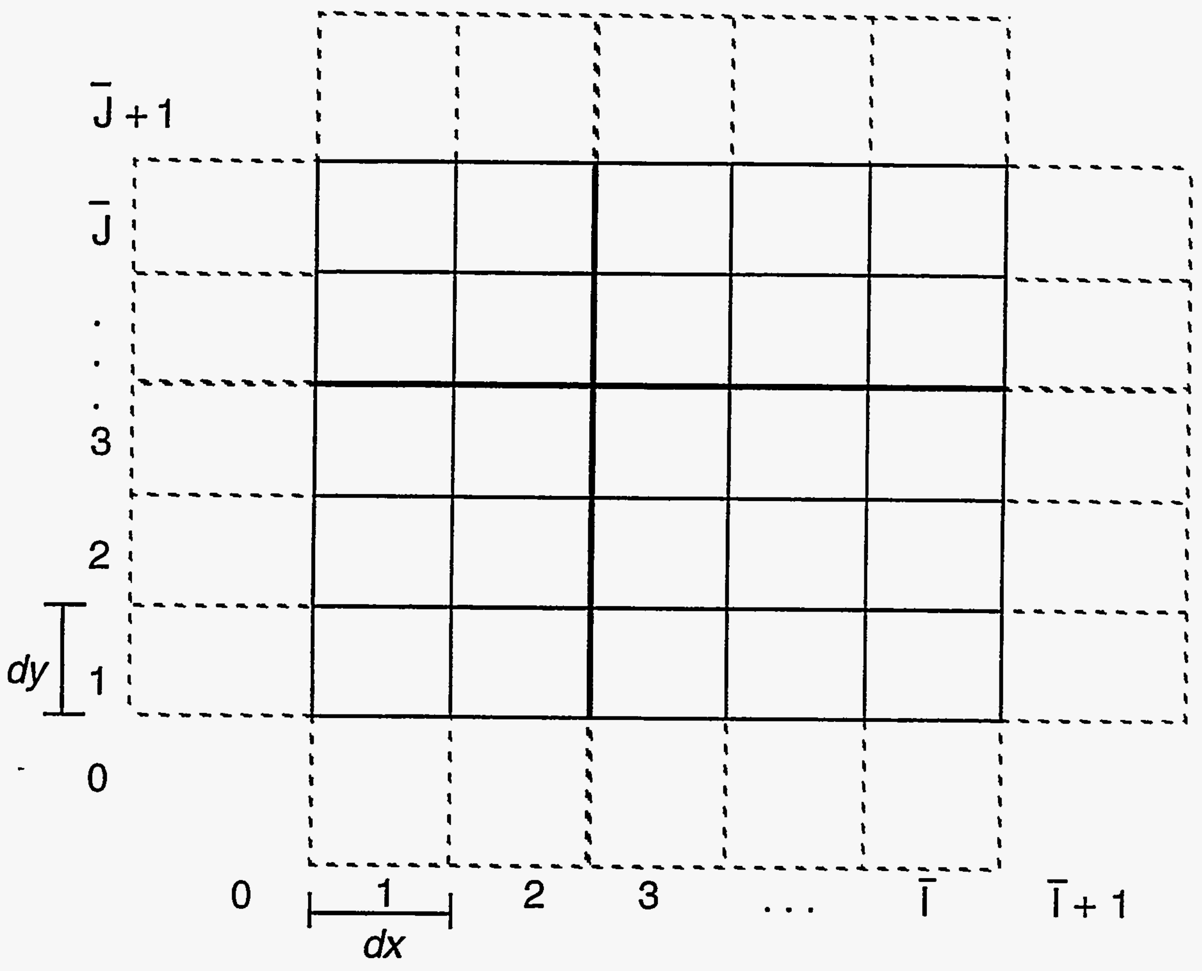


Figure VII-1

Notice that each cell is referenced by two numbers: i and j. The position of cell quantities will now be referenced by two subscripts instead of one: $[\text{Quantity}]_{i,j}$. Notice also that there are now four one-dimensional arrays of fictitious zones: $(i, 0)$, $(i, \bar{j}+1)$, $(0, j)$, and $(\bar{i}+1, j)$. These are needed to represent the two-dimensional boundary conditions.

We will be using this mesh to simulate incompressible fluid flow in two dimensions. Our simulations will be Eulerian, allowing for a mesh of rectangles with fixed positions. Three main variables will be used, as shown in the following table:

$$P^n_{i,j} = p(i,j) = \text{pressure per unit density}$$

$$u^n_{i+1/2,j} = u(i,j) = \text{horizontal velocity}$$

$$v^n_{i,j+1/2} = v(i,j) = \text{vertical velocity}$$

Note that pressure per unit density is represented at the cell centers, horizontal velocity at the right and left walls of the cells, and vertical velocity at the top and bottom cell walls. A mesh in which variables are configured in this manner is known as a STAGGERED MESH. A pictorial representation of a staggered mesh zone appears in Fig. VII-2.

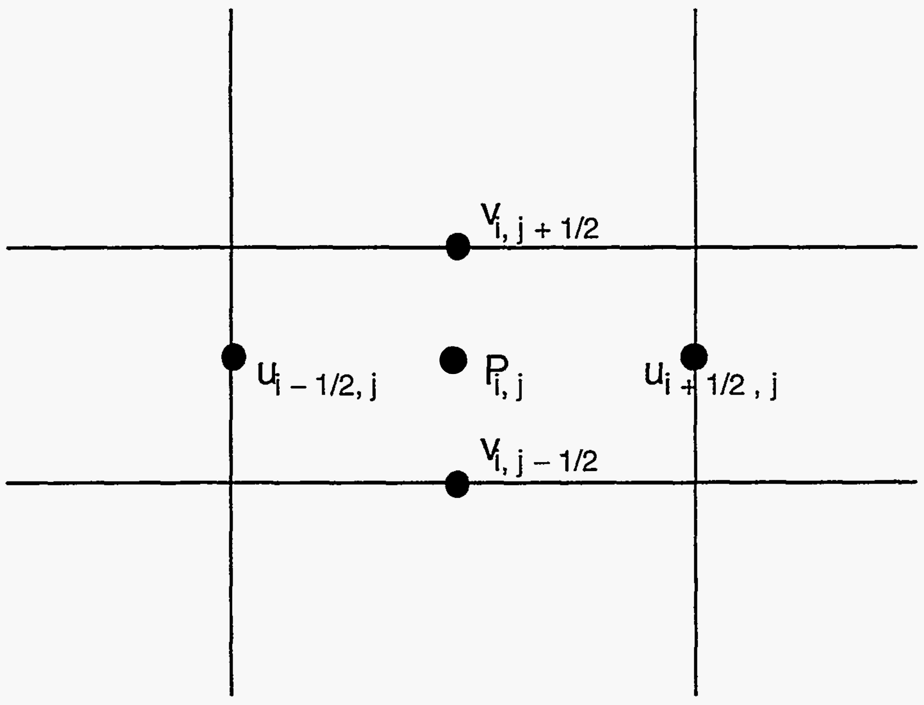


Figure VII-2

Internal energy, density, and artificial (viscous) pressure arrays are not needed since the flow is incompressible; and therefore there are no changes in density, no changes in energy due to pdV work, and no shocks that would require an artificial viscosity. Velocity in two directions and pressure per unit density are the only variables that are needed in the equations that describe two-dimensional incompressible fluid flow.

B. The Equations of Two-Dimensional Incompressible Fluid Flow

To derive the transport equations of two-dimensional incompressible Eulerian fluid flow, we begin with our principles of flux and conservation. Advective flux of mass is again equal to density times velocity:

$$\text{Mass Flux}_{\text{Adv}} = \rho \text{ velocity} . \tag{VII-1}$$

The total amount of mass moving across a boundary in a given time step is equal to the mass flux multiplied by the area of the boundary multiplied by the time step:

$$\Delta \text{ Mass} = \text{Flux}\, A\, dt . \tag{VII-2}$$

Applying this equation along with Eq. (VII-1) to a cell in the mesh gives the following expressions for changes in mass due to flux across the left, right, bottom, and top cell walls:

$$\Delta \text{Mass}_{\text{left}} = \rho u_{i-1/2,j} W\, dy\, dt \tag{VII-3}$$

$$\Delta \text{Mass}_{\text{right}} = -\rho u_{i+1/2,j} W\, dy\, dt \tag{VII-4}$$

$$\Delta \text{Mass}_{\text{bottom}} = \rho v_{i,j-1/2} W\, dx\, dt \tag{VII-5}$$

$$\Delta \text{Mass}_{\text{top}} = -\rho v_{i,j+1/2} W\, dx\, dt . \tag{VII-6}$$

In these equations, ρ is the constant value for the density of the fluid and W is the width of a cell, its thickness in the third dimension. The change in mass is negative in Eqs. (VII-4) and (VII-6) because they represent mass being carried out of the cell by rightward velocities through the right cell wall and upward velocities through the top.

Mass conservation tells us that the total change in mass is equal to the sum of the masses that are fluxed across each of the boundaries:

$$\Delta \text{Mass}_{\text{total}} = \Delta \text{Mass}_{\text{right}} + \Delta \text{Mass}_{\text{left}} + \Delta \text{Mass}_{\text{top}} + \Delta \text{Mass}_{\text{bottom}} . \tag{VII-7}$$

For the incompressible Eulerian case, the total mass of a cell remains constant. Therefore,

$$\Delta\,\mathrm{Mass}_{\mathrm{total}} = 0\ . \tag{VII-8}$$

Combining this equation with Eq. (VII-7) gives

$$\Delta\,\mathrm{Mass}_{\mathrm{right}} + \Delta\,\mathrm{Mass}_{\mathrm{left}} + \Delta\,\mathrm{Mass}_{\mathrm{top}} + \Delta\,\mathrm{Mass}_{\mathrm{bottom}} = 0\ ; \tag{VII-9}$$

or, using Eqs. (VII-3)–(VII-6),

$$\rho W \left[-u_{i+1/2,j}dy + u_{i-1/2,j}dy - v_{i,j+1/2}dx + v_{i,j-1/2}dx\right] = 0\ , \tag{VII-10}$$

which reduces to

$$dy\left(u_{i+1/2,j} - u_{i-1/2,j}\right) + dx\left(v_{i,j+1/2} - v_{i,j-1/2}\right) = 0 \tag{VII-11}$$

or

$$\frac{u_{i+1/2,j} - u_{i-1/2,j}}{dx} + \frac{v_{i,j+1/2} - v_{i,j-1/2}}{dy} = 0\ . \tag{VII-12}$$

Eq. (VII-12) is the two-dimensional finite-difference equation for Eulerian mass flux in an incompressible system, one of the two major equations that will be used in our code. It is closely related to the Eulerian one-dimensional mass flux equation, namely

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u}{\partial x} = 0\ . \tag{V-37}$$

In two dimensions, Eq. (V-37) becomes

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u}{\partial x} + \frac{\partial \rho v}{\partial y} = 0\ , \tag{VII-13}$$

which, assuming a constant ρ, becomes

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0\ , \tag{VII-14}$$

which is a partial differential representation of Eq. (VII-12).

The second major equation that is used to simulate incompressible fluid flow is the momentum equation. In a two-dimensional system, this equation becomes two equations, one representing vertical momentum and one representing horizontal momentum. Each of these equations is solved over a momentum cell that is staggered such that its center exists where a velocity is directly represented. Two such momentum cells appear in Fig. VII-3.

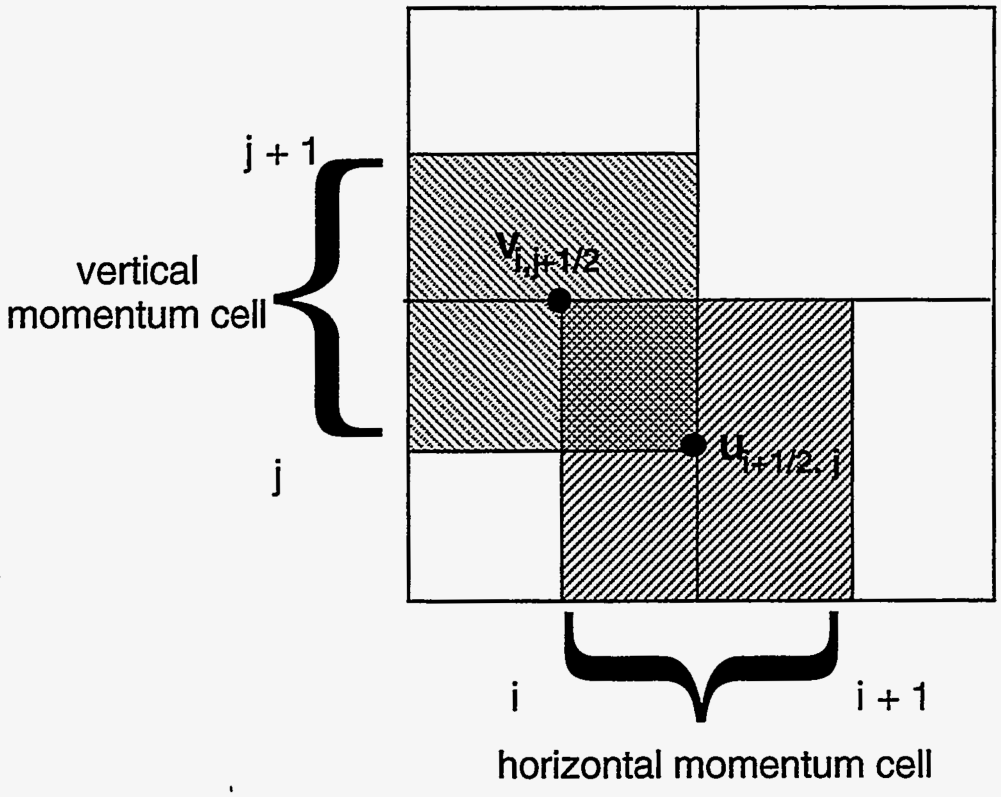


Figure VII-3

Let us consider the case of the horizontal momentum zone, in which momentum is in terms of u. Like the mass equation, the momentum equation that is applied to the two-dimensional cell is very similar to the one-dimensional equation, Eq. (V-38):

$$\frac{\partial \rho u}{\partial t} + \frac{\partial \rho u^2}{\partial x} = -\frac{\partial (p+q)}{\partial x} \, . \tag{V-38}$$

This equation is made up of three terms: the rate of change of momentum $\left(\frac{\partial \rho u}{\partial t}\right)$, an advective term $\left(\frac{\partial \rho u^2}{\partial x}\right)$, and a pressure term $\left(\frac{\partial (p+q)}{\partial x}\right)$. These same three types of terms appear in the two-dimensional equation but with several changes made to the advective and pressure contributions.

Advective flux of momentum in two dimensions occurs in much the same manner as advective flux of mass. There are four surfaces on a momentum cell across which momentum can be carried: the left, right, top, and bottom cell walls. Just as mass flux in Eq. (VII-1) was density times velocity, momentum flux across each of the surfaces in the momentum cell is momentum density (fluid velocity × mass density) multiplied by carrying velocity:

$$\text{Momentum Flux}_{\text{Adv}} = \rho \text{velocity}_{\text{fluid}} \text{velocity}_{\text{carrying}} \,. \qquad \text{(VII-15)}$$

These advective fluxes are illustrated pictorially in Fig. VII-4.

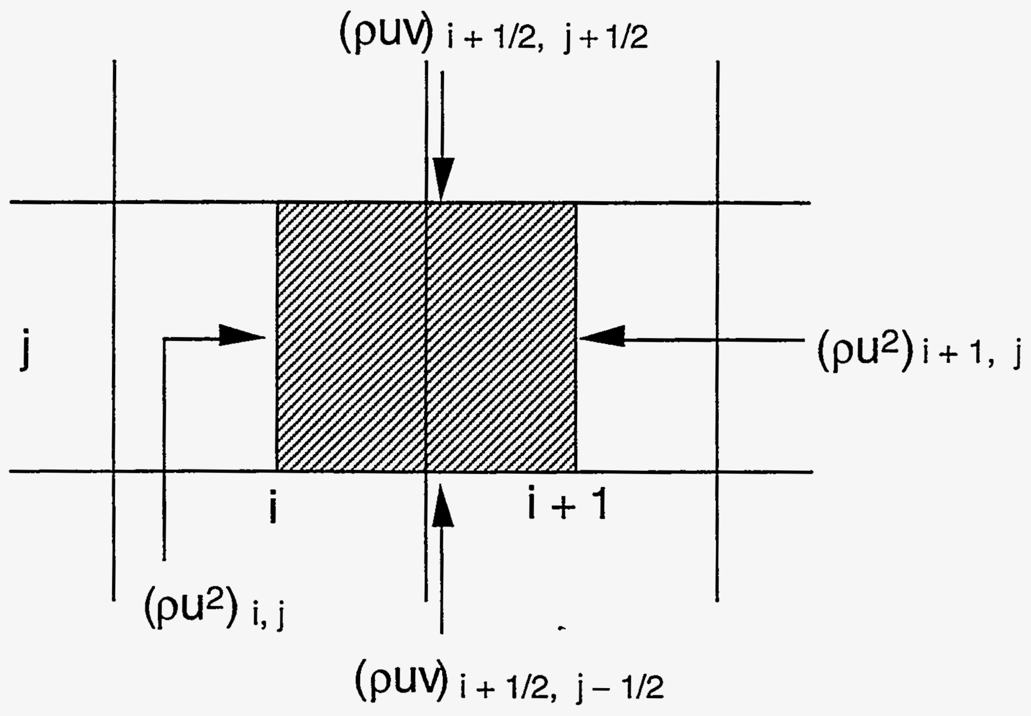


Figure VII-4

As was the case with mass flux, the total amount of momentum moving across a boundary in a given time step due to advective flux is equal to the momentum flux multiplied by the area of the boundary multiplied by the time step:

$$\Delta\,\mathrm{Momentum}_{\mathrm{Adv}} = \mathrm{Flux}_{\mathrm{Adv}} A\, dt\,. \tag{VII-16}$$

Using this equation to determine the change in the momentum of a cell $i + 1/2, j$ due to the fluxes shown in Fig. VII-4, we obtain

$$\Delta\,\mathrm{Momentum}_{\mathrm{Adv_{left}}} = \rho u_{i,j}^2 W\, dy\, dt \tag{VII-17}$$

$$\Delta\,\mathrm{Momentum}_{\mathrm{Adv_{right}}} = -\rho u_{i+1,j}^2 W\, dy\, dt \tag{VII-18}$$

$$\Delta\,\mathrm{Momentum}_{\mathrm{Adv_{bottom}}} = \rho(uv)_{i+1/2,j-1/2} W\, dx\, dt \tag{VII-19}$$

$$\Delta\,\mathrm{Momentum}_{\mathrm{Adv_{top}}} = -\rho(uv)_{i+1/2,j+1/2} W\, dx\, dt\,. \tag{VII-20}$$

These four equations can be combined to form an equation for the total change in momentum due to advection:

$$\begin{aligned}\Delta\,\mathrm{Momentum}_{\mathrm{Adv_{total}}} &= W\, dt\, dy\left((\rho u^2)_{i,j} - (\rho u^2)_{i+1,j}\right)\\ &\quad + W\, dt\, dx\left((\rho uv)_{i+1/2,j-1/2} - (\rho uv)_{i+1/2,j+1/2}\right)\,.\end{aligned} \tag{VII-21}$$

Dividing both sides by the volume of a cell ($W\, dx\, dy$) to generate an expression in terms of momentum density, we obtain

$$\Delta \rho u_{\mathrm{adv}} = dt\left[\frac{(\rho u^2)_{i,j} - (\rho u^2)_{i+1,j}}{dx} + \frac{(\rho uv)_{i+1/2,j-1/2} - (\rho uv)_{i+1/2,j+1/2}}{dy}\right] \tag{VII-22}$$

or

$$\frac{\Delta \rho u_{\mathrm{adv}}}{dt} = -\left[\frac{(\rho u^2)_{i+1,j} - (\rho u^2)_{i,j}}{dx} + \frac{(\rho uv)_{i+1/2,j+1/2} - (\rho uv)_{i+1/2,j-1/2}}{dy}\right]\,, \tag{VII-23}$$

which can be represented in partial differential form as

$$\left(\frac{\partial \rho u}{dt}\right)_{\mathrm{adv}} = -\frac{\partial \rho u^2}{\partial x} - \frac{\partial \rho uv}{\partial y}\,. \tag{VII-24}$$

In the two-dimensional momentum equation for the horizontal direction, the term that is analogous to the advective term in Eq. (V-38) $\left(\frac{\partial \rho u^2}{\partial x}\right)$ is

$$-\left[\frac{(\rho u^2)_{i+1,j} - (\rho u^2)_{i,j}}{dx}\right] - \left[\frac{(\rho uv)_{i+1/2,j+1/2} - (\rho uv)_{i+1/2,j-1/2}}{dy}\right]$$

or

$$-\frac{\partial \rho u^2}{\partial x} - \frac{\partial \rho uv}{\partial y} .$$

The pressure term of Eq. (V-38) is written for two-dimensional incompressible flow by first separating it into two components:

$$\frac{\partial (p+q)}{dx} = \frac{\partial p}{\partial x} + \frac{\partial q}{\partial x} . \qquad \text{(VII-25)}$$

The effect of the real pressure (p) on momentum in the x-direction in two dimensions is the same as in one dimension; this term remains in the same form. The viscous pressure term in two dimensions, however, is rewritten in terms of a true viscosity. This true viscosity parallels artificial viscous pressure in a manner that can be seen by dividing the viscous pressure equation into two terms:

$$\frac{\partial q_x}{\partial x} + \frac{\partial q_y}{\partial y} .$$

These terms can be rewritten using the equation for viscous pressure,

$$q_j^n = q_0\, \rho_j^n\, c \left(u_{j-1/2}^n - u_{j+1/2}^n\right) \qquad \text{if positive}$$

$$\text{or if negative} \qquad q_j^n = 0 \qquad \text{(IV-24)}$$

or

$$q_x = -q_0\, \rho\, c\, dx \left(\frac{u_r - u_l}{dx}\right) , \qquad \text{(VII-26)}$$

which appears in partial-differential form as

$$q_x = -q_0\, \rho\, c\, dx \frac{\partial u}{\partial x} . \qquad \text{(VII-27)}$$

The x-direction term becomes

$$\frac{\partial q_x}{\partial x} = -\frac{\partial}{\partial x}\left(\rho\, q_0\, c\, dx \frac{\partial u}{\partial x}\right) , \tag{VII-28}$$

and the y-direction term becomes

$$\frac{\partial q_y}{\partial y} = -\frac{\partial}{\partial y}\left(\rho\, q_0\, c\, dy \frac{\partial u}{\partial y}\right) . \tag{VII-29}$$

When combined, these terms appear as

$$\frac{\partial q_x}{\partial x} + \frac{\partial q_y}{\partial y} = -\frac{\partial}{\partial x}\left(\rho\, c\, dx \frac{\partial u}{\partial x}\right) - \frac{\partial}{\partial y}\left(\rho\, c\, dx \frac{\partial u}{\partial y}\right) , \tag{VII-30}$$

which can be rewritten in the incompressible case as

$$\frac{\partial q_x}{\partial x} + \frac{\partial q_y}{\partial y} = \rho\left(q_0\, c\, dx \frac{\partial u^2}{\partial x^2} + q_0\, c\, dy \frac{\partial u^2}{\partial y^2}\right) . \tag{VII-31}$$

But the term that is used to represent viscosity in two dimensions is actually

$$-\rho\,\nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) ,$$

where ν is a constant known as the KINEMATIC VISCOSITY. The reason that this constant is used rather than the $q_0 c\, dx$ and $q_0 c\, dy$ values in Eq. (VII-31) is to represent the rate of diffusion of momentum in the fluid in an ISOTROPIC manner, a manner that does not prefer one direction over another.

Our use of a real viscous term of this type is not meant to imply that $dy = dx$ but rather to represent diffusion in a way that is not preferential to any direction. The constant ν in this equation is the simplest manner for representing the physical phenomena of viscosity, which parallels the concept of thermometric conductivity used in our equations of heat transfer. Both these equations appear as a coefficient times a second derivative:

$$\sigma\frac{\partial^2 T}{\partial x^2} \quad \text{and} \quad \rho\nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) ,$$

and both represent the diffusive propagation of a quantity.

There are more complicated methods for simulating viscosity which attempt to preserve the isotropy of the diffusion terms while reconciling the disparity between the q equation and the diffusive terms. These methods are beyond the scope of this work, however, and have never been fully successful. Therefore we will use a constant ν.

Now that terms for advective flux and pressure (now pressure plus diffusion) have been determined, a two-dimensional equation for Eulerian change in momentum in the x-direction can be written

$$\frac{\partial \rho u}{\partial t} - \text{ adv term } + \text{ pressure term} = 0 \qquad \text{(VII-32)}$$

$$\frac{\partial \rho u}{\partial t} + \frac{\partial \rho u^2}{\partial x} + \frac{\partial \rho uv}{\partial y} + \frac{\partial p}{\partial x} - \rho\nu \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) = 0\,. \qquad \text{(VII-33)}$$

Because density is constant in the incompressible case, we divide by ρ to obtain

$$\frac{\partial u}{\partial t} + \frac{\partial u^2}{\partial x} + \frac{\partial uv}{\partial y} = -\frac{\partial P}{\partial x} + \nu \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right)\,, \qquad \text{(VII-34)}$$

where P is equal to p/ρ. Similarly, the equation for change in momentum in the y-direction is

$$\frac{\partial v}{\partial t} + \frac{\partial uv}{\partial x} + \frac{\partial v^2}{\partial y} = -\frac{\partial P}{\partial y} + \nu \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right)\,. \qquad \text{(VII-35)}$$

These two equations, along with the mass equation, Eq. (VII-14) [in finite-difference form Eq. (VII-12)] make up yet another form of the Navier-Stokes Equations and form our mathematical model of two-dimensional Eulerian incompressible fluid flow.

C. Solving Two-Dimensional Fluid-Flow Equations

To solve the equations of two-dimensional incompressible fluid flow, we will make use of a method that combines both explicit and implicit solving techniques. This method is used because of a stability condition that is present in the pressure term in the momentum equation.

This stability condition can be explained by examining the general equation for sound speed. For an ADIABATIC system, meaning a system that contains no processes that either absorb or generate heat, this equation is

$$c^2 = \frac{dp}{d\rho}\,. \tag{VII-36}$$

It can be manipulated to obtain

$$dp = c^2 d\rho\,. \tag{VII-37}$$

The pressure term for the momentum equation $\left(\frac{1}{\rho}\frac{\partial p}{\partial x}\right)$ can then be rewritten as

$$\frac{1}{\rho}\frac{\partial p}{\partial x} = \frac{c^2}{\rho}\frac{\partial \rho}{\partial x}\,. \tag{VII-38}$$

In an incompressible system ρ is a constant, meaning that $\frac{\partial \rho}{\partial x} = 0$. But, pressure is not a constant, indicating that $\frac{\partial p}{\partial x} \neq 0$. We see then that $c^2 \rightarrow \infty$, but the Courant condition states that

$$\frac{(|u| + c)dt}{dx} < 1\,, \tag{VII-39}$$

indicating that our system will be unstable if pressures are computed explicitly.

Our one-dimensional fluid simulations have shown, however, that the advective terms of the momentum equation can be solved explicitly without becoming unstable, and the viscous terms are limited only by the diffusional stability condition:

$$\frac{\nu dt}{dx^2} < \frac{1}{2} \tag{VII-40}$$

$$\frac{\nu dt}{dy^2} < \frac{1}{2}\,. \tag{VII-41}$$

While the pressure contributions in the momentum equations must be computed implicitly, then, the rest of the terms can be computed faster using an explicit method. While it is conceivable that all terms of these equations can be computed using an implicit method, a better technique is to calculate the advective and viscous terms using an explicit solver, then calculate the change in pressures with an implicit solver.

This calculation is done by first grouping together the advective and viscous terms of the momentum equations. Equations (VII-34) and (VII-35) become

$$\frac{\partial u}{\partial t} = \left(-\frac{\partial u^2}{\partial x} - \frac{\partial uv}{dy} + \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right)\right) - \frac{\partial P}{\partial x} \qquad \text{(VII-42)}$$

$$\frac{\partial v}{\partial t} = \left(-\frac{\partial uv}{\partial x} - \frac{\partial v^2}{dy} + \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right)\right) - \frac{\partial P}{\partial y}\,. \qquad \text{(VII-43)}$$

Using the finite-difference approximation of $\frac{\partial u}{\partial t}$ and $\frac{\partial v}{\partial t}$, we obtain

$$u^{n+1}_{i+1/2,j} = u^{n}_{i+1/2,j} + dt\left(-\frac{\partial u^2}{\partial x} - \frac{\partial uv}{\partial y} + \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right)\right) - dt\frac{\partial P}{\partial x} \qquad \text{(VII-44)}$$

$$v^{n+1}_{i,j+1/2} = v^{n}_{i,j+1/2} + dt\left(-\frac{\partial uv}{\partial x} - \frac{\partial v^2}{\partial y} + \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right)\right) - dt\frac{\partial P}{\partial y}\,. \qquad \text{(VII-45)}$$

We now define quantities $\bar{u}$ and $\bar{v}$ such that

$$\bar{u}^{n+1}_{i+1/2,j} \equiv u^{n}_{i+1/2,j} + dt\left(-\frac{\partial u^2}{\partial x} - \frac{\partial uv}{\partial y} + \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right)\right)$$

$$\bar{v}^{n+1}_{i,j+1/2} \equiv v^{n}_{i,j+1/2} + dt\left(-\frac{\partial uv}{\partial x} - \frac{\partial v^2}{\partial y} + \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right)\right)\,.$$

These terms represent the horizontal and vertical velocities at the next time step, barring any contribution made by pressure. They allow us to rewrite our momentum equations as

$$u^{n+1}_{i+1/2,j} = \bar{u}^{n+1}_{i+1/2,j} - dt\frac{\partial P}{\partial x} \qquad \text{(VII-46)}$$

$$v^{n+1}_{i,j+1/2} = \bar{v}^{n+1}_{i,j+1/2} - dt\frac{\partial P}{\partial y}\,. \qquad \text{(VII-47)}$$

In finite-difference form, $\bar{u}$ and $\bar{v}$ appear as

$$\bar{u}^{n+1}_{i+1/2,j} = u^{n}_{i+1/2,j} - dt\Bigg[\left(\frac{u^2_{i+1,j} - u^2_{i,j}}{dx}\right) + \left(\frac{(uv)_{i+1/2,j+1/2} - (uv)_{i+1/2,j-1/2}}{dy}\right)$$
$$- \nu\left(\frac{u_{i+3/2,j} + u_{i-1/2,j} - 2u_{i+1/2,j}}{dx^2}\right) - \nu\left(\frac{u_{i+1/2,j+1} + u_{i+1/2,j-1} - 2u_{i+1/2,j}}{dy^2}\right)\Bigg] \qquad \text{(VII-48)}$$

$$\bar{v}^{n+1}_{i,j+1/2} = v^{n}_{i,j+1/2} + dt\Bigg[\left(\frac{(uv)_{i+1/2,j+1/2} - (uv)_{i-1/2,j+1/2}}{dx}\right) + \left(\frac{v^2_{i,j+1} - v^2_{i,j}}{dy}\right)$$
$$- \nu\left(\frac{v_{i+1,j+1/2} + v_{i-1,j+1/2} - 2v_{i,j+1/2}}{dx^2}\right) - \nu\left(\frac{v_{i,j+3/2} + v_{i,j-1/2} - 2v_{i,j+1/2}}{dy^2}\right)\Bigg]\,. \qquad \text{(VII-49)}$$

In these equations the advective terms can again be calculated using either a donor-cell or cell-centered technique. In this case, a cell-centered approach is allowable because the viscous term prevents numerical instability in a manner described in Chapter VI. This approach results in a code that is able to resolve delicate physical phenomena that occur at low viscosities.

The KARMAN VORTEX STREET, a type of turbulent fluid flow, is one such phenomenon. This type of flow can be examined by simulating a system in which fluid flows in from the top and bottom thirds of the left wall and out through the entire right wall. At low viscosities, this system will form a fluctuating stream called a vortex street. This phenomenon will be discussed in more detail later in this chapter.

Use of the donor-cell technique results in a code that is able to handle more violent phenomena, such as the rushing of fluid over a stationary block. It allows for systems at higher velocities and with more change in velocity to be simulated without becoming numerically unstable but loses much of the precision of the cell-centered technique. Because the Karman vortex street will not evolve with this imprecision, the cell-centered technique is used in the examples in this chapter.

Using the appropriate technique to express the advective terms, two-dimensional arrays of $\bar{u}$ and $\bar{v}$ are computed using an explicit method. The resulting arrays are then used to compute the velocities and pressures implicitly. The equations that are implicitly solved are the mass equation (VII-12) and finite-difference versions of Eqs. (VII-46) and (VII-47):

$$u^{n+1}_{i+1/2,j} = \bar{u}^{n+1}_{i+1/2,j} - dt\left(\frac{P^{n+1}_{i+1,j} - P^{n+1}_{i,j}}{dx}\right) \tag{VII-50}$$

$$v^{n+1}_{i,j+1/2} = \bar{v}^{n+1}_{i,j+1/2} - dt\left(\frac{P^{n+1}_{i,j+1} - P^{n+1}_{i,j}}{dy}\right) . \tag{VII-51}$$

The function that converges to zero in the iterative solution is the mass equation. This convergence is achieved by defining a quantity $D_{i,j}$ such that it is the difference between the momentum equation and its desired value of zero:

$$D_{i,j} \equiv \frac{u_{i+1/2,j} - u_{i-1/2,j}}{dx} + \frac{v_{i,j+1/2} - v_{i,j-1/2}}{dy} .$$

Because $\bar{u}$ and $\bar{v}$ are calculated before any implicit calculations are done, they do not change during the iterations. The vertical and horizontal velocities in our definition of D are therefore functions only of pressures, as shown in Eqs. (VII-46) and (VII-47). Newton's method [Eq. (III-24)] can therefore be applied to the solution of the pressures with $D(P)$ replacing $f(x)$. The resulting equation is

$$P_{i,j}^{\text{new}} = P_{i,j}^{\text{old}} - \frac{D\left(P_{i,j}^{\text{old}}\right)}{\left(\frac{\partial D}{dP}\right)_{i,j}} . \tag{VII-52}$$

$\left(\frac{\partial D}{\partial P}\right)_{i,j}$ is computed using the chain rule, which says that for a function

$$\begin{gathered} y = F\big(a(x),\ b(x)\ c(x)\ \ldots\big) \\ \frac{dy}{dx} = \frac{\partial F}{\partial a}\frac{\partial a}{\partial x} + \frac{\partial F}{\partial b}\frac{\partial b}{\partial x} + \frac{\partial F}{\partial c}\frac{\partial c}{\partial x} \cdots . \end{gathered} \tag{VII-53}$$

In our case this means

$$\begin{aligned} \left(\frac{\partial D}{\partial P}\right)_{i,j} = & \frac{\partial D_{i,j}}{\partial u_{i+1/2,j}}\frac{\partial u_{i+1/2,j}}{\partial P_{i,j}} + \frac{\partial D_{i,j}}{\partial u_{i-1/2,j}}\frac{\partial u_{i-1/2,j}}{\partial P_{i,j}} \\ & + \frac{\partial D_{i,j}}{\partial v_{i,j+1/2}}\frac{\partial v_{i,j+1/2}}{\partial P_{i,j}} + \frac{\partial D_{i,j}}{\partial v_{i,j-1/2}}\frac{\partial v_{i,j-1/2}}{\partial P_{i,j}} . \end{aligned} \tag{VII-54}$$

These partial-differential terms can be rewritten using Eqs. (VII-50) and (VII-51) and our definition of D:

$$\begin{aligned} \left(\frac{\partial D}{\partial P}\right)_{ij} = & \frac{1}{dx}\left(\frac{dt}{dx}\right) + \left(-\frac{1}{dx}\right)\left(-\frac{dt}{dx}\right) + \\ & \frac{1}{dy}\left(\frac{dt}{dy}\right) + \left(-\frac{1}{dy}\right)\left(-\frac{dt}{dy}\right) , \end{aligned} \tag{VII-55}$$

which reduces to

$$\left(\frac{\partial D}{\partial P}\right)_{i,j} = 2dt\left(\frac{1}{dx^2} + \frac{1}{dy^2}\right) . \tag{VII-56}$$

Consequently, our equation for the implicit updating of pressures is

$$P_{i,j}^{\text{new}} = P_{i,j}^{\text{old}} - \beta D_{i,j} \,, \tag{VII-57}$$

where

$$\beta = \frac{1}{2dt\left(\frac{1}{dx^2} + \frac{1}{dy^2}\right)} \,. \tag{VII-58}$$

Through the use of this equation along with the $\bar{u}$ and $\bar{v}$ equations [Eqs. (VII-48) and (VII-49)], two-dimensional incompressible Eulerian fluid flow can be accurately represented. These equations make up a method that uses both explicit and implicit solving techniques to simulate the motion of an incompressible fluid computationally.

D. Computational Implementation of Equations

Our program is once again structured in five major sections: setup, checks and incrementations, boundary conditions, variable updating, and output; but the interaction of these sections is slightly different from that of previous codes. Our two-dimensional code is configured as in Fig. VII-5.

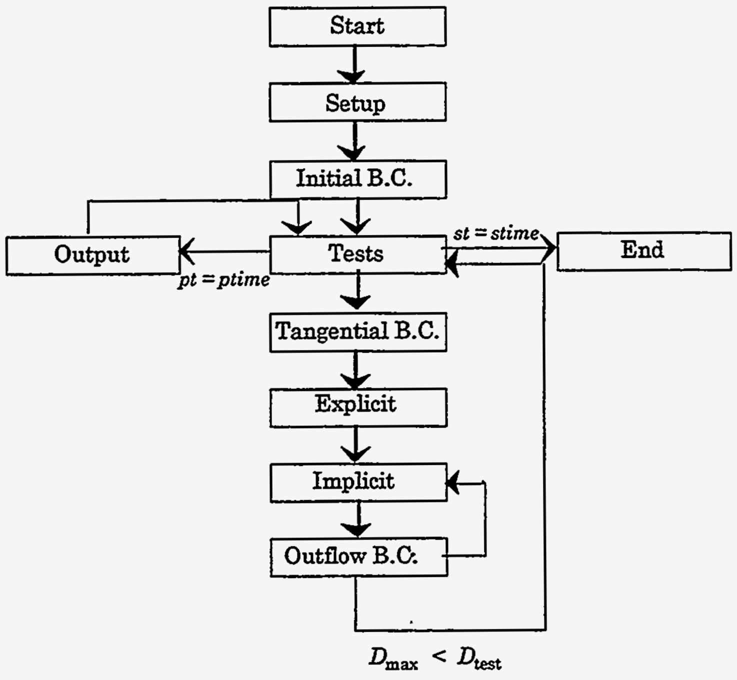


Figure VII-5

Notice that in this figure, three different types of boundary conditions appear, along with an explicit/implicit solver. Each of these will be discussed, but first let us examine the setup routine.

This procedure includes the same time variables present in our previous finite-difference codes (dt, pt, etc.) and the following additional variables:

ibar and jbar – – the number of interior zones in the x- and y-directions

xlen and ylen – – lengths in the x and y-directions

dx and dy – – – the horizontal and vertical lengths of a zone

anu – – – – – – – – the kinematic viscosity

P, u and v – – – – two-dimensional arrays of pressures and velocities

P0, u0, and v0 – – the intial values of the P, u, and v arrays

beta – – – – – – – – $\beta = \frac{1}{2dt\left(\frac{1}{dx^2}+\frac{1}{dy^2}\right)}$

ur, ul – – – – – – – the fluid velocities normal to the right and left walls

vt, vb – – – – – – – the fluid velocities normal to the top and bottom walls

Dtest – – – – – – – the accuracy to which $D_{i,j}$ should converge

In certain cases, output may be required at regular intervals beginning after a certain time. (Every 10 seconds after 100 seconds have elapsed, for example.) To do this, an optional variable btime can be used to represent the time that the program should begin to call the output procedure. The check for output would then become an "and" statement that checks both for pt ¿ ptime and st ¿ btime.

The setup procedure assigns all read in constants and calculate dx, dy, and beta. Velocities and pressures should be set to their initial values. Dtest has the dimensions of s^{-1}, a velocity divided by a distance, and should be assigned a value of a characteristic velocity of the system multiplied by a value between 1/100 and 1/1000 and divided by a length of a cell. For example, for a system where the major inflow of fluid was from the left, 0.005 * ul/dx would be a reasonable value to assign to Dtest.

Optionally, a perturbation can be added to the initialization routine, which allows for turbulent phenomena such as the Karman vortex street to develop more rapidly in our simulations. A perturbation is added by assigning a pinwheel of velocities surrounding a point somewhere near the center of the mesh: $i = \bar{i}/3$, $j = \bar{j}/2$, for example. In this vicinity, velocities should be assigned:

$$u_{i+1/2,j} = perturb \tag{VII-59}$$

$$v_{i+1,j+1/2} = perturb \tag{VII-60}$$

$$u_{i+1/2,j+1} = -perturb \tag{VII-61}$$

$$v_{i,j+1/2} = -perturb\ , \tag{VII-62}$$

where *perturb* is a velocity typical to the system. For example, in a system where the major inflow of fluid is at the left wall, *perturb* could be equal to ul.

Flow velocities at the cell wall must be assigned so that the amount of inward flow is equal to the outward flow. That is

$$W dy \Sigma u_{\text{out}} + W dx \Sigma v_{\text{out}} = W dy \Sigma u_{\text{in}} + W dx \Sigma v_{\text{in}} \tag{VII-63}$$

or

$$dy \Sigma u_{\text{out}} + dx \Sigma v_{\text{out}} = dy \Sigma u_{\text{in}} + dx \Sigma v_{\text{in}}\ , \tag{VII-64}$$

where Σu_{out} and Σv_{out} are the sums of the velocities of all outward flowing zones and Σu_{in} and Σv_{in} are the sums of the velocities of all inward flowing zones. This constraint is necessary to ensure the conservation of mass in an incompressible system and must be applied according to the system that one wishes to represent. For example, if velocities at the top and bottom are set to zero, inflow of fluid is from the bottom and top thirds of the left wall, and outflow occurs all along the left wall, ur would be equal to $\frac{2}{3}$ul.

These velocities are assigned at the wall in an initial boundary condition procedure that is called only once during the program. This procedure assigns the ur value to the appropriate zones in the u(0,j) array, ul to the u (ibar,0) array, vb to v(i,0), and vt to the v(i,jbar) array. In the Karman vortex street problem, for example, ur, vt, and vb are assigned to all the elements of their respective arrays, and ul is assigned over the range u(0,1) to u(0,jbar/3) and u(0,jbar-(jbar/3)) to u(0,jbar). A value of zero is assigned from $\frac{\bar{j}}{3}+1$ to $\bar{j}-\frac{\bar{j}}{3}-1$. Note that a value $\bar{j}-\frac{\bar{j}}{3}$ is used rather than $\frac{2\bar{j}}{3}$ to preserve the symmetry of the system in cases when $\bar{j}$ is not divisible by three. The initial boundary need not assign pressures for the ghost zones, however, as the pressure values in the ghost zones are never used.

The initial boundary condition routine that sets flow velocities in and out of the system is contrasted with the tangential boundary condition routine, which is called at the beginning of each time cycle. This routine sets the flow velocities in the ghost zones that run parallel to the walls of the system: v at the left and right, and u at the top and bottom. These velocities are usually assigned using a FREE-SLIP METHOD, which assumes that fluid running parallel to the walls experiences no friction with that surface. This sort of boundary condition is used in situations where the layer of fluid that is affected by friction with the wall is much smaller than a cell. In these cases the effect of friction is negligible, allowing it to be approximated through the use of tangential velocities outside the walls equal to the tangential velocities inside the walls. This condition is expressed in the following equations:

$$u(i,0) = u(i,1) \tag{VII-65}$$

$$u(i,jbar+1) = u(i,jbar) \tag{VII-66}$$

$$v(0,j) = v(1,j) \tag{VII-67}$$

$$v(ibar+1,j) = v(ibar,j) \tag{VII-68}$$

which make up the tangential boundary condition routine that must be computed every time cycle.

There is also a third type of boundary condition procedure, the outflow boundary condition routine. This routine is called at every implicit iteration to update the velocities at any wall at which outflow occurs. Its purpose is to assign outflow velocities that accurately represent the outflow velocities of the system, while maintaining the balance between inward and outward flow as shown in Eq. (VII-64).

In order to accurately represent the physical system, the outward flow at a given cell wall must be proportional to the outward flow before that wall. That is

$$u_{\text{out}\bar{i}+1/2,j} \propto u_{\bar{i}-1/2,j}$$

$$u_{\text{out}1/2,j} \propto u_{3/2,j}$$

$$v_{\text{out}i,\bar{j}+1/2} \propto v_{i,\bar{j}-1/2}$$

$$v_{\text{out}i,1/2} \propto v_{i,3/2} \ ,$$

while also maintaining Eq. (VII-64).

By specializing to the case where there is no inward or outward flow at the top and bottom walls, Eq. (VII-64) can be written as

$$dy\Sigma u_{\bar{i}+1/2,j} = dy\Sigma u_{1/2,j} \qquad \text{(VII-69)}$$

or

$$\Sigma u_{\bar{i}+1/2,j} = \Sigma u_{1/2,j} \ . \qquad \text{(VII-70)}$$

We can now derive an expression for $u_{\bar{i}+1/2,j}$. As $u_{\bar{i}+1/2,j}$ is proportional to $u_{\bar{i}-1/2,j}$,

$$u_{\bar{i}+1/2,j} = A\,u_{\bar{i}-1/2,j} \ , \qquad \text{(VII-71)}$$

where A is a constant.

Substituting this equation into Eq. (VII-70) we obtain

$$A\,\Sigma u_{\bar{i}-1/2,j} = \Sigma u_{1/2,j} \qquad \text{(VII-72)}$$

or

$$A = \frac{\Sigma u_{1/2,j}}{\Sigma u_{\bar{i}-1/2,j}} \ . \qquad \text{(VII-73)}$$

We then substitute this value of A into Eq. (VII-71) to obtain an expression for outward boundary conditions for the Karman vortex street problem:

$$u_{\bar{i}+1/2,j} = u_{\bar{i}-1/2,j} \left(\frac{\Sigma u_{1/2,j}}{\Sigma u_{\bar{i}-1/2,j}} \right) . \tag{VII-74}$$

This equation is applied to the array of right fictitious zones at every implicit iteration. It describes the outward boundary conditions for the Karman vortex street problem.

Returning to the question of solving our equations using a partially-implicit method, we see that the code that must be written has been described to a large extent. The variable updating portion of the program is divided up into two sections: an explicit routine that computes $\bar{u}$ and $\bar{v}$, and an implicit routine that computes pressures and horizontal and vertical velocities.

The explicit routine is simply a double loop that computes $\bar{u}$ and $\bar{v}$ values for all cell edges except for those at the boundaries of the system; $\bar{u}$ and $\bar{v}$ are calculated only once each time step, using Eqs. (VII-48) and (VII-49).

The implicit routine is similar to the solver used in Chapter III. This section consists of a loop that iterates until the values of D have converged to within *Dtest*. At the beginning of each iteration, a variable *Dmax* is assigned a value of zero. The program then moves into a double loop that calculates $D_{i,j}$ for every point within the mesh and stores the largest absolute value for $D_{i,j}$ as *Dmax*. Pressures are reassigned according to Eq. (VII-57), velocities are updated, and outflow boundary conditions are implemented. A test is then made between *Dmax* and *Dtest*: if D has converged to within *Dtest*, the program moves to the next time step; if D has not yet converged, the loop iterates. This implicit loop should not be repeated more than about 100 times. An example for the coding of this loop is the following:

```
      times = 0
 100  Dmax = 0
c.. compute new D's
```

```
          do 200 i = 1, ibar
               do 200 j = 1, jbar
                  D(i, j) = (u(i, j)-u(i-1, j))/dx + (v(i, j)-v(i, j-1))/dy
                  if (abs(D(i, j)).gt.Dmax) Dmax = abs(D(i, j))
 200        continue
c.. compute P's
          do 300 i = 1, ibar
               do 300 j = 1, jbar
                  P(i, j) = P(i, j) - (beta*D(i, j))
300       continue
c.. compute u's and v's
          do 300 i = 1, ibar-1
               do 300 j = 1, jbar
                  u(i, j) = ubar(i, j) + (dt/dx)*(P(i, j)-P(i+1, j))
 300        continue
          do 400 i = 1, ibar
               do 400 j = 1, jbar-1
                  v(i, j) = vbar(i, j) + (dt/dy)*(P(i, j)-P(i, j+1))
 400        continue
        times = times + 1
c.. reset boundary conditions
          total = 0
          do 500 j=1, jbar
                  total = u(ibar-1, j) + total
 500        continue
          do 600 j=1, jbar
               u(ibar, j) = u(ibar-1, j)*(2./3.)*(jbar*ul/total)
```

```
600        continue
            if ((Dmax.lt.Dtest).or.(times.gt.100)) goto 1000
          goto 100
  1000  return
          end
```

The program should contain an output routine that represents the system in a way that is meaningful to the user. One useful output technique is the plotting of STREAMLINES, lines that indicate the path along which the fluid is flowing. These are determined by examining the direction of the motion of an arbitrary point in the fluid. Consider the following case:

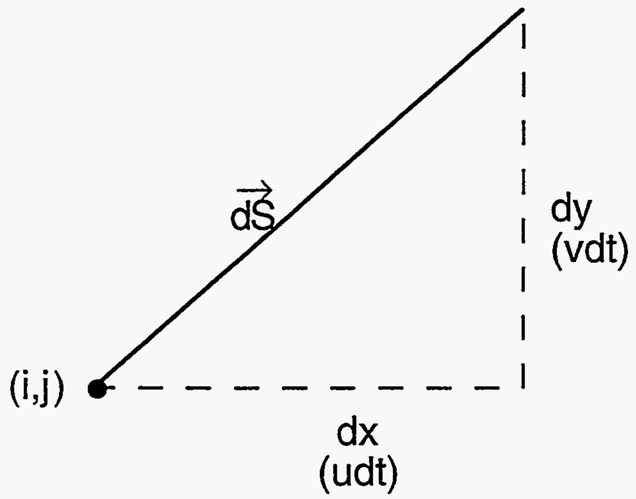


Figure VII-6

In this diagram, $\vec{dS}$ represents the displacement of the fluid at a point i,j over a time dt. This vector is made up of two components, dx and dy. The values of dx and dy are the horizontal and vertical velocities multiplied by the time step:

$$dx = u\,dt \tag{VII-75}$$

$$dy = v\,dt\,. \tag{VII-76}$$

A function $\psi(x, y)$ can be defined such that it is constant along the motion of the fluid. Along $d\vec{S}$, then,

$$d\psi = 0 \, . \tag{VII-77}$$

Returning to the mathematical identity expressed in Eq. (VII-53), we see that

$$d\psi = \frac{\partial\psi}{\partial x}dx + \frac{\partial\psi}{\partial y}dy \, . \tag{VII-78}$$

Substituting values from Eqs. (VII-75), (VII-76), and (VII-77) this equation becomes

$$0 = \frac{\partial\psi}{\partial x}udt + \frac{\partial\psi}{\partial y}vdt \tag{VII-79}$$

or

$$0 = \frac{\partial\psi}{\partial x}u + \frac{\partial\psi}{\partial y}v \, . \tag{VII-80}$$

To solve this equation for $\frac{\partial\psi}{\partial x}$ and $\frac{\partial\psi}{\partial y}$, we substitute in the following values:

$$\frac{\partial\psi}{\partial x} = -Av \tag{VII-81}$$

and

$$\frac{\partial\psi}{\partial y} = -Bu \, , \tag{VII-82}$$

where A and B are new variables. Equation (VII-80) then becomes

$$-A(uv) + B(uv) = 0 \tag{VII-83}$$

or

$$A = B \, . \tag{VII-84}$$

If we choose $A = B = 1$, Eqs. (VII-81) and (VII-82) become

$$\frac{\partial\psi}{\partial x} = -v \tag{VII-85}$$

and

$$\frac{\partial\psi}{\partial y} = u \, . \tag{VII-86}$$

By using Eq. (VII-85) in the mass equation (VII-14), we obtain

$$\frac{\partial}{\partial x}\left(\frac{\partial \psi}{\partial y}\right)+\frac{\partial}{\partial y}\left(-\frac{\partial \psi}{\partial x}\right)=0 \qquad \text{(VII-87)}$$

$$\frac{\partial^2 \psi}{\partial x \partial y}-\frac{\partial^2 \psi}{\partial x \partial y}=0\,. \qquad \text{(VII-88)}$$

This equation indicates that if either of Eqs. (VII-85) or (VII-86) is used to compute ψ over a system that satisfies the Eulerian mass equation, the other one will be automatically satisfied. Values of ψ, as described by either Eq. (VII-85) or Eq. (VII-86), are constant along the direction of fluid flow. A contour plot of the two-dimensional array of ψ's will therefore indicate the shape of the flow in the system. This array is numerically calculated by using ψ values that exist at the cell corners:

$$psi(i,j)=\psi_{i+1/2,j+1/2}\,. \qquad \text{(VII-89)}$$

Because velocities at the bottom cell wall are constant and therefore define a streamline, psi's along this boundary can all be set to a constant, 0 for example. The rest of the psi array can be calculated using the finite-difference approximation of Eq. (VII-86), namely

$$\frac{\psi_{i+1/2,j+1/2}-\psi_{i+1/2,j-1/2}}{dy}=u_{i+1/2,j} \qquad \text{(VII-90)}$$

or

$$\psi_{i+1/2,j+1/2}=\psi_{i+1/2,j-1/2}+dy u_{i+1/2,j}\,. \qquad \text{(VII-91)}$$

In code form this equation becomes

psi(i,j) = psi(i,j-1) + dy * u(i,j) . (VII-92)

Lines of constant psi can be plotted by using a contour plot routine, which will result in graphs that indicate the motion of the fluid at any given time step. These graphs use the reference frame where the obstacle is stationary but can be placed in the reference frame of the fluid by calculating psi as

psi(i,j) = psi(i,j-1) + dy * (u(i,j) -ur) . (VII-93)

Contour plots of ψ in both reference frames appear in the results portion of this chapter.

The output procedure is the last section that must be written in our incompressible two-dimensional Eulerian fluid-flow code. This code is structured as was shown in Fig. VII-5. A version of this code that includes streamlines can be used to examine the Karman vortex street problem, generating results such as appear below.

E. Simulation of the Karman Vortex Street

There are several different cases that can be examined using a two-dimensional fluid code. By varying the dimensions of the object, the speed of the flow, and the viscosity of the fluid, flow at various REYNOLDS NUMBERS can be examined. The Reynolds number is a dimensionless quantity that measures the ratio of advective effects to viscous effects in a system. For the Karman Vortex Street problem, it is calculated as

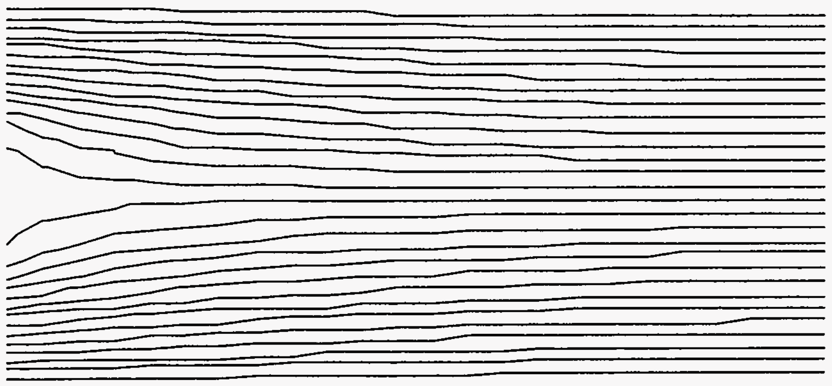
$$Re = \frac{h_{\text{obs}} u_{\infty}}{\nu} , \tag{VII-94}$$

where h_{obs} is the height of the obstacle, ν is the viscosity of the fluid, and u is the velocity of the fluid at a point far away from the obstacle. In our case

$$u_{\infty} \approx \text{ur} . \tag{VII-95}$$

As the Reynolds number increases, the system is likely to become more and more turbulent.

At low Reynolds numbers (numbers lower than about 4), the flow is steady and exhibits no flow separation. This behavior can be seen in Fig. VII-7 which is taken at a time of 25 (s) using the following parameters: xlen = 50 (cm), ylen 15 (cm), ibar= 50, jbar = 30, anu = 1.25 (cm^2/s), ul = 1.5 (cm/s), dt = 0.1 (s), and P0, u0, and v0 are all 0. The object is 5 (cm) wide, taking up the middle third of the left wall. This system has a Reynolds number of 4.

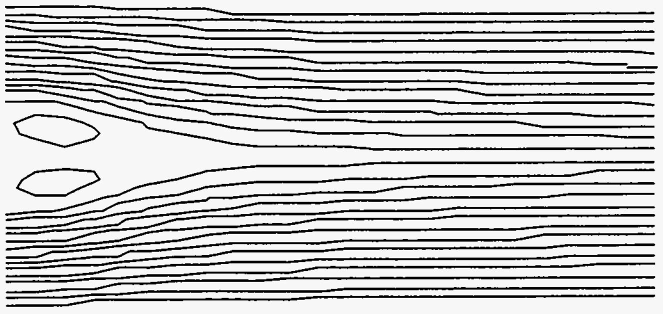


Streamlines at time 25
Reynolds Number = 4

Figure VII-7

Note that in this simulation *dx* is not equal to *dy*; this inequality demonstrates the fact that these quantities need not be equal for accurate results to be obtained.

At slightly higher Reynolds numbers (numbers above 4 to about 40), a pair of VORTICES form behind the object. These are areas where fluid is not moving along with the main flow, but rather circling behind the object. At these Reynolds numbers, the direction of the flow in some areas behind the object is opposite to that of the main flow stream. A flow containing vortices is illustrated in Fig. VII-8, which is generated using the same parameters as Fig. VII-7 except for *anu* which is 0.2 (cm/s^2).



Streamlines at time 25
Reynolds Number = 25

Figure VII-8

At Reynolds numbers between 40–500, the vortices become larger and begin to move away from the object as is illustrated in Fig. VII-9 (*anu* = 0.02; all other parameters are the same as in the previous graph).

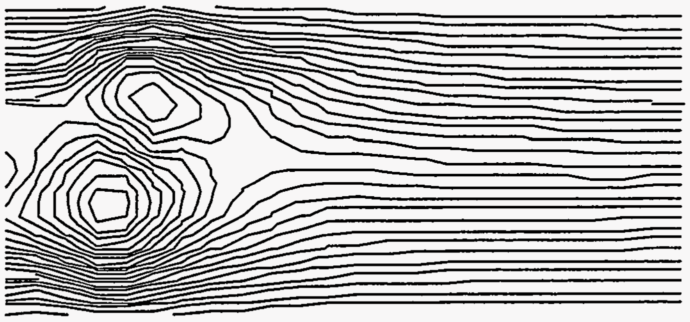


Figure VII-9

Notice that in this graph, the vortices are asymmetric. In nature, this asymmetry is initiated by the presence of miniature "flaws" in the fluid. Numerically, this asymmetry is a consequence of the perturbation that was added in the setup procedure.

The vortices move away from the object one at a time in an alternating fashion, creating a fluctuating stream, the Karman vortex street. Figure VII-10 shows the same system as Fig. VII-9 but at a later time, when the Karman vortex street has has time to develop.

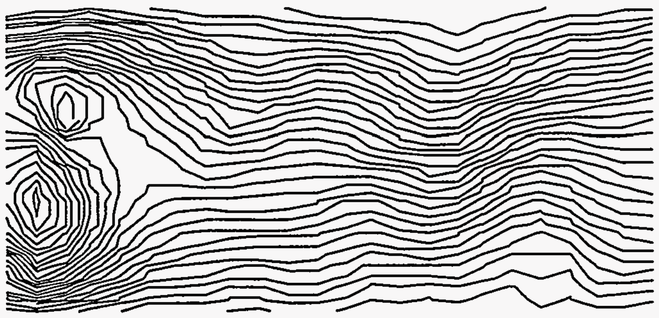
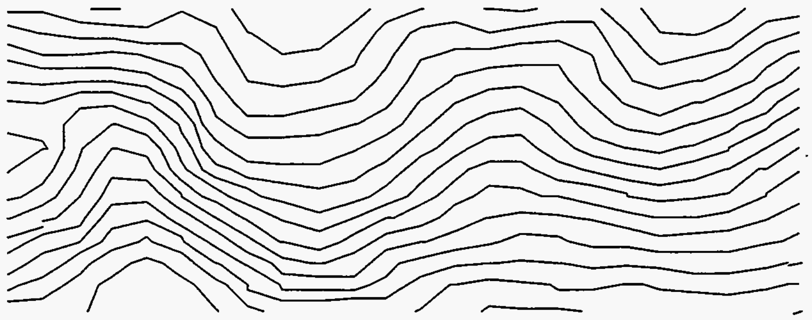


Figure VII-10

Note that the vortices that are shed from the object move downstream to the right with the main fluid flow, and cannot be seen in this graph.

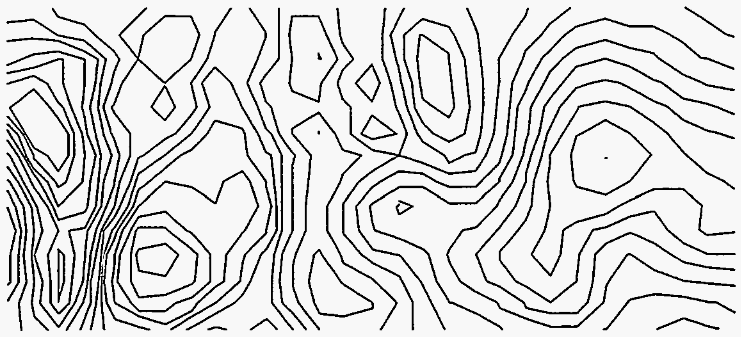
Figure VII-11 shows a fully developed street at a time of 100 (s). Other parameters are the same as in the previous graphs, except for *jbar*, which has been lowered to 15 in order to demonstrate that the vortex street can be simulated at relatively coarse resolutions.



Streamlines at time 100
Reynolds Number = 250

Figure VII-11

The vortex street can be better seen by placing the streamlines in the reference frame of the fluid, as if the object were moving and the fluid were stationary. This approach results in graphs such as Fig. VII-12.



Streamlines at time 100
Reynolds Number = 250
u_{ref} = 1.0 (fluid reference frame)

Figure VII-12

The fluctuations in the Karman vortex street occur at regular periods, as can be seen in the next series of graphs, obtained using the same parameters as the previous graph. Graphs appear in the reference frame of the object ($u_{\text{ref}} = 0.$).

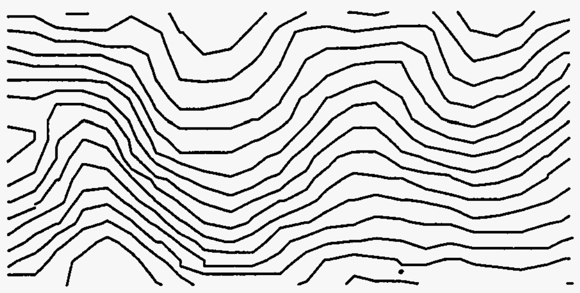


Figure VII-13

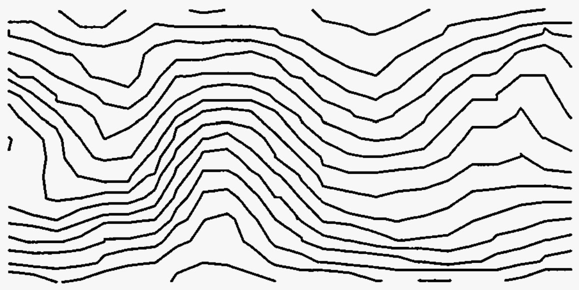


Figure VII-14

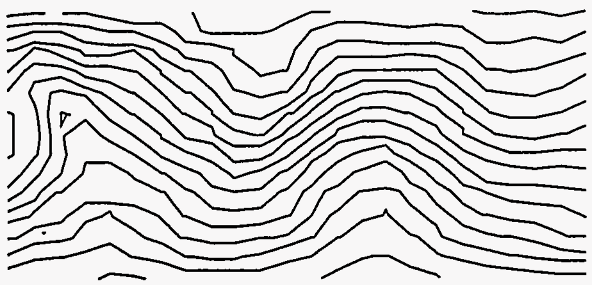


Figure VII-15

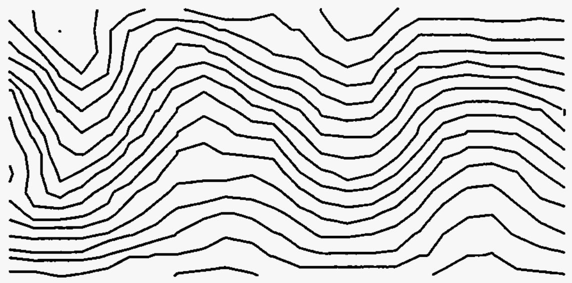


Figure VII-16

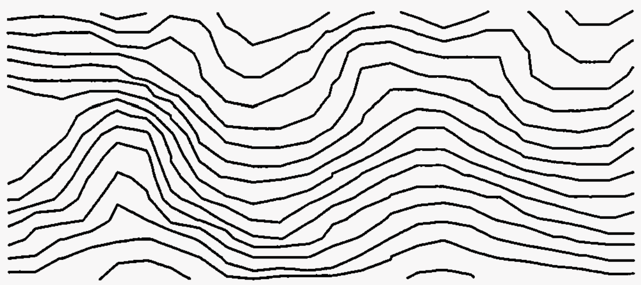


Figure VII-17

Analysis of these graphs indicates that the stream is fluctuating with a period of roughly 15 seconds.

We can use this period to calculate the STROUHAL NUMBER, which relates the period of the stream to the size of the object and the rate of the flow. The Strouhal number is a dimensionless quantity that is calculated as

$$St = \frac{h_{obs}}{u_\infty \tau_{\text{street}}}\,, \tag{VII-96}$$

where h_{obs} is the height of the obstacle, u_∞ is the velocity of the fluid at a point far away (in our case u_r), and τ_{street} is the period of one oscillation.

Experimentally, the Strouhal number in a Karman vortex street has been observed to be about 0.20. For our computational system, we calculate a Strouhal number of about 0.33. This difference in values can be explained by examining the differences between the laboratory experiments and our computational system.

In the laboratory, the Strouhal number is calculated by sending flow over a cylinder, whereas the computational results are obtained by blocking off the flow in a portion of a boundary. These two methods differ in that the fluid flow around the computational "object" moves parallel to the main flow, whereas the fluid flow around the laboratory cylinder moves outwards around the cylinder, spreading out before finally becoming parallel to the main fluid flow. Consequently, the object simulated computationally corresponds with a smaller experimental object. This effect is illustrated in Fig. VII-18.

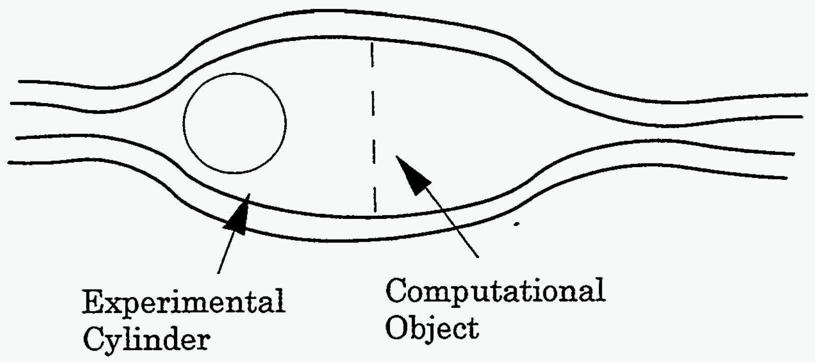
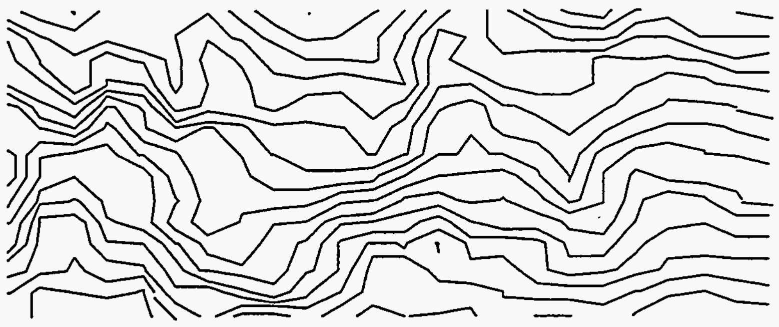


Figure VII-18

Our experimental Strouhal number is then smaller than the number calculated numerically. Assuming a ratio of about 2:3 between our real and computational objects, our computational Strouhal number would then relate to a experimental number of about 0.22, a value consistent with observed data.

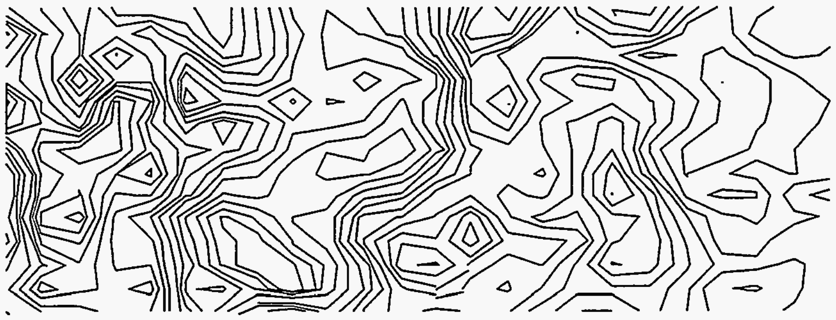
At very high Reynolds numbers (above about 500), miniature turbulent fluctuations occur within the vortex street and begin to drown out the fluctuating stream itself. Our computational results seem to simulate this case ($anu = 0.005$):



Streamlines at time 100
Reynolds Number = 1000

Figure VII-19

What we are actually observing in this graph, however, is not the turbulent fluctuations that drown out the vortex street, but rather a numerical instability that results from a violation of the diffusional stability condition. Viscosity has been reduced to a level at which it no longer counteracts the negative diffusion intrinsic to the centered difference momentum equation, and the solution becomes full of random highs and lows. This instability can be seen clearly by placing the graph in the reference frame of the fluid.



Streamlines at time 100
Reynolds Number = 1000
u_{ref} = 1.0 (fluid reference frame)

Figure VII-20

We have seen that the Karman vortex street can be modeled computationally and have discussed some of the theory associated with this phenomenon. We have also examined some of the inaccuracies that can result from our numerical approximations. In Chapter VIII, we will apply our two-dimensional fluid code to the simulation of more complicated systems, examining the modeling of obstacles placed within the flow passage itself and the simulation of heat flow.

VIII. ADDITIONS TO TWO-DIMENSIONAL FLUID CODE

A. Flow Regions with Obstacles

In this chapter we will be discussing several additional problems that can be modeled using a two-dimensional fluid flow code. The first of these problems is one in which an OBSTACLE is present in the flow region. For our purposes, we will define an obstacle as an object that prevents fluid from flowing through a specified region. Unlike the object simulated by the use of boundary conditions in the Karman vortex street problem, the obstacles that we will be examining in this section are found within the mesh and can be placed adjacent to the walls or anywhere in the flow region.

Obstacles are simulated by creating a boundary that exists within the flow region. For purposes of simplicity, we will limit the shape of our obstacles to be rectangles, but in principle, obstacles can be of many different shapes. A diagram of a rectangular obstacle appears in Fig. VIII-1:

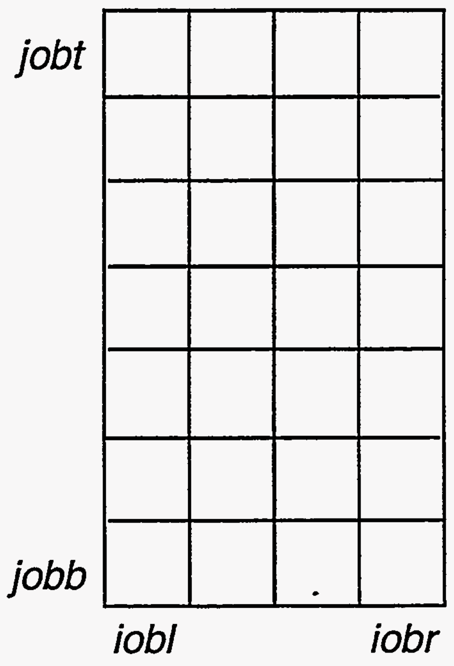


Figure VIII-1

In this figure, iobl and iobr are the i values at the left and right of the obstacle, and jobb and jobt are the j values at the bottom and top of the obstacle. These can be chosen to have values anywhere within the mesh, including adjacent to the walls.

Over the walls of the subregion described by these four values, the velocities normal to the obstacle are set to zero, and the tangential velocities are set according to the desired boundary conditions, for example, free slip boundary conditions. Values are assigned in an obstacle routine that is called at every implicit iteration.

This routine is made up of two main parts, the first of which sets the normal velocities to zero. This means that $u_{iobl-1/2,j}$ and $u_{iobr+1/2,j}$ are set to zero from jobb to jobt, and $v_{i,jobb-1/2}$ and $v_{i,jobt+1/2}$ are set to zero from iobl to iobr. In this is done with two loops, which appear as follows:

```
      do 100 j=jobb,jobt
         u(iobl-1,j) = 0
         u(iobr,j) = 0
100   continue

      do 200 i=iobl,iobr
         v(i,jobb-1) = 0
         v(i,jobt) = 0
200   continue
```

Note that in this code, u(iobl-1,j) and v(i,jobb-1) are set to zero rather than u(iobl,j) and v(i,jobb), because velocities exist at the right and top of the cells, whereas the normal velocities at the bottom and left of the obstacle are at the bottom and left of the cells.

If free-slip boundary conditions are desired, tangential velocities at the obstacle walls should be set to the value of the tangential velocities of the surrounding flow. The assignments are similar to those of the wall tangential boundary conditions described in the previous chapter. In this case

$$v_{iobl,j+1/2} = v_{iobl-1,j+1/2} \tag{VIII-1}$$

and

$$v_{iobr,j+1/2} = v_{iobr+1,j+1/2} \ , \tag{VIII-2}$$

from $jobb + 1/2$ to $jobt - 1/2$, and

$$u_{i+1/2,jobb} = u_{i+1/2,jobb-1} \tag{VIII-3}$$

and

$$u_{i+1/2,jobt} = u_{i+1/2,jobt+1} \ , \tag{VIII-4}$$

from $i_{obr+1/2}$ to $i_{obl-1/2}$.

Tangential velocities are not set to zero at the corner of the object because they would act as normal velocities at these points. Equations (VIII-1) through (VIII-4) appear in code form as:

```
      do 100 j=jobb,jobt-1
         v(iobl,j) = v(iobl-1,j)
         v(iobr,j) = v(iobr+1,j)
 100  continue

      do 200 i=iobl,iobr-1
         u(i,jobb) = u(i,jobb-1)
         u(i,jobt) = u(i,jobt+1)
 200  continue
```

Note that in this code, the loops run from jobb to jobt-1 and iobl to iobr-1, again due to the use of a staggered mesh with u's and v's that exist at the right and top cell walls.

With these two elements, the setting of the normal velocities to zero and the use of free slip boundary conditions, a routine can be written that creates a rectangular obstacle in any subregion of the mesh. Multiple objects can be simulated by multiple calls to the obstacle routine, with jobb1, jobt1, iobr1, and iobl1 specifying the dimensions of the first obstacle; jobb2, jobt2, iobr2, and iobl2 specifying the dimensions of the second obstacle; etc. These calls must be made at every implicit iteration, resulting in a two-dimensional fluid code as illustrated in Fig. VIII-2.

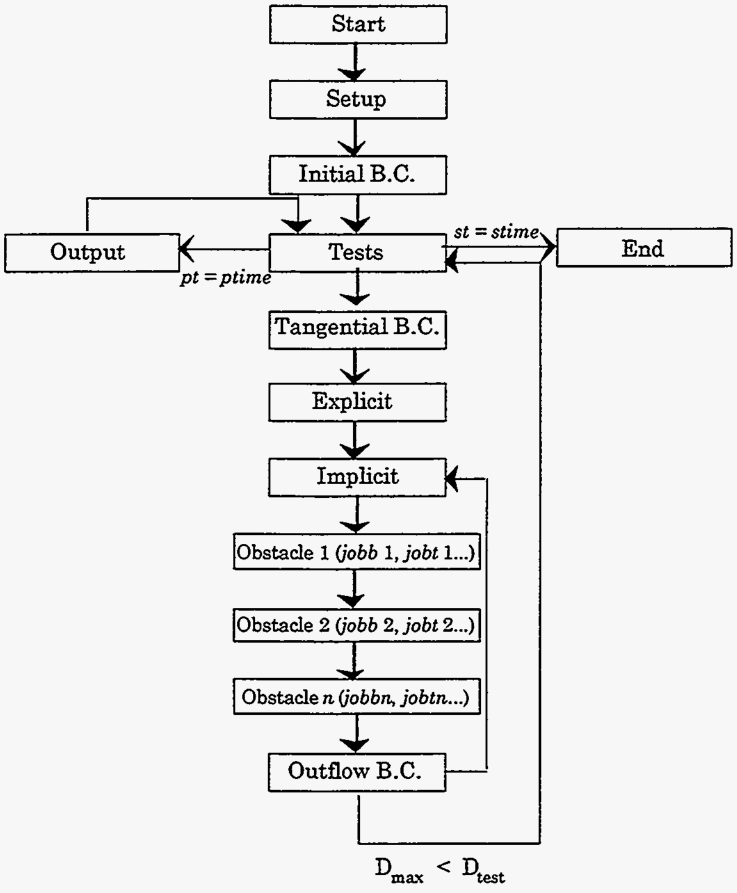
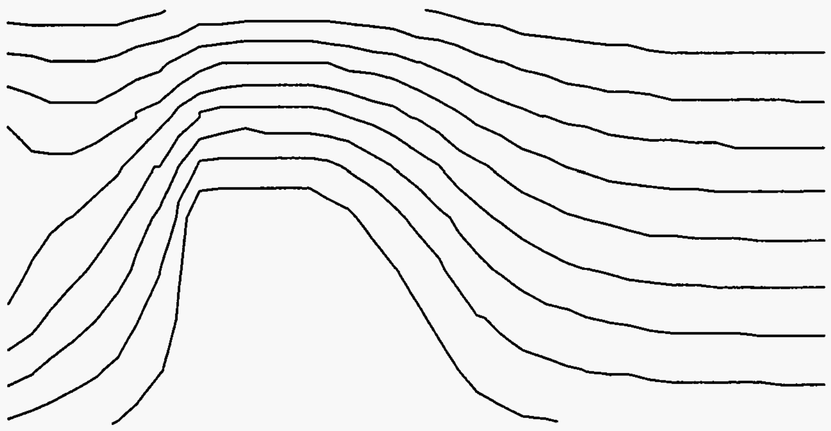


Figure VIII-2

Our program can now be used to simulate a number of interesting situations. This first series of plots uses the following parameters: xlen = 40 (cm) ylen = 10 (cm), ibar = 40, jbar = 10, and ul = ur = 1.0 (cm/s). The obstacle parameters are jobb = 1, jobt = 5, iobl =11, and iobr = 15. The dimensions of the obstacle are 5 (cm) $\times$ 5 (cm), and it is placed 10 cm down the flow passage.

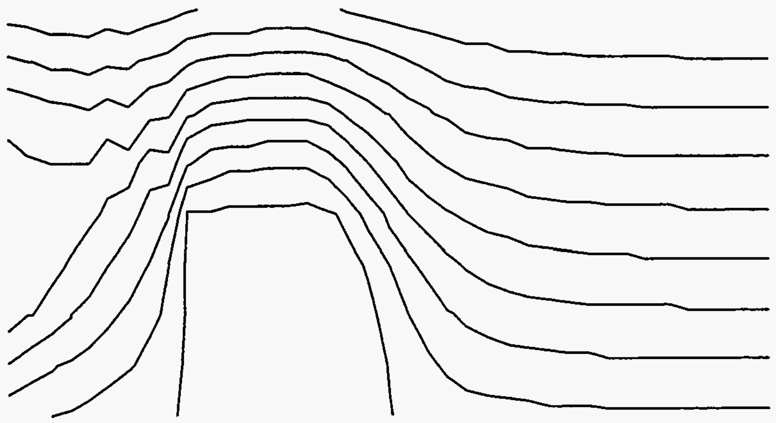
Figure VIII-3 is taken at a time of 5 (s) with a viscosity of 1 (cm^2/s). This results in a Reynolds number of 5.



Streamlines at time 5
Reynolds Number = 5

Figure VIII-3

If the viscosity is reduced to 0.1, so that the Reynolds number is 50, a vortex forms behind the object. This is illustrated in Figs. VIII-4 through VIII-6.



Streamlines at time 5
Reynolds Number = 50

Figure VIII-4

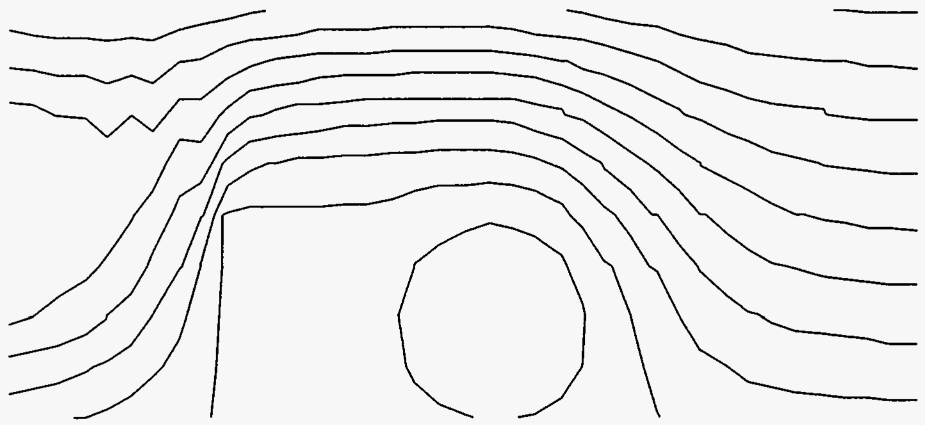

Streamlines at time 10
Reynolds Number = 50

Figure VIII-5

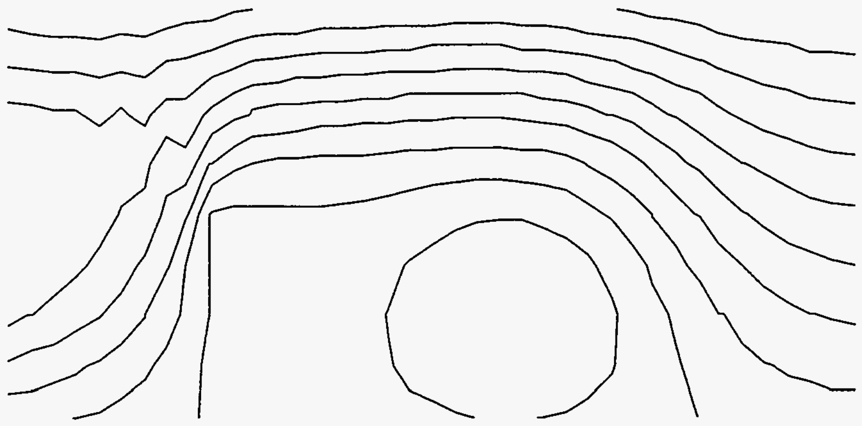
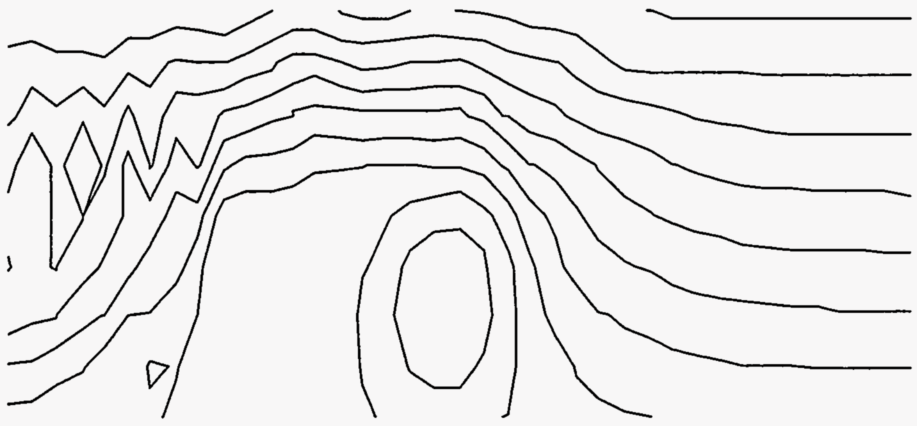

Streamlines at time 15
Reynolds Number = 50

Figure VIII-6

In these last three graphs, we can see the presence of a numerical instability that occurs when fluid accelerates. A careful truncation error analysis indicates that the finite-difference approximation of the momentum equation has a negative diffusion term that is associated with the acceleration of the fluid. When the fluid is accelerating, as is the case when the fluid moves over the object from the right, this negative diffusion results in a numerically unstable solution. We can see this instability in the jagged streamlines in this portion of the graph. When the fluid is decelerating, as is the case as the fluid moves away from the object into the open flow channel, there is an additional positive diffusion. Hence, this portion of the graph remains numerically stable.

As viscosity is again lowered, the contrast between these stable and unstable regions becomes clearer. Figure VIII-7 illustrates the results of a simulation with a viscosity of 0.02 (cm^2/s)



Streamlines at time 15
Reynolds Number = 250

Figure VIII-7

B. Heat Transfer

The second topic in our study of additions to a two-dimensional incompressible fluid flow code is the modeling of heat transfer. This modeling requires the addition of a new array of temperatures that exists at the cell centers, and must be declared and initialized in the setup procedure. The resulting mesh is pictured in Fig. VIII-8.

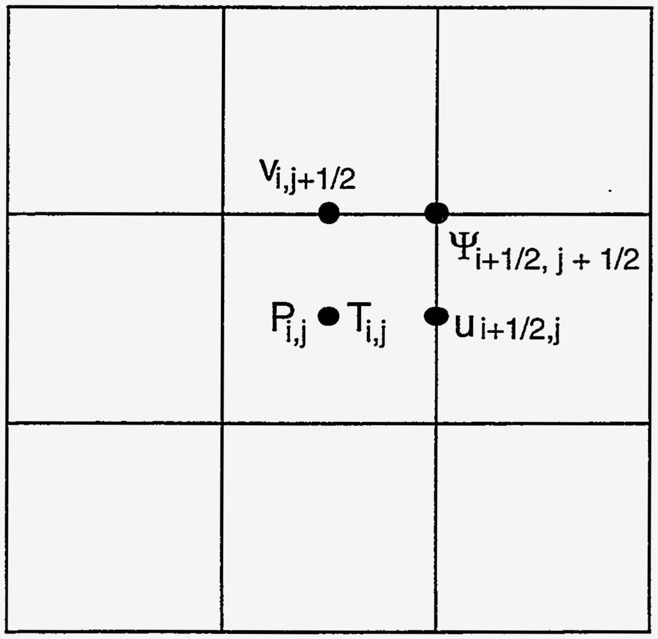


Figure VIII-8

The equation that describes the evolution of temperature is similar to the two-dimensional momentum equation, Eq. (VII-34). The two-dimensional temperature equation is

$$\frac{\partial T}{\partial t} + \frac{\partial uT}{\partial x} + \frac{\partial vT}{\partial y} = \sigma \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right) , \tag{VIII-5}$$

where σ is once again the thermometric conductivity of the material.

This equation is made up of three major types of terms: the explicit change in temperature with time $\left(\frac{\partial T}{\partial t}\right)$, the advective terms in both directions $\left(\frac{\partial uT}{\partial x} , \frac{\partial vT}{\partial y}\right)$, and the diffusion term $\left(\sigma\left(\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial y^2}\right)\right)$. These terms are the result of an analysis similar to that used to derive the two-dimensional momentum equation in Chapter VII.

In finite-difference form, Eq. (VIII-5) appears as

$$\frac{T_{i,j}^{n+1}-T_{i,j}^{n}}{dt}+\frac{(uT)_{i+1/2,j}^{n}-(uT)_{i-1/2,j}^{n}}{dx}+\frac{(vT)_{i,j+1/2}^{n}-(vT)_{i,j-1/2}^{n}}{dy}= \\ \sigma\left(\frac{T_{i+1,j}^{n}+T_{i-1,j}^{n}-2T_{i,j}^{n}}{dx^2}\right)+\sigma\left(\frac{T_{i,j+1}^{n}+T_{i,j-1}^{n}-2T_{i,j}^{n}}{dy^2}\right) \tag{VIII-6}$$

or

$$T_{i,j}^{n+1}=T_{ij}^{n}+dt\Big(\frac{(uT)_{i-1/2,j}^{n}-(uT)_{i+1/2,j}^{n}}{dx}+\frac{(vT)_{i,j-1/2}^{n}-(vT)_{i,j+1/2}^{n}}{dy} \\ +\frac{\sigma}{dx^2}\left(T_{i+1,j}^{n}+T_{i-1,j}^{n}-2T_{i,j}^{n}\right)+\frac{\sigma}{dy^2}\left(T_{i,j+1}^{n}+T_{i,j-1}^{n}-2T_{i,j}^{n}\right)\Big)\,. \tag{VIII-7}$$

Temperatures at cell walls are calculated through the use of the donor-cell method, by creating two two-dimensional arrays: idnr and jdnr. These arrays are made up of integers that are calculated at the cell walls. idnr is calculated from the u velocities at position $i+1/2, j$ and is zero if the flow is from left to right and 1 if the flow is from right to left. jdnr is calculated from the v velocities at position $i, j+1/2$ and is zero if the flow is upwards and 1 if the flow is downwards. These two arrays are then used in a double lookup fashion, as was done in Chapter V. The advective terms of Eq. (VIII-5), thus appear as

$$\frac{\partial uT}{\partial x}=\text{(u(i,j)*T(i+idnr(i,j),j) - u(i-1,j)*T(i-1+idnr(i-1,j),j))/dx}$$
$$\frac{\partial vT}{\partial y}=\text{(v(i,j)*T(i,j+jdnr(i,j)) - v(i,j-1)*T(i,j-1+jdnr(i,j-1)))/dy}\,.$$

Equation (VIII-7) is implemented in a double loop just before the the explicit calculation of $\bar{u}$ and $\bar{v}$. The addition of this double loop is the first major modification that must be made to our code to simulate the transfer of heat.

The second major modification is the addition of a buoyancy term to the $\bar{v}$ equation. This term represents the upward acceleration created by a decrease in density due to the heating of the fluid. This upward acceleration is equal to the gravitational force on the system multiplied by the ratio of the density of the fluid to a given reference density:

$$-\frac{\rho_{i,j}}{\rho_0}g\,,$$

where $\rho_{i,j}$ is the density at a point i, j, ρ_0 is the base density of the fluid, and g is the gravity of the system, defined as negative in the downward direction. On Earth $g = -9.8$ m/s^2.

This use of a change in density creates an apparent inconsistency between the buoyancy term and the the rest of the terms in the vertical momentum equation. The vertical momentum equation now appears as

$$\frac{\partial v}{\partial t} + \frac{\partial uv}{\partial x} + \frac{\partial v^2}{\partial y} = -\frac{\partial P}{\partial y} + v\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial dy^2}\right) - \frac{\rho}{\rho_0} g \, . \tag{VIII-8}$$

In this equation, all terms assume constant density except the buoyancy term. This approximation is called the BOUSSINESQ APPROXIMATION for natural convection problems. It can be used in cases where the driving forces for velocity are the result of small changes in density. Here, the buoyancy term is $O(d\rho)$ while the effect of changing density on the other terms in $O(d\rho^2)$. Since $d\rho$ is very small, $d\rho^2$ is negligible. For the buoyancy term, the following analysis is used to express the driving force in terms of temperature.

We begin with Eq. (IV-23):

$$p = (\gamma - 1)\rho I \, . \tag{IV-23}$$

and rewrite I, the internal energy, as the temperature times the specific heat, TC_v, to obtain

$$p = (\gamma - 1)\rho C_v T \tag{VIII-9}$$

or, solving for ρ

$$\rho = \frac{p}{(\gamma - 1)C_v T} \, . \tag{VIII-10}$$

Pressure in this equation is actually made up of two different pressures: the reference pressure, or nominal pressure of the surroundings, and the dynamic pressure, which changes according to the motion of the fluid. Because our pressure terms have only dealt with the change in pressure, our p from Chapter VII was essentially p_{dyn}, the dynamic pressure. In Eq. (VIII-10) p is no longer p_{dyn} but the sum of the nominal and dynamic pressures:

$$p = p_n + p_{dyn} \, . \tag{VIII-11}$$

Substituting this term into Eq. (VII-10) gives us

$$\rho = \frac{p_n + p_{dyn}}{(\gamma - 1)C_v T} \; . \tag{VIII-12}$$

But the nominal pressure of the system is much larger than the dynamic pressure, so we can ignore p_{dyn} in this equation and write

$$\rho = \frac{p_n}{(\gamma - 1)C_v T} \; . \tag{VIII-13}$$

The substitution of this equation into our definition of the buoyancy term yields

$$\frac{\rho}{\rho_0} g = g \frac{\frac{p_n}{(\gamma+1)C_v T}}{\frac{p_n}{(\gamma+1)C_v T_0}} \; , \tag{VIII-14}$$

where T_0 is the reference temperature of the system. Assuming that the nominal pressure is unchanging, this equation becomes

$$\frac{\rho}{\rho_0} g = g \frac{T_0}{T} \; . \tag{VIII-15}$$

If we define a variable δ such that

$$\delta \equiv \frac{T - T_0}{T_0} \; ,$$

then

$$T = T_0(1 + \delta) \; ; \tag{VIII-16}$$

and the buoyancy term becomes

$$\frac{\rho g}{\rho_0} = \frac{g}{1 + \delta} \; . \tag{VIII-17}$$

By expanding the $\frac{1}{1+\delta}$ term we obtain a series

$$\frac{1}{1 + \delta} = 1 - \delta + \delta^2 - \delta^3 \; . \tag{VIII-18}$$

Ignoring the terms of second and higher order, we can use this expansion to write our buoyancy term as

$$\frac{\rho g}{\rho_0} \approx \left[1 - \left(\frac{T - T_0}{T_0} \right) \right] g \; . \tag{VIII-19}$$

If we look at this term in conjunction with the pressure term and expand pressure to represent both the nominal and dynamic elements we have

$$-\frac{\partial \frac{p_n}{\rho_0}}{\partial y} - \frac{\partial \frac{p_{dyn}}{\rho_0}}{\partial y} + g - g\frac{T - T_0}{T_0} \ .$$

To maintain atmospheric equilibrium, the nominal pressure must satisfy the equation

$$p_n = g\rho_0 y + C \ , \tag{VIII-20}$$

where C is some constant. Equation (VIII-20) can also be written as

$$\frac{p_n}{\rho_0} = gy + \frac{C}{\rho_0} \ . \tag{VIII-21}$$

We can use this value in our reference pressure term to obtain

$$\frac{\partial \frac{p_n}{\rho_0}}{\partial y} = g \ . \tag{VIII-22}$$

This g cancels with the g from the buoyancy equation, leaving

$$-\frac{\partial P}{\partial y} - g\frac{T - T_0}{T_0} \ .$$

If we choose T_0 to be 273°K, then the buoyancy term becomes

$$-g\frac{T}{273} \ ,$$

where T is in expressed in °C. This is more often written as

$$-g\beta T \ ,$$

where $\beta \equiv \frac{1}{T_0} \equiv \frac{1}{273}$. Our equation for $\bar{v}$ is then equal to the old $\bar{v}$ modified by a buoyancy term,

$$\bar{v}_{\text{buoy}} = \bar{v}_{\text{noheat}} - g\beta T dt \ , \tag{VIII-23}$$

where g is negative for a downward force of gravity.

The third major modification that must be made to simulate the effect of heat in a two-dimensional incompressible fluid is the implementation of thermal boundary conditions. These conditions should be calculated once per time cycle in a procedure that is called after the tangential boundary conditions. There are two types of thermal boundary conditions that we will use: INSULATED and PRESCRIBED.

Insulated means that there is no heat fluxed across the wall in question. This situation occurs when there is no temperature gradient across the wall:

$$T_{\text{outside}} = T_{\text{inside}} \,. \tag{VIII-24}$$

Insulated boundaries are contrasted with prescribed boundary conditions, which were used in the one-dimensional heat flow problem. For this condition the temperature gradient across the wall is chosen such that the temperature at the wall is a constant:

$$T_{\text{outside}} = 2T_{\text{wall}} - T_{\text{inside}} \,. \tag{VIII-25}$$

Together these two boundary conditions may be used, for example, to create a system that is insulated on three walls and a portion of the fourth one but contains a HOT SPOT which uses prescribed boundary conditions. This system would appear as in Fig. VIII-9.

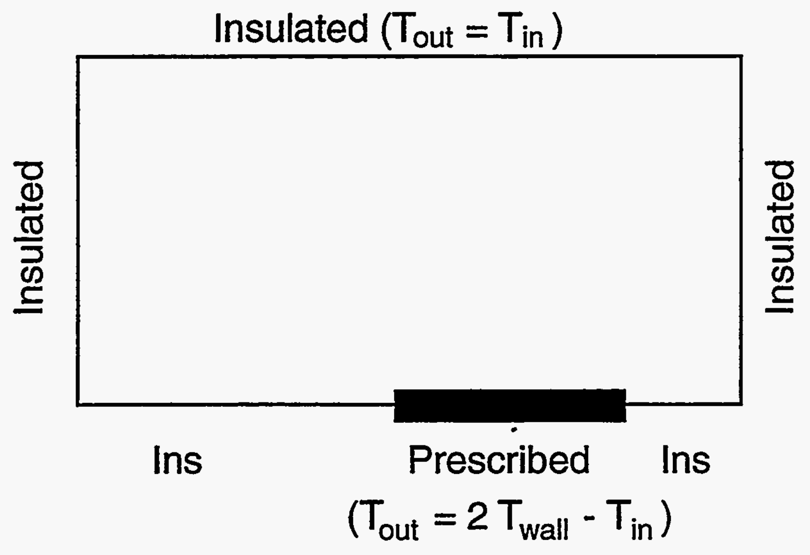

Insulated ($T_{out} = T_{in}$)
Insulated
Insulated
Ins
Prescribed
Ins
($T_{out} = 2\,T_{wall} - T_{in}$)

Figure VIII-9

Each of these sections of wall is described by assigning T_{out} values to the appropriate arrays in the system. For example, a procedure that implements insulated boundary conditions along the top wall, the sides, and the left half of the bottom wall, and implements prescribed boundary conditions at the right half of the bottom, appears as follows:

```
c.. insulated sides
        do 100 j =1,jbar
           T(0,j) = T(1,j)
           T(ibar+1,j) = T(ibar,j)
 100    continue

c.. insulated top and bottom
        do 200 i =1,ibar
           T(i,0) = T(i,1)
           T(i,jbar+1) = T(i,jbar)
 200    continue

c.. hot spot
        do 300 i = ibar/2,ibar
           T(i,0) = 2*Twall - T(i,1)
 300    continue
```

With these three major elements: the calculation of the heat transfer equation, the use of a buoyancy term from the Boussinesq approximation, and the implementation of insulated and prescribed temperature boundary conditions, heat transfer in a two-dimensional incompressible fluid can be modeled computationally. A diagram illustrating the interactions of these three elements appears below:

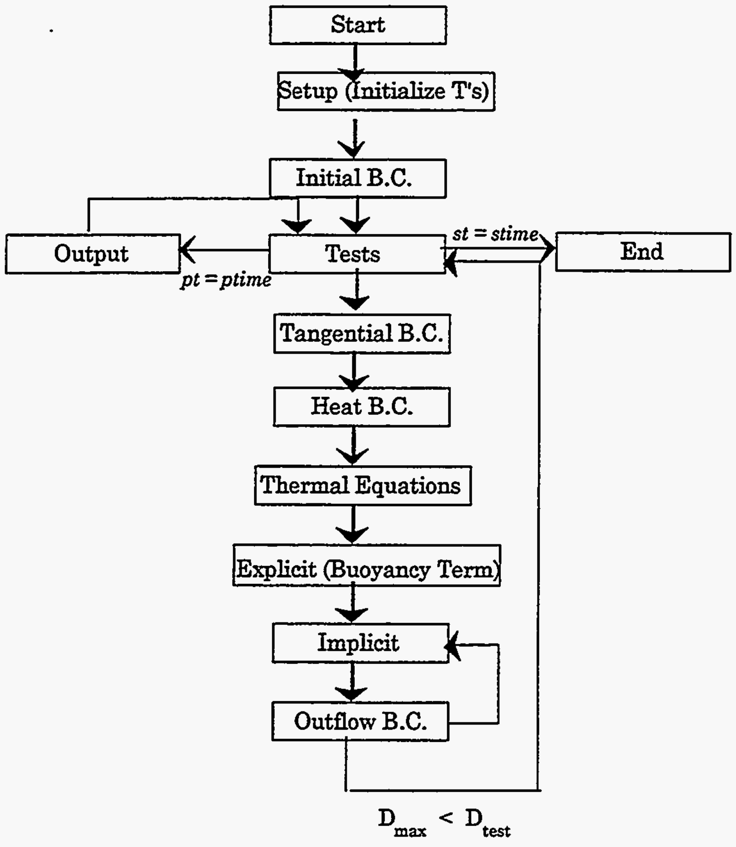


Figure VIII-10

C. Convection Calculations

Our two-dimensional fluid code that includes a heat transfer model can be used to study the phenomenon of NATURAL CONVECTION. Natural convection is the circulating motion of fluid between regions of different temperatures due to the difference in the fluid density at each of these temperatures. It can be described by using the example of an initially cold room in which a heat source is placed in one corner. The heat source heats the air around it, consequently reducing the density of that air. The heated air then moves upwards and across the ceiling, where it is cooled back to its original temperature. Once again dense, the cool air moves down towards the floor as new heated air flows up from the heat source. Finally, the dense air finds it way back to the heat source, and the cycle is repeated. This cycle is illustrated in Fig. VIII-11.

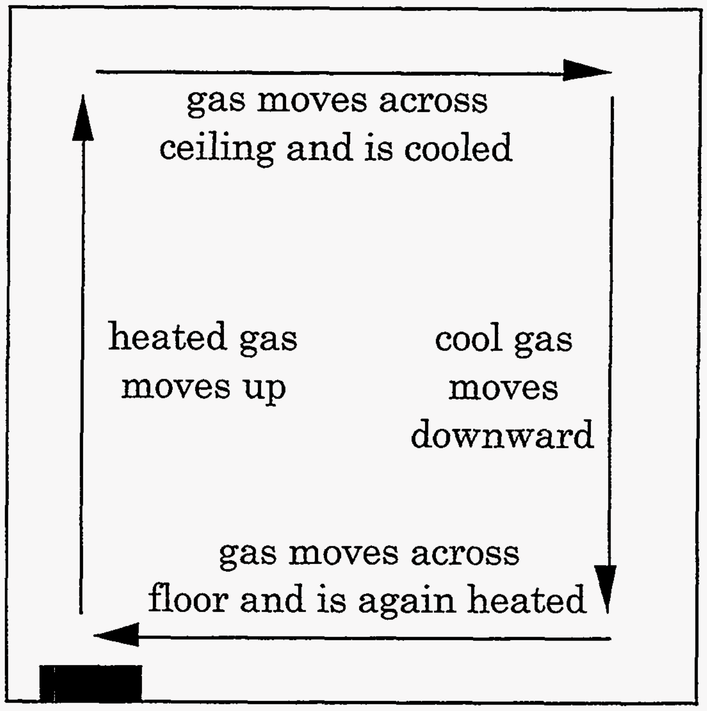


Figure VIII-11

Using our two-dimensional fluid code with heat, we can generate results that demonstrate this process. Figures VIII-12, 13, 14, and 15 are plots of streamlines of a fluid experiencing natural convection. These plots use the following set of parameters: T0 = 0 (°C) ibar = 15, jbar = 15, xlen = 3 (m), ylen = 3 (m), anu = 1 $\times 10^{-4}$ (m^2/s). All ghost zones use insulated boundary conditions except zones 1–7 on the bottom wall, where the wall is set to a prescribed temperature of 100°C.

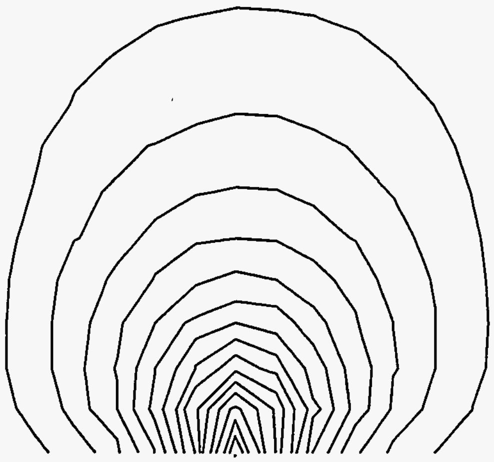

Streamlines at time 5 (s)

Figure VIII-12

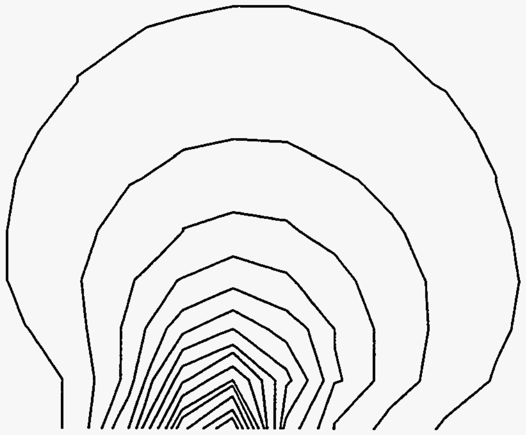

Streamlines at time 20 (s)

Figure VIII-13

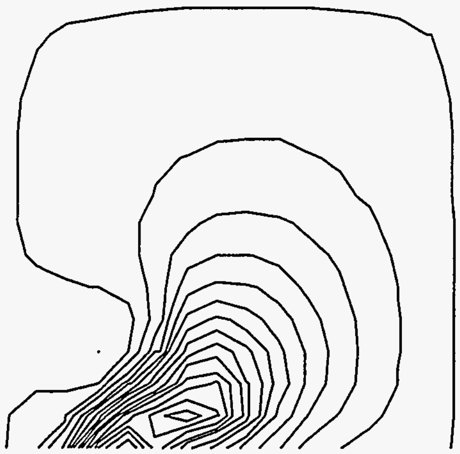

Streamlines at time 40 (s)

Figure VIII-14

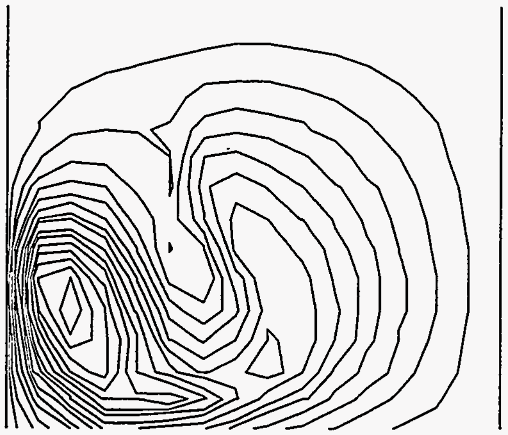

Streamlines at time 60 (s)

Figure VIII-15

Figures VIII-16 through VIII-19 are contour plots of temperature in this fluid at times of 5, 20 40, and 60 seconds. These plots illustrate the flow of heat from the hot spot.

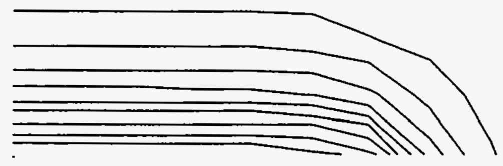


Figure VIII-16

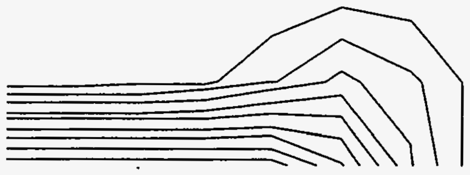


Figure VIII-17

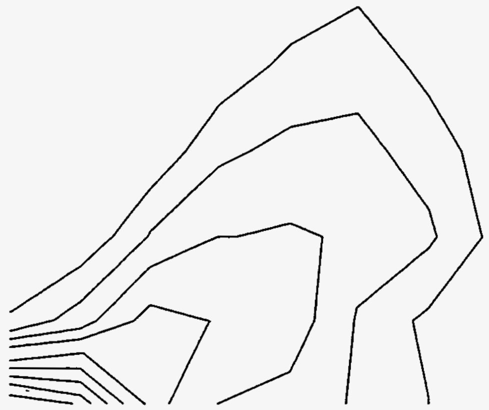


Figure VIII-18

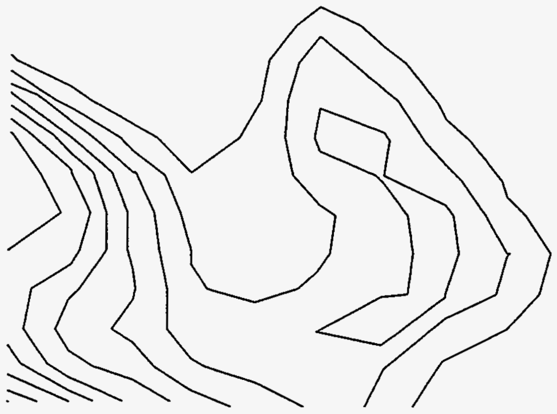


Figure VIII-19

In a fluid in which natural convection occurs, the rate of heat flow is greater than that of a fluid that is not in motion, because heat is not only being conducted but advected by the circular motion of fluid. A ratio can be formed between the total heat flux in a system and the heat flux due only to convection, such that

$$Nu = \frac{\text{Total}}{\text{Conductive Flux}}, \tag{VIII-26}$$

where Nu is a dimensionless quantity called the NUSSELT NUMBER.

An example of a system for which the Nusselt number is often calculated is the BENARD PROBLEM. This system is made up of a long, narrow flow passage that is heated at the bottom, cooled at the top, and insulated along each side, as illustrated in Fig. VIII-20.

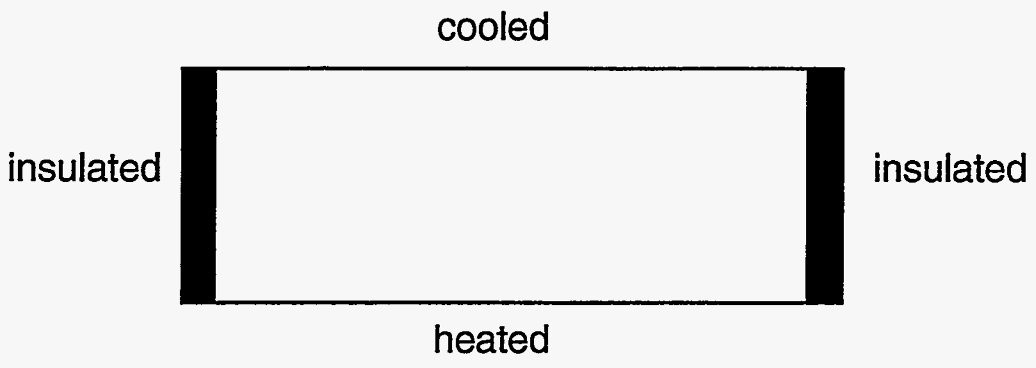


Figure VIII-20

Parameters are variable, such that the Nusselt number in this system can be observed at different RAYLEIGH NUMBERS. The Rayleigh number relates the magnitudes of the buoyancy and viscous forces in a system. In the Benard problem, the Rayleigh number is calculated as:

$$Ra = \frac{-gh^3\beta\Delta T}{\nu\sigma}\ , \qquad \text{(VIII-27)}$$

where g is the acceleration of gravity (defined as negative if downward), h is the height of the passage, ΔT is the difference in temperatures between the top and the bottom of the passage, ν is the viscosity of the fluid, σ is the themometric conductivity of the fluid, and β is the volumetric coefficient of expansion, which for gases is the reciprocal of the reference temperature $\left(\frac{1}{T_0}\right)$.

The equation for the computational calculation of the Nusselt number can be derived by examining the conductive and actual heat fluxes. In this system, the conductive heat flux is calculated by Fick's Law, expressed in terms of themometric conductivity:

$$\text{conductive flux} = \frac{\sigma \cdot \rho \cdot b \cdot (T_{\text{bot}} - T_{\text{top}})}{\bar{j}dy}\ , \qquad \text{(VIII-28)}$$

where ρ is the density and b is the specific heat of the fluid. The actual flux of heat across a given plane existing at a vertical position of $j + 1/2$ is made up of both conductive and advective fluxes. This flux appears as

$$\text{Actual Flux} = \rho \cdot b \left(\frac{\sigma}{dy} \frac{\sum_{i=1}^{\bar{i}} T_{i,j} - T_{i,j+1}}{\bar{i}} - \frac{\sum_{i=1}^{\bar{i}} v_{i,j+1/2} T_{i,j+1/2}}{\bar{i}} \right) . \tag{VIII-29}$$

These equations can be substituted into Eq. (VIII-26) to obtain

$$Nu = \frac{\rho \cdot b \left(\frac{\sigma}{dy} \frac{\sum_{i=1}^{\bar{i}} T_{i,j} - T_{i,j+1}}{\bar{i}} - \frac{\sum_{i=1}^{\bar{i}} v_{i,j+1/2} T_{i,j+1/2}}{\bar{i}} \right)}{\sigma \cdot \rho \cdot b \frac{(T_{\text{bot}} - T_{\text{top}})}{\bar{j} dy}} , \tag{VIII-30}$$

which reduces to:

$$Nu = \frac{\bar{j} \left(\sum_{i=1}^{\bar{i}} T_{i,j} - T_{i,j+1} - \frac{dy}{\sigma} \sum_{i=1}^{\bar{i}} v_{i,j+1/2} T_{i,j+1/2} \right)}{\bar{i} \left(T_{\text{bot}} - T_{\text{top}} \right)} . \tag{VIII-31}$$

If we choose to compute the Nusselt number at the bottom and top of the system, then we have no advective flux, and our equation becomes

$$Nu_{\text{bot}} = \frac{2\bar{j}}{\bar{i}} \frac{\sum_{i=1}^{\bar{i}} T_{\text{bot}} - T_{i,1}}{T_{\text{bot}} - T_{\text{top}}} \tag{VIII-32}$$

and

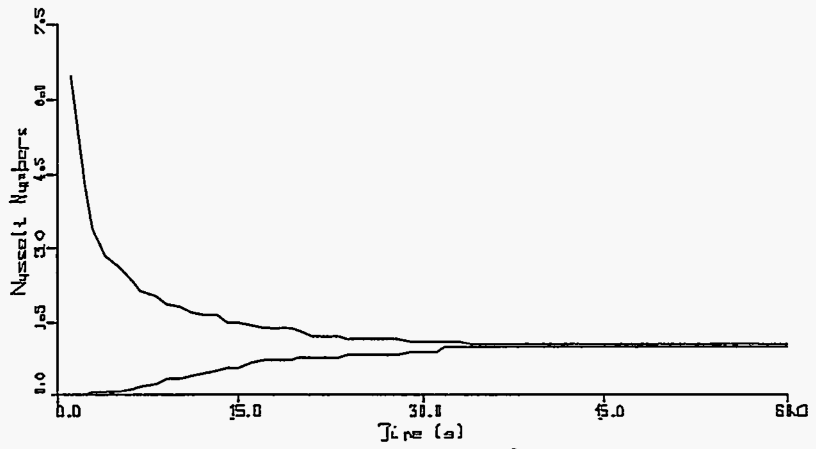
$$Nu_{\text{top}} = \frac{2\bar{j}}{\bar{i}} \frac{\sum_{i=1}^{\bar{i}} T_{i,\bar{j}} - T_{\text{top}}}{T_{\text{bot}} - T_{\text{top}}} . \tag{VIII-33}$$

When both of these numbers have equal values, heat flow into the system from the bottom is equal to heat flow out of the system through the top, and the system has reached a steady state.

A routine to compute Nusselt numbers can be added to the output portion of our program. The code should be similar to the following:

```
      ttot = 0.
      btot = 0.
      do 100 i=2,ibar+1
         ttot = ttot - (tTmp-T(i,jbar+1))
         btot = btot + (bTmp-T(i,2))
100   continue
      nm = (2*jbar*dx)/(xlen*(bTmp-tTmp))
      tnuss = ttot*nm
      bnuss = btot*nm
```

Using this routine, we can calculate the Nusselt number at the top and bottom in systems with various Rayleigh numbers. The next set of graphs are of a system with the following parameters: ibar = 40, jbar=8, xlen = 5 (m), ylen = 1(m), g = -10 (m/s^2), sigma = 0.01 (m^2/s^2), anu = 0.01 (m^2/s^2), beta = 1/300 (1/°C), T_{top} = 0 (°C). These parameters correspond to a Rayleigh number that is equal to the $T_{\text{bottom}} \times 333$. At a temperature of 1°C, the top and bottom Nusselt numbers converge to 1 as is shown in the following graph:

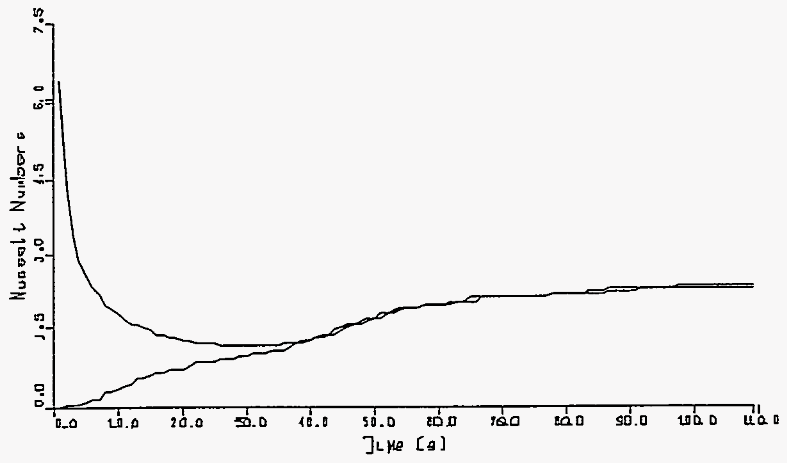


Nusselt numbers
Rayleigh number = 333

Figure VIII-21

In this system, heat transfer is purely by conduction.

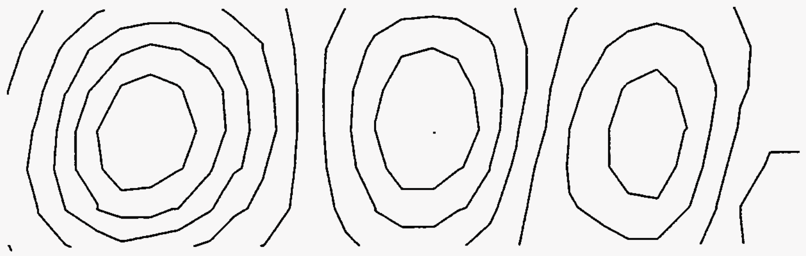
At a bottom temperature of 5°C, the top and bottom Nusselt numbers converge at a value of 2.33, indicating that the system has become more convective in nature. This system is illustrated in Fig. VIII-22.



Nusselt numbers
Rayleigh number = 1665

Figure VIII-22

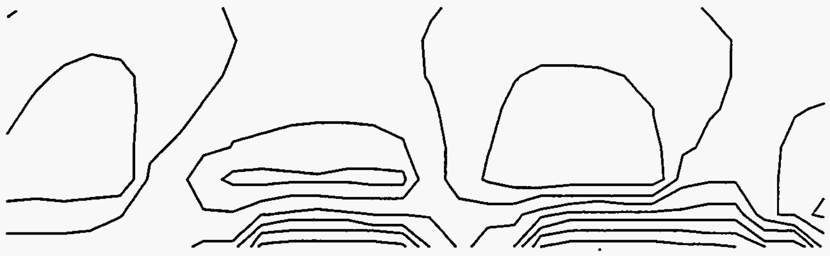
Flow at this Rayleigh number appears as is shown in Fig. VIII-23.



Streamlines at time 10 (s)
Rayleigh number = 1665

Figure VIII-23

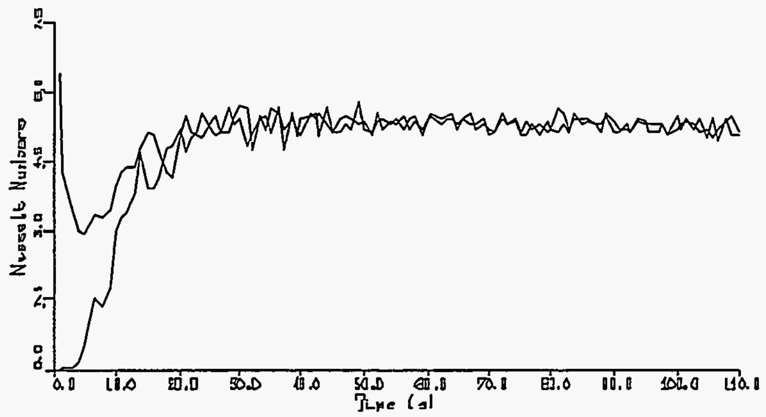
Heat contours appear in Fig. VIII-24.



Heat Contours at time 10 (s)
Rayleigh number = 1665

Figure VIII-24

At higher Rayleigh numbers, such as 2×10^4, corresponding with a temperature gradient of 67°C, the Nusselt number is even higher, but the Courant instability begins to affect the calculation of these values, as can be seen in Fig. VIII-25



Nusselt numbers
Rayleigh number = 2×10^4

Figure VIII-25

Despite the presence of this instability, it is possible to use our code to calculate Nusselt numbers at different Rayleigh numbers to within a reasonable degree of accuracy. The data from one such study appears below:

T_{bot}	Nu	Ra
.5	1.00	167
1	1.00	333
3	1.01	1000
4	1.40	1332
5	2.33	1665
10	3.03	3330
33	3.92	10^4
67	5.25	2×10^4
333	7.00	10^5
1000	8.50	3.33×10^5

If we compare these Nusselt numbers with numbers that have been generated from numerous different experiments, we see that our computational values are very similar, as can be seen in Fig. VIII-26. The experimental data for this graph is taken from S. Chandrasekar, *Hydrodynamic and Hydromagnetic Stability* (Dover, New York 1961).

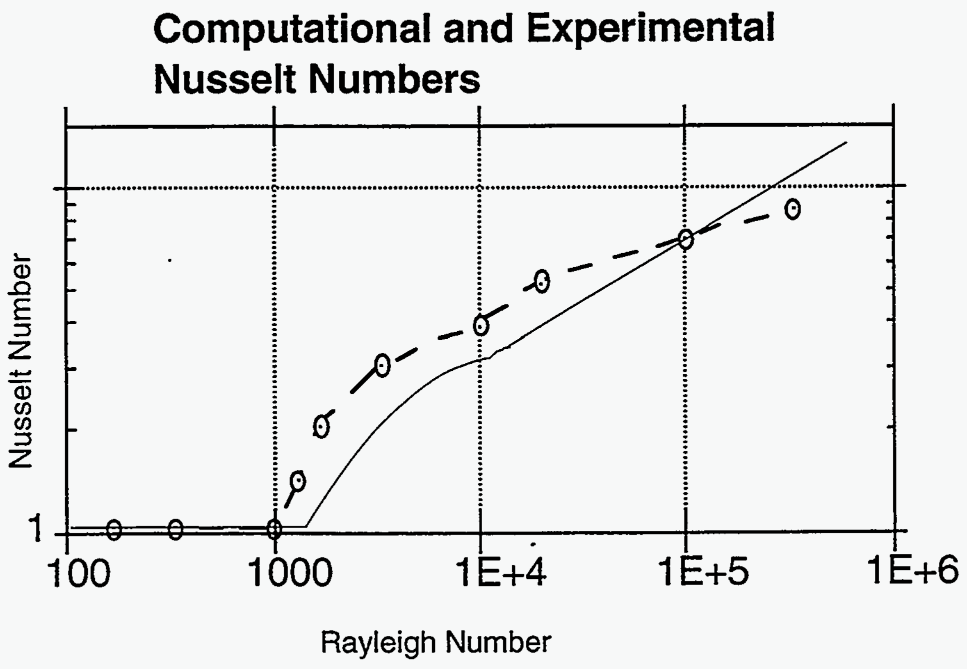


Figure VIII-26

In this graph, computational Nusselt numbers are plotted as circles and are connected with a dotted line. Experimental values appear as a solid line. Discrepancies between computational results and experimental results are most likely due to the coarseness of the mesh used in these simulations and the fact that our mesh is two-dimensional whereas the laboratory flow passage exists in three dimensions.

D. Two-Dimensional Compressible Flow

Our last topic in this chapter of additions to a two-dimensional Eulerian code is more an extension of previous concepts than an addition of a new element to an already existing code. An Eulerian two-dimensional compressible code is based on the same Navier-Stokes compressible flow equations that were used in the one-dimensional Eulerian code, but extended to two dimensions. In the one-dimensional code the equations were

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u}{\partial x} = 0 \tag{V-37}$$

$$\frac{\partial \rho u}{\partial t} + \frac{\partial \rho u^2}{\partial x} + \frac{\partial P}{\partial x} = 0 \tag{V-58}$$

$$\frac{\partial \rho I}{\partial t} + \frac{\partial \rho u I}{\partial x} + \frac{P \partial u}{\partial x} = 0\,. \tag{V-59}$$

In two-dimensions the system of equations becomes

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u}{\partial x} + \frac{\partial \rho v}{\partial y} = 0 \tag{VIII-34}$$

$$\frac{\partial \rho u}{\partial t} + \frac{\partial \rho u^2}{\partial x} + \frac{\partial \rho u v}{\partial y} + \frac{\partial P}{\partial x} = 0 \tag{VIII-35}$$

$$\frac{\partial \rho v}{\partial t} + \frac{\partial \rho u v}{\partial x} + \frac{\partial \rho v^2}{\partial y} + \frac{\partial P}{\partial y} = 0 \tag{VIII-36}$$

$$\frac{\partial \rho I}{\partial t} + \frac{\partial \rho u I}{\partial x} + \frac{\partial \rho v I}{\partial y} + P\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) = 0\,, \tag{VIII-37}$$

where P signifies total pressure $(p+q)$ rather than pressure per unit density. Note that the momentum equation becomes two equations when extended to two dimensions.

These equations are implemented in two dimensions much as they were in one. Six two-dimensional arrays are created: u, v, rho, sie, p, and q. These are located as in Fig. VIII-27.

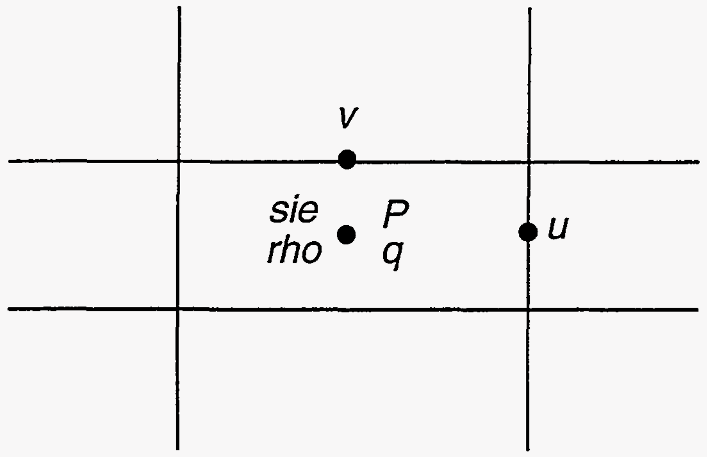


Figure VIII-27

These arrays are initialized to their desired values in an initialization routine. This routine also sets time counters as was done in previous programs.

Boundary conditions are set such that each wall of the system acts in one of three ways: as a rigid wall, a specified boundary, or an outflow boundary. Rigid walls are represented by setting a normal velocity of zero in all cells along the desired boundary. Specified boundaries are created by setting the normal velocity to some specified value as well as supplying sie and rho values for this flow. These values can be set according to the infinite-strength shock equations,

$$\rho = \frac{\gamma + 1}{\gamma - 1}\rho_0 \tag{IV-45}$$

and

$$I = \frac{u^2}{2}\rho_0 \,, \tag{V-61}$$

set to the same values as the initial rho's and sie's in the mesh, or set to some other values, such as those present in a rarefaction wave. Outflow boundaries are created by setting the velocities normal to a wall equal to the normal velocities directly before the wall (e.g., u(ibar,j) = u(ibar-1,j)). It is neither necessary nor desirable for outflow boundaries to be calculated as they were in the two-dimensional incompressible case, because in the compressible case, the amount of mass in the system is not a constant. Tangential boundary conditions are not necessary for the case in which the viscosity does not include shear stresses.

Obstacles can be included in a two-dimensional compressible code by using the same process that was discussed in Section A of this chapter. Once again, it is not necessary that tangential boundary conditions be included.

The two-dimensional Eulerian compressible transport equations are calculated similarly to those of the one-dimensional Eulerian code. This calculation is done explicitly with a routine that first calculates mass density, momentum density, and internal energy density and then uses these values to determine new values of rho, sie, u, v, p, and q. The equations that determine the new densities are finite-difference versions of Eqs. (VIII-34) through (VIII-37). These appear as

$$\rho_{i,j}^{n+1} = \rho_{i,j}^{n} - dt\left(\frac{(\rho \mathrm{u})_{\mathrm{i+1/2,j}}^{\mathrm{n}} - (\rho \mathrm{u})_{\mathrm{i-1/2,j}}^{\mathrm{n}}}{dx} + \frac{(\rho \mathrm{v})_{\mathrm{i,j+1/2}}^{\mathrm{n}} - (\rho \mathrm{v})_{\mathrm{i,j-1/2}}^{\mathrm{n}}}{dy}\right) \tag{VIII-38}$$

$$(\rho u)_{i+1/2,j}^{n+1} = (\rho u)_{i+1/2,j}^{n} - dt\left[\frac{(\rho \mathrm{u}^2)_{\mathrm{i+1,j}}^{\mathrm{n}} - (\rho \mathrm{u}^2)_{\mathrm{i,j}}^{\mathrm{n}}}{dx} + \frac{(\rho \mathrm{uv})_{\mathrm{i+1/2,j+1/2}}^{\mathrm{n}} - (\rho \mathrm{uv})_{\mathrm{i+1/2,j-1/2}}^{\mathrm{n}}}{dy} + \frac{p_{i+1,j} - p_{i,j}}{dx} + \frac{q_{i+1,j} - q_{i,j}}{dx}\right] \tag{VIII-39}$$

$$(\rho v)_{i,j+1/2}^{n+1} = (\rho v)_{i,j+1/2}^{n} - dt\left[\frac{(\rho \mathrm{uv})_{\mathrm{i+1/2,j+1/2}}^{\mathrm{n}} - (\rho \mathrm{uv})_{\mathrm{i-1/2,j+1/2}}^{\mathrm{n}}}{dx} + \frac{(\rho \mathrm{v}^2)_{\mathrm{i,j+1}}^{\mathrm{n}} - (\rho \mathrm{v}^2)_{\mathrm{i,j}}^{\mathrm{n}}}{dy} + \frac{p_{i,j+1} - p_{i,j}}{dy} + \frac{q_{i,j+1} - q_{i,j}}{dy}\right] \tag{VIII-40}$$

$$(\rho I)^{n+1}_{i,j} = (\rho I)^n_{i,j} - dt\left[\frac{(\rho \mathbf{uI})^{\mathbf{n}}_{\mathbf{i+1/2,j}} - (\rho \mathbf{uI})^{\mathbf{n}}_{\mathbf{i-1/2,j}}}{dx} + \frac{(\rho \mathbf{vI})^{\mathbf{n}}_{\mathbf{i,j+1/2}} - (\rho \mathbf{vI})^{\mathbf{n}}_{\mathbf{i,j-1/2}}}{dy} + (p+q)^n_{i,j}\left(\frac{u^n_{i+1/2,j} - u^n_{i-1/2,j}}{dx} + \frac{v^n_{i,j+1/2} - v^n_{i,j-1/2}}{dy}\right)\right], \tag{VIII-41}$$

where the bold terms are donor-cell terms, which can be calculated using either a series of if/then checks or a double-lookup technique.

If a double-lookup technique is used, integer arrays of ones and zeros must be set for six different circumstances: horizontal flux at right cell walls, vertical flux at cell tops, horizontal flux and vertical flux at cell centers, and horizontal and vertical flux at cell corners. These six different locations are illustrated in Fig. (VIII-28).

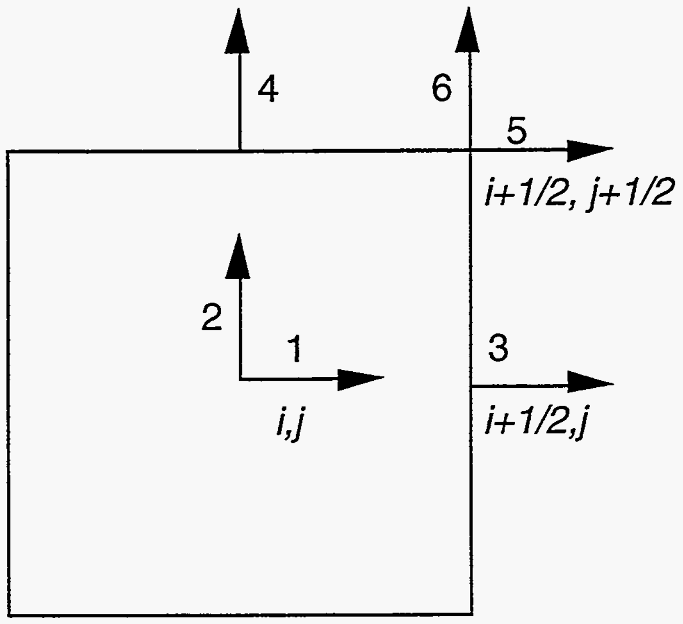


Figure VIII-28

Six arrays must therefore be defined for each of the six types of advective fluxes. We then have: idnr and jdnr, which represent fluxes in the i and j directions at the cell centers

(1 and 2 in the previous figure); idnrw, which represent flux across the right cell wall (3); jdnrw, which represents flux across the top of the cell (4); and idnrc and jdrnc, which represent fluxes in the i- and j-directions at the cell corners (5 and 6). These donor-cell arrays are calculated each time cycle and are used in the advective terms of the transport equations according to the position and direction of the flux that is being represented.

After quantity densities are calculated using either a series of if/then statements or a double-lookup technique, rho, sie, u, v are determined by setting the arrays equal to the appropriate density arrays divided by the mass densities, if necessary. Then p and q are determined from these arrays. In two dimensions the pressure equation is the same polytropic equation of state,

$$p_j^n = (\gamma - 1)\rho_j^n I_j^n \; ; \qquad \text{(IV-23)}$$

but the q equation is modified to respond to velocities in both the i- and j-directions. The two-dimensional q equation is

$$q_{i,j}^n = q_0 \rho_{i,j}^n C \sqrt{\frac{dx^2 + dy^2}{2}} \left(\frac{u_{i-1,2,j}^n - u_{i+1/2,j}^n}{dx} + \frac{v_{i,j-1/2}^n - v_{i,j+1/2}^n}{dy} \right) \qquad \text{if positive}$$

$$\text{or if negative} \qquad q_{i,j}^n = 0 \; . \qquad \text{(VIII-42)}$$

Note that if *dy* is equal to *dx* and there is no vertical motion of the fluid, this equation is identical to the one-dimensional q equation.

The output routine for a two-dimensional flow code can contain contour plots of density, internal energy, pressure, and viscous pressure. Streamlines are rarely used in the compressible case, however, because the divergence of the velocity is not equal to zero, and therefore Eq (VII-88) is not valid.

The sections in this program interact in much the same way as did the sections in the two-dimensional incompressible fluid code. The two-dimensional compressible code is structured as is illustrated in Fig. VIII-29.

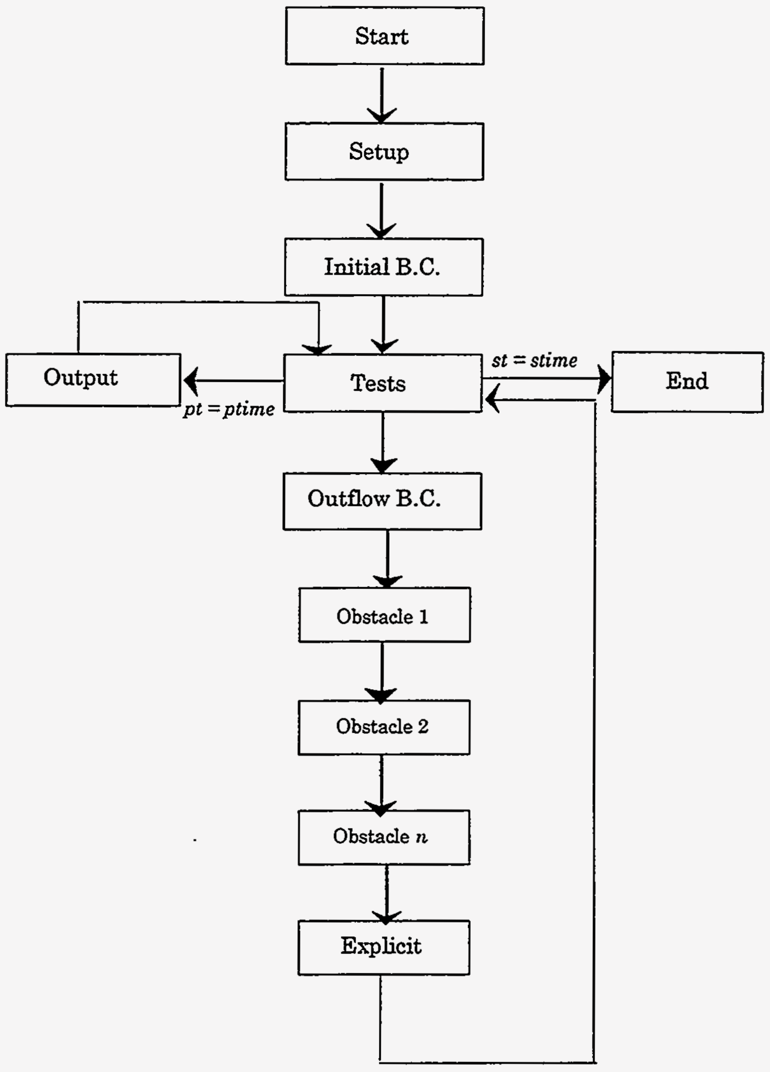
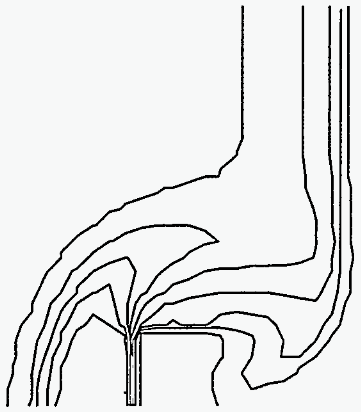


Figure VIII-29

E. Results of Two-Dimensional Compressible Flow

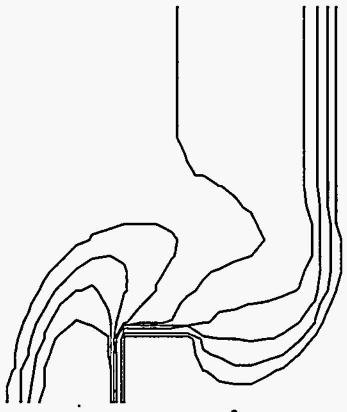
Using our two-dimensional compressible flow code, we can model a number of different problems. One simple problem that can be simulated is the piston problem that was discussed in Chapters IV and V. By specifying parameters such that there is variation of flow parameters in a single direction, our two-dimensional code can be used to obtain the same type of results we saw in our one-dimensional simulations. In fact, if they are set to the same parameters, both the two-dimensional code and the one-dimensional Eulerian compressible code should yield exactly the same results. Comparing these two codes is a good method for removing errors from the two-dimensional code.

The two-dimensional code can also extend the piston problem, such that the shock is moving down a flow passage with an obstacle in it. The following set of plots show the effect of an infinite strength shock moving down such a passage. Parameters are xlen= 30 (cm), ylen = 10 (cm), ibar = 75, jbar = 25, iobl = 21, iobr = 25, jobb = 1, jobt = 5, rho0 = 1 $\left(\frac{g}{cm^3}\right)$, sie0 = 0, gamma = 5/3, dt = .1(s), and ul = 1 (cm/s); rho's and sie's at the input boundary are defined using the equations of infinite strength shocks.



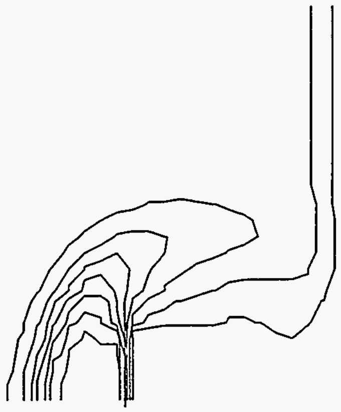
Densities at time 10 (s)

Figure VIII-30



Internal Energies at time 10 (s)

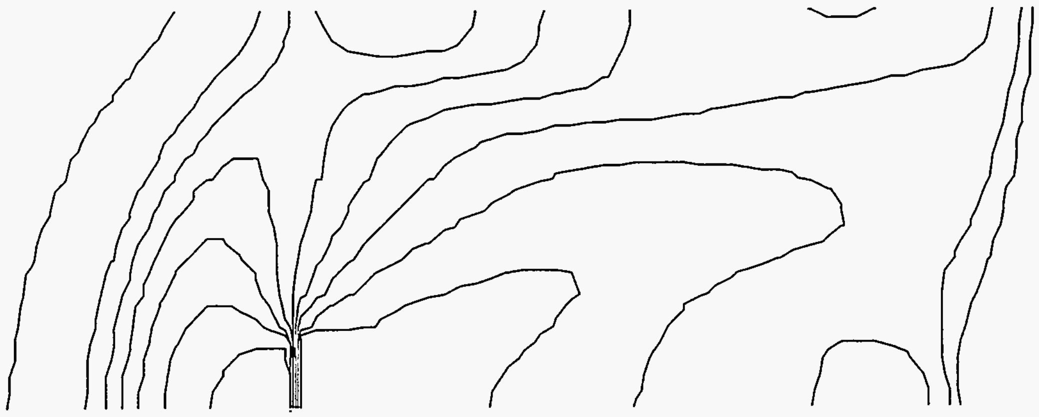
Figure VIII-31



Pressures at time 10 (s)

Figure VIII-32

Notice that while the upper portion of this shock is passing over the obstacle, the bottom portion of the shock is being reflected back towards the front of the flow passage. This reflected shock becomes further detached as time progresses, as can be seen in the following pressure plot.



Pressures at time 20 (s)

Figure VIII-33

In this plot we see that the reflected shock has moved upwards towards the top of the passage and leftwards towards the inlet. The shock is reflected off the top of the passage, and the shock and its reflection form a MACH STEM which will close off the incoming flow, choking the channel. A Mach stem is a shock that is formed between a shock that hits a wall and the resulting reflected shock. A Mach stem is always perpendicular to the wall. It is illustrated in the following figure.

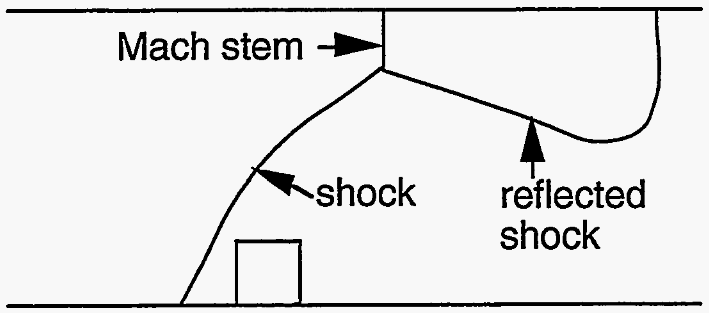
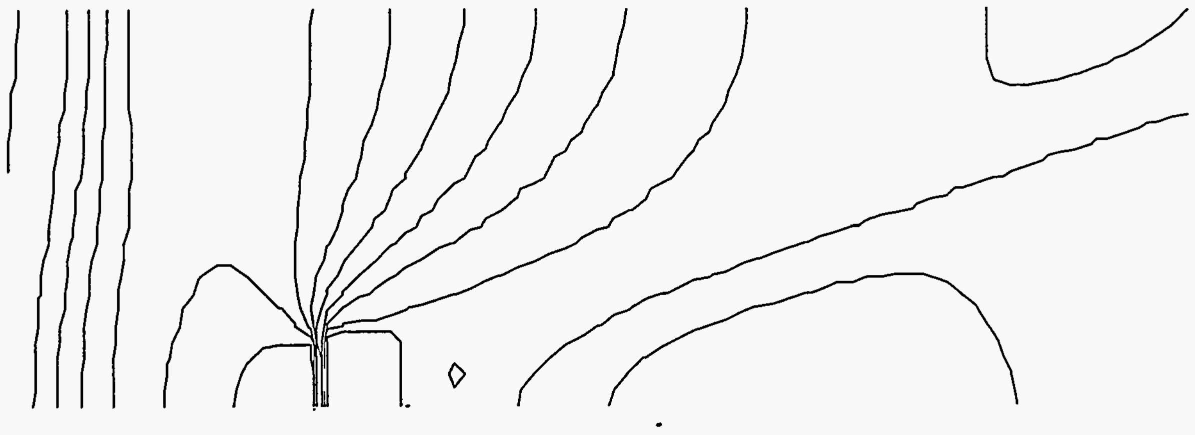


Figure VIII-34

The formation of a Mach stem is dependent on the angle at which the shock hits the top of the flow passage. Experiments have found that if the shock reaches the top wall at an angle of less than about 40 degrees, it will form a reflected shock but not a Mach stem and will eventually reach a steady state. If the shock reaches the wall at an angle greater than about 40 degrees; however, a Mach stem will form. In our plots we can scarcely see the Mach stem due to the coarseness of resolution; we can, however, see its effect of choking off the channel as is shown by the contour lines above the obstacle in the next figure.



Pressures at time 40 (s)

Figure VIII-35

A reflected shock that does not form a Mach stem can be generated by modifying our program such that we are no longer dealing with an reflected infinite strength shock moving over a stationary obstacle. Instead, we simulate the problem of an obstacle creating a shock as it moves through stationary flow. This simulation is done by setting the velocities, densities, and internal energies of the internal zones equal to the input values, as if one is traveling in the reference frame of the obstacle.

Setting up the code in this way allows us to examine flows at high MACH NUMBERS. The Mach number is the ratio of the velocity of a shock to the sound speed ahead of that shock:

$$M \equiv \frac{v}{c_{\text{sound}}} .$$

The lower the Mach number, the less intense the shock.

We can create shocks at any specified Mach number by using our equation for the sound speed,

$$c = \sqrt{\gamma(\gamma-1)I} . \qquad \text{(IV-26)}$$

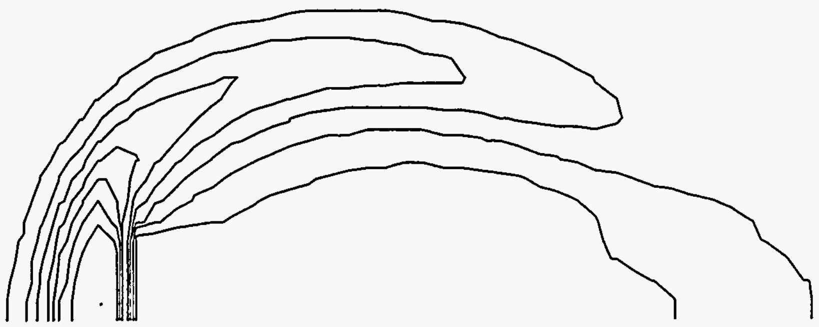
If we substitute this value into our definition of M, we obtain

$$M = \frac{v}{\sqrt{\gamma(\gamma-1)I}} . \qquad \text{(VIII-43)}$$

Solving for I, this becomes

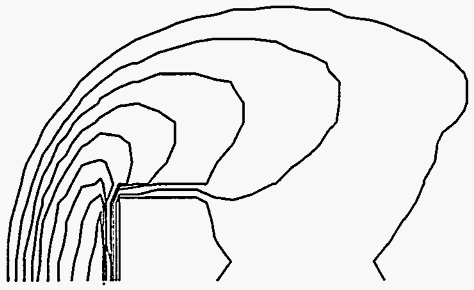
$$I = \frac{v^2}{M^2\gamma(\gamma-1)} . \qquad \text{(VIII-44)}$$

In the infinite-strength shock problem, input I's were defined as $\frac{1}{2}ul^2$, causing the Mach number of the flow to have a maximum of $\frac{1}{\sqrt{\gamma(\gamma-1)1/2}}$. In the moving obstacle case; however, I can be defined at any specified value, allowing for flows of any Mach number to be examined. For example by using Eq. (VIII-44), we can create a system with a Mach number of 10 by specifying an initial internal energy of 0.009 cm^2/s^2. Input rho's and sie's are set to rho0 and sie0 respectively, and initial velocities are set equal to the input velocity at the left. These parameters correspond to the simulation of an obstacle that is moving to the left. The results of this simulation appear in Figs. VIII-36 through VIII-41.



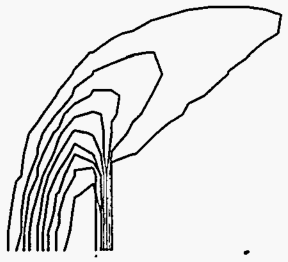
Densities at time 10 (s)

Figure VIII-36



Internal Energies at time 10 (s)

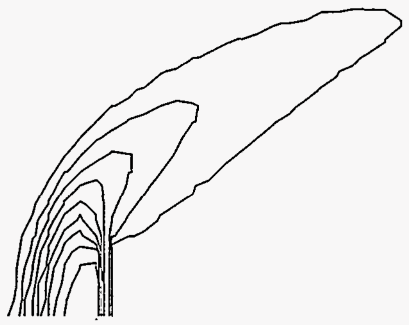
Figure VIII-37



Pressures at time 10 (s)

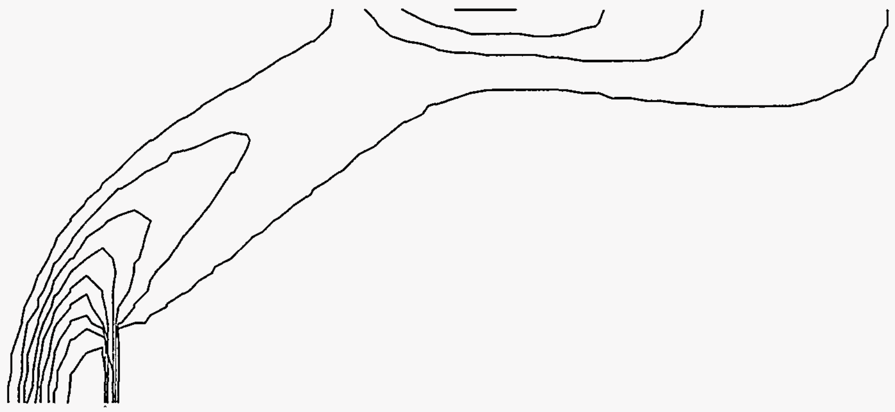
Figure VIII-38

Note that the shock formed in front of the obstacle is more swept back than the lower Mach number shock (see Figs. VIII-30 through VIII-32). This shock hits the top wall at angle less than 40 degrees and hence does not choke the channel. Instead it is reflected off the top wall and reaches a steady state. This reflection can be seen in the following three plots of pressure:



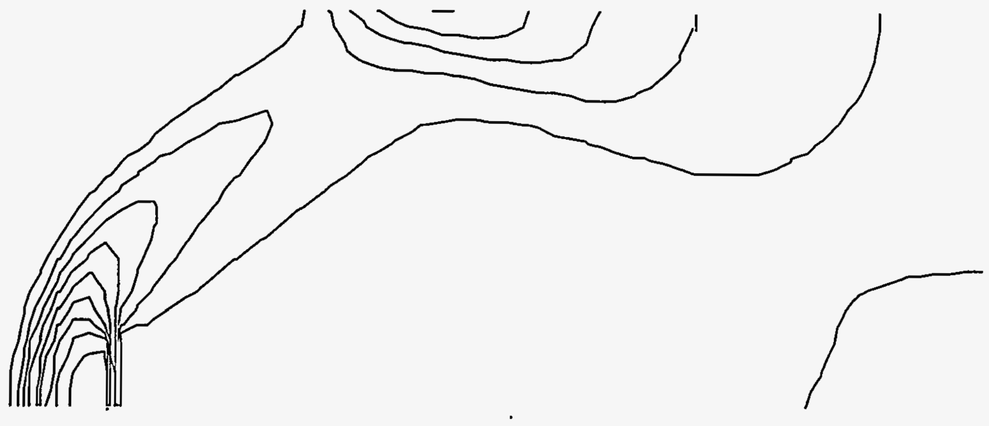
Pressures at time 20 (s)

Figure VIII-39



Pressures at time 40 (s)

Figure VIII-40



Pressures at time 100 (s)

Figure VIII-41

Note that the slight upward turn of the shock in the region near to the wall is an effect of the approximation of the actual shock by finite zones.

Another problem that can be modeled using a two-dimensional compressible flow code is the wedge problem, in which a shock passes over a wedge of a specified angle and the angle at which the shock reflects is measured. For this problem the difference between the angle of the shock and the wedge can be determined using the following equation, which can be found in LA-4700:

$$\tan(\theta - \alpha) = \frac{2 + (\gamma - 1)M_0^2 \sin^2\theta}{(\gamma + 1)M_0^2 \sin\theta\cos\theta}, \qquad \text{(VIII-45)}$$

where θ is the angle of the shock, α is the angle of the wedge, and M_0 is the Mach number of incoming flow. We can create a wedge of this type by making multiple calls to the object routine and stacking these objects in a triangle shape. These following set of figure are of a system with the parameters xlen= 5 (cm), ylen = 5 (cm), ibar = 50, jbar = 50, $M_0 = 10$, and with a wedge that begins in zone 11 and goes to the end of the mesh, ending at a height of 20 zones, and corresponding to an angle of 27 degrees. Figure VIII-42 is a contour plot of pressures in this system at a time of 50 seconds.

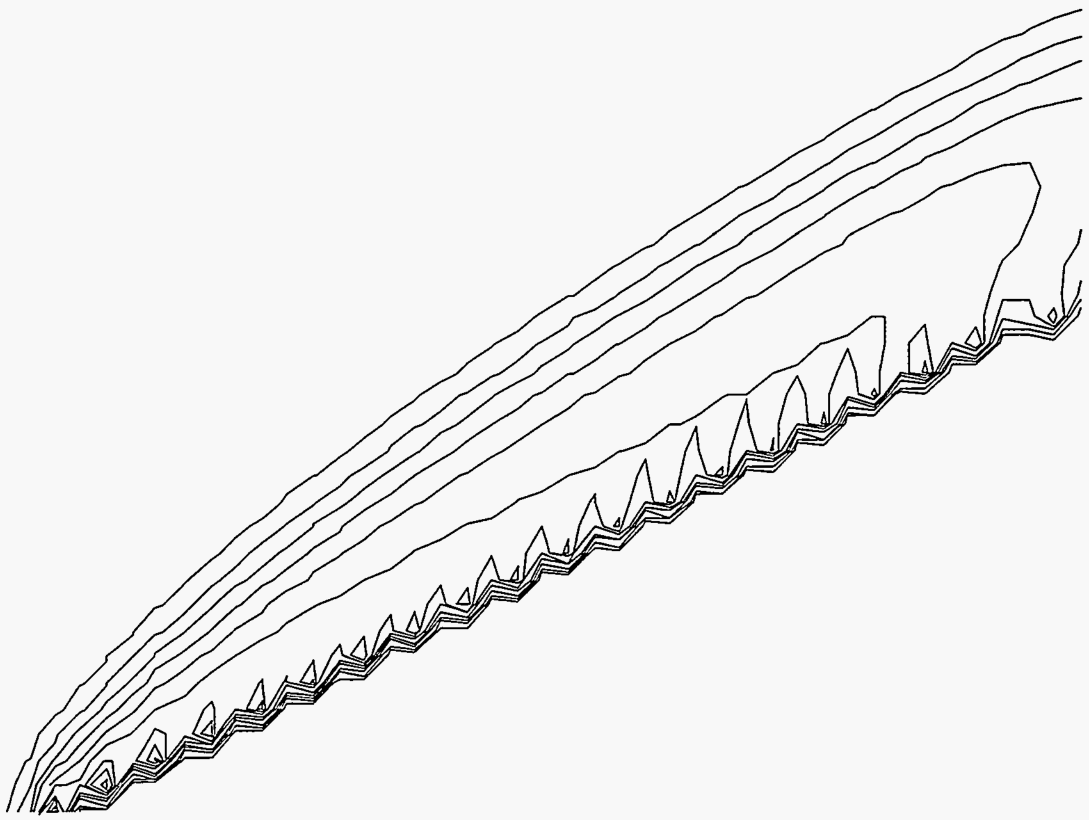


Figure VIII-42

In this set of circumstances the predicted angle of reflection is approximately 38 degrees. This angle is extremely close to the computationally calculated angle of 37 degrees.

If we now use the same set of parameters but for a wedge with a height of 10 zones (corresponding to an angle of 14 degrees), we obtain the following results. Again a contour plot of pressures is displayed.

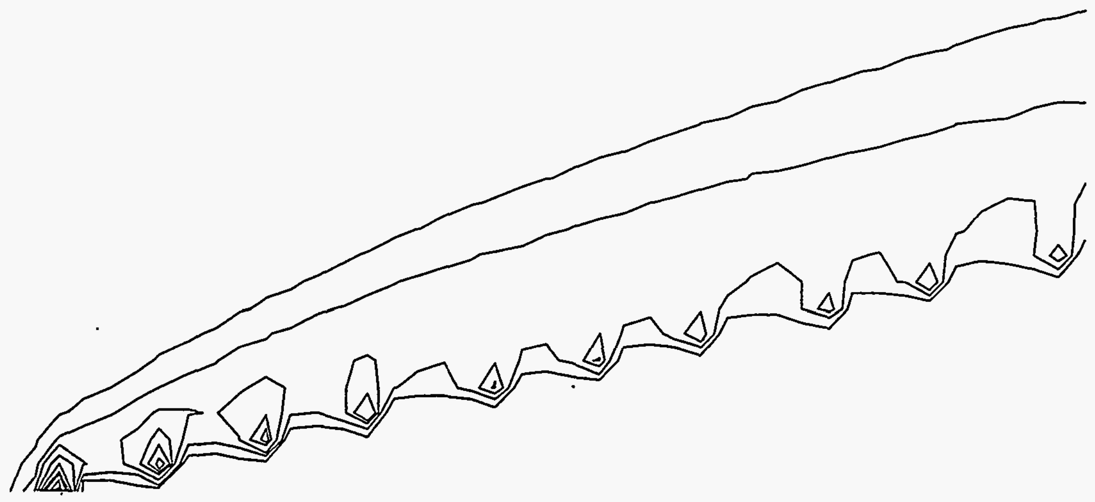


Figure VIII-43

For this set of circumstances, the analytical solution predicts an angle of 19 degrees whereas the computational solution yields an angle of approximately 21 degrees.

In this chapter, we examined three different modifications that can be made to a two-dimensional incompressible Eulerian flow code and discussed some additional problems that can be modeled with codes that include these modifications. In the next chapter, we will again be making an addition to our compressible fluid code, but this additional element will be different from the ones discussed in this chapter. Up to now, our equations have followed directly from mathematical manipulation of equations derived from basic physical principles, but this will not be the case in the next chapter. Instead, a complex mathematical model will be constructed to successfully approximate rigorously derived equations that are not able to be directly computed. Our turbulence transport equations will contain many of the properties, but will not directly represent, the computational calculation of the miniature fluctuations that are present in fluid flow of sufficiently high Reynolds number.

IX. TURBULENCE TRANSPORT

A. Tensor Notation

Before we discuss the equations of turbulence transport, it will be helpful to first examine a shorthand notation that can be used to express them. One such notation, CARTESIAN TENSOR NOTATION, is based on the idea that a system will act in the same manner regardless of the coordinates that are chosen to describe it. We can see this property in the momentum equations (VII-34 and VII-35), where an equation that expressed motion in the x-direction is coupled with a similar equation for motion in the y-direction; Eq. (VII-35) is simply Eq. (VII-34) with the x's exchanged with the y's and the u's exchanged with the v's. This same concept is also present in the heat equation, where the advective and viscous terms are symmetric with respect to the x- and y-directions.

Cartesian tensor notation makes use of subscripts to express the general directionality of a quantity without explicitly stating that it is in a particular x-, y-, or z-direction. This concept can be demonstrated by an example such as the two-dimensional heat-flow equation. In partial differential form, the heat-flow equation is Eq. (VIII-5):

$$\frac{\partial T}{\partial t} + \frac{\partial uT}{\partial x} + \frac{\partial vT}{\partial y} = \sigma \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right) . \tag{VIII-5}$$

Each of the u, v, x, and y terms in this equation is in reality a component of a vector, associated with either the x- or y-direction. If we let the symbol x represent a scalar (i.e., no direction) length and u represent a scalar speed, we can represent lengths and velocities in definite directions with subscripts. Lengths and velocities in the x-direction become x_x and u_x, and lengths and velocities in the y-direction become x_y and u_y. Equation (VIII-5) would then appear as follows:

$$\frac{\partial T}{\partial t} + \frac{\partial u_x T}{\partial x_x} + \frac{\partial u_y T}{\partial x_y} = \sigma \left(\frac{\partial^2 T}{\partial x_x^2} + \frac{\partial^2 T}{\partial x_y^2} \right) . \tag{IX-1}$$

If we further replace u_x with u_1, u_y with u_2, x_x with x_1, and x_y with x_2, the equation becomes

$$\frac{\partial T}{\partial t} + \frac{\partial u_1 T}{\partial x_1} + \frac{\partial u_2 T}{\partial x_2} = \sigma \left(\frac{\partial^2 T}{\partial x_1^2} + \frac{\partial^2 T}{\partial x_2^2} \right) . \tag{IX-2}$$

From this form, our equation can be rewritten using general subscripts (i, j, etc.) rather than specific numbered directions. This rewriting is done using the two major rules that govern equations written in Cartesian tensor notation.

The first of these rules is that any repeated index indicates the sum of all the elements in each of the available dimensions. For a three-dimensional system for example:

$$\frac{\partial \rho u_i}{\partial x_i} = \frac{\partial \rho u_1}{\partial x_1} + \frac{\partial \rho u_2}{\partial x_2} + \frac{\partial \rho u_3}{\partial x_3} . \tag{IX-3}$$

This rule allows us to condense combinations of terms such as $\frac{\partial u_1 T}{\partial x_1} + \frac{\partial u_2 T}{\partial x_2}$ into a single term, $\frac{\partial u_i T}{\partial x_i}$.

The second rule of Cartesian tensor notation is that any "free" (i.e., not repeated) index in one term must be the same in every other term. For example, in two dimensions,

$$\frac{\partial \rho u_i}{\partial t} + \frac{\partial \rho u_i u_j}{\partial x_j} = -\frac{\partial P}{\partial x_i} \tag{IX-4}$$

would become two equations:

$$\frac{\partial \rho u_1}{\partial t} + \frac{\partial \rho u_1^2}{\partial x_1} + \frac{\partial \rho u_1 u_2}{\partial x_2} = -\frac{\partial P}{\partial x_1} \tag{IX-5}$$

$$\frac{\partial \rho u_2}{\partial t} + \frac{\partial \rho u_1 u_2}{\partial x_1} + \frac{\partial \rho u_2^2}{\partial x_2} = -\frac{\partial P}{\partial x_2} . \tag{IX-6}$$

This rule of consistency of free indices coupled with the summation rule for repeated indices forms the basis of Cartesian tensor notation. We can use this notation to express Eq. (IX-2) as follows:

$$\frac{\partial T}{\partial t} + \frac{\partial u_i T}{\partial x_i} = \sigma \frac{\partial}{\partial x_j}\left(\frac{\partial T}{\partial x_j}\right) . \tag{IX-7}$$

In a similar manner, both momentum equations,

$$\frac{\partial u}{\partial t} + \frac{\partial u^2}{\partial x} + \frac{\partial uv}{\partial y} = -\frac{\partial P}{\partial x} + \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \tag{VII-34}$$

and

$$\frac{\partial v}{\partial t} + \frac{\partial uv}{\partial x} + \frac{\partial v^2}{\partial y} = -\frac{\partial P}{\partial y} + \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) , \tag{VII-35}$$

can be written as one equation:

$$\frac{\partial u_i}{\partial t} + \frac{\partial u_i u_j}{\partial x_j} = -\frac{\partial P}{\partial x_i} + \nu \frac{\partial}{\partial x_j}\left(\frac{\partial u_i}{\partial x_j}\right) . \qquad \text{(IX-8)}$$

Notice that in each of these tensor equations [Eqs. (IX-7) and (IX-8)], all the terms have the same number of free indices.

Equation (IX-7) has no free indices in any of its terms; it is made up of scalars. Such a scalar equation can be said to be made up of terms of TENSOR ORDER zero. Equation (IX-8), on the other hand, has one free index in each of its terms and therefore is made up of vectors, or first-order tensors. As the number of free indices increases in an equation, its tensor order similarly increases. Tensors of any order may exist, but all the terms of a given equation must be of the same order.

Cartesian tensor notation will be useful when discussing the equations of turbulence transport. It greatly increases the clarity of these equations and simplifies the notation in the complex derivations that are used to generate turbulence-transport models.

B. Turbulence Transport and $K - \epsilon$ Models

Before we examine the equations of turbulence transport, we must first define what we mean by turbulence. Any flow can be divided into steady and fluctuating parts. For our purposes, we will define TURBULENCE as the fluctuating part of that flow. The underlying average velocities over which these turbulent fluctuations exist will be called the MEAN FLOW.

Precisely what we define to be the mean flow and what we consider to be turbulence is a matter of choice. In the Karman vortex street problem, for example, there exists at low viscosities a fluctuating stream moving to the right. This rightward flow could be considered the mean flow, whereas the up-and-down oscillations could be called turbulence. But another definition could be chosen: both the rightward velocity and the up-and-down motion could be considered part of the mean flow, whereas the miniature fluctuations that are present within the up-and-down stream could be labeled as turbulence. Both of these definitions are valid. Flow is divided into mean flow and turbulence, and the threshold between these two types of flow is set at any arbitrary resolution.

In our simulations of turbulence, we will consider this threshold to be at the level of resolution of the mesh. Flow that can be resolved through the use of u's and v's will be mean flow, and fluctuations that are smaller than the area of a cell will be considered to be turbulence. In principle, however, our turbulence equations can resolve fluctuations even greater than the resolution of the mesh. Although this is rarely desired, it is interesting to note that turbulence equations can be used to represent fluctuations at any scale.

Our turbulence model will not resolve the turbulent fluctuations themselves but rather the TURBULENT KINETIC ENERGY per unit mass, the amount of kinetic energy per unit mass present in the turbulent fluctuations. This two-dimensional array will be defined at the cell centers and designated by a K. A variable, ϵ, will also be calculated over the mesh to represent the rate of dissipation of turbulent kinetic energy in different subregions in the fluid. The resulting placement of variables on the mesh appears below:

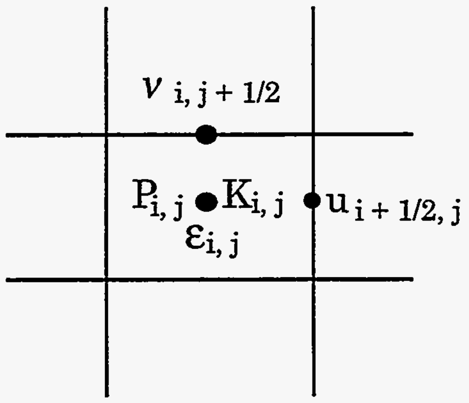


Figure IX-1

The method of simulating turbulence that makes use of these variables is known as the $K-\epsilon$ TURBULENCE MODEL, named after the two variables in its transport equations.

In order to derive an expression for the effect of K and ϵ on the transport of momentum, we begin with the momentum equation as expressed in tensor form (IX-8).

Each velocity and pressure in this equation is made up of a mean value and a fluctuating value:

$$u_i \equiv \bar{u}_i + u_i'$$

$$P \equiv \bar{P} + P' ,$$

where $\bar{u}_i$ and $\bar{P}$ represent the average parts, and u_i' and P' represent the fluctuating parts of u_i and P. Substituting these definitions into Eq. (IX-8) gives us

$$\frac{\partial(\bar{u}_i + u_i')}{\partial t} + \frac{\partial(\bar{u}_i + u_i')(\bar{u}_j + u_j')}{\partial x_j} = -\frac{\partial(\bar{P} + P')}{\partial x_i} + \nu\left(\frac{\partial^2(\bar{u}_i + u_i')}{\partial x_j^2}\right) \qquad \text{(IX-9)}$$

or

$$\frac{\partial \bar{u}_i}{\partial t} + \frac{\partial u_i'}{\partial t} + \frac{\partial \bar{u}_i \bar{u}_j}{\partial x_j} + \frac{\partial \bar{u}_i u_j'}{\partial x_j} + \frac{\partial u_i' \bar{u}_j}{\partial x_j} + \frac{\partial u_i' u_j'}{\partial x_j} = -\frac{\partial \bar{P}}{\partial x_i} - \frac{\partial P'}{\partial x_i} + \nu\left(\frac{\partial^2 \bar{u}_i}{\partial x_j^2} + \frac{\partial^2 u_i'}{\partial x_j^2}\right) . \qquad \text{(IX-10)}$$

Because the turbulent fluctuations are symmetric about the mean flow, the time averages of the fluctuations $(\bar{u}_i', \bar{u}_j', \bar{P}')$ are equal to zero. Therefore, all terms that contain a single fluctuating factor are also equal to zero when averaged, and the time average of Eq. (IX-10) can be written as

$$\frac{\partial \bar{u}_i}{\partial t} + \frac{\partial \bar{u}_i \bar{u}_j}{\partial x_j} + \frac{\partial \overline{u_i' u_j'}}{\partial x_j} = -\frac{\partial \bar{P}}{\partial x_i} + \nu\left(\frac{\partial^2 \bar{u}_i}{\partial x_j^2}\right) . \qquad \text{(IX-11)}$$

This equation is almost identical to the original momentum equation, (IX-8) but contains an additional term. While u_j' and u_i' are both equal to zero when averaged over time, the time average of their product $(\overline{u_i' u_j'})$ is not equal to zero, resulting in the $\frac{\partial \overline{u_i' u_j'}}{\partial x_j}$ term in Eq. (IX-11). $(\overline{u_i' u_j'})$ is called the REYNOLDS STRESS TENSOR, abbreviated as $R_{i,j}$. This second-order tensor represents the effect of turbulence on the mean flow.

Computationally, this tensor is approximated by calculating a turbulent viscosity that is added to the molecular viscosity to represent the total viscous forces on the fluid. To make this approximation, we substitute the variable $R_{i,j}$ for $(\overline{u_i' u_j'})$. Equation (IX-11) becomes

$$\frac{\partial \bar{u}_i}{\partial t} + \frac{\partial \bar{u}_i \bar{u}_j}{\partial x_j} = -\frac{\partial \bar{P}}{\partial x_i} + \nu\left(\frac{\partial^2 \bar{u}_i}{\partial x_j^2}\right) - \frac{\partial R_{i,j}}{\partial x_j} \qquad \text{(IX-12)}$$

which is a somewhat limited case of the fluid-flow equation. It is valid only when ν is a constant and the fluid is incompressible. This equation is more properly written as

$$\frac{\partial \bar{u}_i}{\partial t} + \frac{\partial \bar{u}_i \bar{u}_j}{\partial x_j} = -\frac{\partial \bar{P}}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\nu \frac{\partial \bar{u}_i}{\partial x_j} + \nu \frac{\partial \bar{u}_j}{\partial x_i} - R_{i,j}\right) . \tag{IX-13}$$

Using this general equation, we then make the approximation that

$$R_{i,j} \approx -\nu_t \left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i}\right) + \frac{2}{3} K \delta_{i,j} \tag{IX-14}$$

where ν_t is the TURBULENT VISCOSITY, a viscosity that results from the presence of turbulence in the system, K is the turbulent kinetic energy, and $\delta_{i,j}$ is the KRONECKER SYMBOL which is one if i equals j, and zero otherwise. The MODELING of $R_{i,j}$ in this manner is a somewhat arbitrary decision. It is made because more rigorous representations of this tensor are unnecessary for the accuracy to which we desire a solution. This representation is chosen because it has the correct dimensions, is of the right tensor order, and has been experimentally demonstrated to be reasonably accurate. This model is known as the BOUSSINESQ APPROXIMATION for the Reynolds stress tensor. It is not to be confused with the Boussinesq approximation for the momentum equation.

Because the $\delta_{i,j}$ term is absorbed in the pressure term, the Boussinesq approximation allows us to rewrite Eq. (IX-13) as

$$\frac{\partial \bar{u}_i}{\partial t} + \frac{\partial \bar{u}_i \bar{u}_j}{\partial x_j} = -\frac{\partial \bar{P}}{\partial x_i} + (\nu + \nu_t)\left(\frac{\partial^2 \bar{u}_i}{\partial x_j^2}\right) . \tag{IX-15}$$

We calculate ν_t by using the variables K and ϵ. In this equation K, the turbulent kinetic energy, has units of energy per unit mass: $\frac{\text{length}^2}{\text{time}^2}$. ϵ is the rate of dissipation of K, and its units are those of energy per unit mass per unit time: $\frac{\text{length}^2}{\text{time}^3}$. Because viscosity has units of $\frac{\text{length}^2}{\text{time}}$, turbulent viscosity can be synthesized dimensionally as

$$C_\nu \frac{K^2}{\epsilon} ,$$

where C_ν is a constant that has been experimentally determined to be about 0.09. Our equation for ν_t at a point (i,j) on the mesh is then

$$\nu_{t(i,j)} = 0.09 \frac{K_{i,j}^2}{\epsilon_{i,j}} \, . \tag{IX-16}$$

Having related K and ϵ to the momentum equation, we must also derive transport equations for these two quantities. K is calculated by relating it to the Reynolds stress tensor, and using the classical equation for kinetic energy:

$$E_{\text{kinetic}} = \frac{1}{2} m v^2 \, . \tag{IX-17}$$

Turbulent kinetic energy, which is expressed as kinetic energy per unit mass, is then

$$K = \frac{1}{2} \overline{u_i' u_i'} \, , \tag{IX-18}$$

which can also be written as

$$K = \frac{1}{2} R_{i,i} \, . \tag{IX-19}$$

Using this equation for K in terms of the Reynolds tensor, we calculate K by first deriving an equation for $R_{i,j}$.

This derivation will not be carried out in detail in this work, but a short overview is included to give some insight into the process: First, Eq. (IX-9) is multiplied by u_j' to obtain an equation in terms of $u_j' \frac{\partial u_i'}{\partial t}$. Then Eq. (IX-9) is written in terms of u_j' and multiplied by u_i' to obtain an equation in terms of $u_i' \frac{\partial u_j'}{\partial t}$. These two equations are added and averaged to obtain an equation in terms of $\overline{u_j' \frac{\partial u_i'}{\partial t}} + \overline{u_i' \frac{\partial u_j'}{\partial t}}$. In this step, all equations containing a single fluctuating term become zero. Then, by the chain rule [Eq. (VII-53)], $\overline{u_j' \frac{\partial u_i'}{\partial t}} + \overline{u_i' \frac{\partial u_j'}{\partial t}}$ becomes $\frac{\partial \overline{u_i' u_j'}}{\partial t}$, which is the time derivative of the Reynolds stress tensor, $\frac{\partial R_{i,j}}{\partial t}$. This equation is contracted to be in terms of $R_{i,i}$ and divided by two. One is then left with a transport equation for K which contains some terms that cannot be computationally represented using only K and ϵ. These terms are then modeled, resulting in a final equation for the transport of K:

$$\underbrace{\frac{\partial K}{\partial t}}_{I} + \underbrace{\frac{\partial K u_i}{\partial x_i}}_{II} = \underbrace{\frac{\partial}{\partial x_k}\left((\nu + \nu_t)\frac{\partial K}{\partial x_k}\right)}_{III} + \underbrace{(\nu + \nu_t)\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\frac{\partial u_i}{\partial x_j}}_{IV} - \underbrace{\quad \epsilon \quad}_{V} \, . \tag{IX-20}$$

In this equation the term denoted by I represents the time rate of change of K, II represents advection, III represents diffusion of turbulent energy, IV represents the generation of turbulence by SHEAR FORCES (forces similar to friction that are caused by flows at different velocities rubbing against each other), and V (ϵ) represents the dissipation of turbulence.

A transport equation for ϵ is "derived" by modeling an equation after the transport equation for K. The ϵ transport equation is

$$\underbrace{\frac{\partial \epsilon}{\partial t}}_{I} + \underbrace{\frac{\partial u_i \epsilon}{\partial x_i}}_{II} = \underbrace{\frac{\partial}{\partial x_k}\left(\frac{\nu + \nu_t}{\sigma_\epsilon}\frac{\partial \epsilon}{\partial x_k}\right)}_{III} + \underbrace{C_{\epsilon 1}\frac{\epsilon}{K}(\nu + \nu_t)\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\frac{\partial u_i}{\partial x_j}}_{IV} \underbrace{- C_{\epsilon 2}\frac{\epsilon^2}{K}}_{V}, \tag{IX-21}$$

where σ_ϵ, $C_{\epsilon 1}$, $C_{\epsilon 2}$ are constants which have been determined as a result of experimentation. Typically,

$$\sigma_\epsilon \approx 1.3$$

$$C_{\epsilon 1} \approx 1.55$$

$$C_{\epsilon 2} \approx 2.0\ .$$

Equation (IX-21), like Eq. (IX-20) is made up of a rate of change term (I), an advection term (II), a diffusion term (III), a generation term (IV), and a dissipation or dampening term (V).

These two transport equations [Eqs. (IX-20) and (IX-21)], combined with the equation for turbulence viscosity [Eq. (IX-16)] make up the $K - \epsilon$ turbulence model. We will employ this model in a two-dimensional fluid-flow code to compute turbulence transport computationally.

C. Computational Implementation of the $K - \epsilon$ Turbulence-Transport Model

The $K - \epsilon$ turbulence model is implemented by creating two new arrays: K and eps. The values of these arrays are specified at cell centers as is illustrated in Fig. IX-1 in Section B. Both two-dimensional arrays are initialized in the setup procedure. A typical initial value for K is one-tenth of the energy per mass of the mean flow. In the Karman vortex street problem, for example, initial K's might be assigned such that

$$K0 = \left(\frac{1}{10}\right)\left(\frac{1}{2}ul^2\right) . \tag{IX-22}$$

A typical value of eps0, the initial value of the eps array, is $K^{3/2}$ divided by some characteristic size of the turbulence of the system. For the Karman vortex problem, a typical turbulence size is half the width of the obstacle. For this problem, the ϵ array is then initialized to

$$eps0 = \frac{K0^{3/2}}{(0.5)(h_{obs})} . \tag{IX-23}$$

In addition to these two arrays, three other arrays are created for the turbulence viscosity at the cell centers, the top of the cells, and the right cell walls. These are configured as in Fig. IX-2.

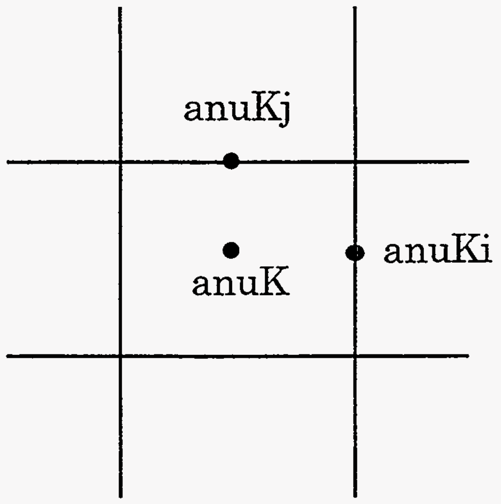


Figure IX-2

These arrays need not be initialized because they will be calculated before they are ever used in the program.

The u, v, ubar, vbar, and pressure arrays are implemented as they were in our other two-dimensional simulations with the exception of the total viscosity (molecular plus turbulent) being used rather than simply the molecular viscosity. Our turbulence transport code will not contain a temperature array or equations of heat transport.

Besides the initialization of the K and eps arrays in the setup procedure, our turbulence transport code contains three major sections that did not exist in the basic two-dimensional incompressible code: the implementation of turbulent boundary conditions, the calculation of turbulent viscosities, and the calculation of the K and ϵ equations themselves. The resulting code is structured as in Fig IX-3.

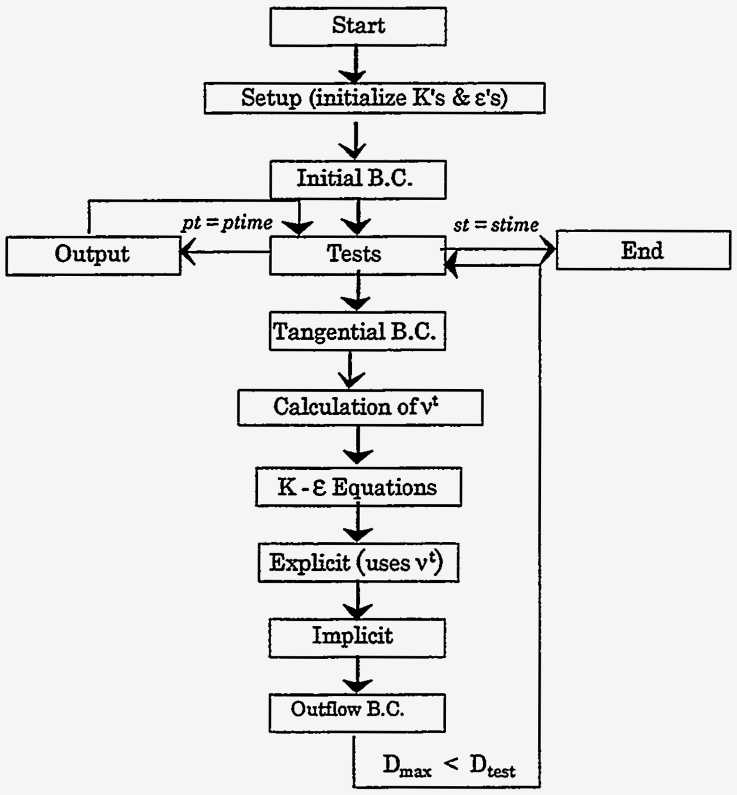


Figure IX-3

The turbulence boundary conditions will be reflective, with the values of K and eps at the ghost zones equal to the values at the real zones directly adjacent to them. This method is only an approximation of the true effect of a wall on the turbulence in a system. Much more accurate boundary conditions exist, using WALL FUNCTIONS, that carefully calculate more appropriate values for the ghost zones. Even these functions have their limitations, however; and for turbulence to be completely modeled at a wall, each component of the Reynolds stress tensor must be calculated separately rather than the overall turbulent kinetic energy (K). Because our simulation is not overly concerned with turbulence at the walls of the system, neither of these methods is necessary. A simple reflective condition where

$$K_{\text{ghost}} = K_{\text{real}} \tag{IX-24}$$

and

$$\epsilon_{\text{ghost}} = \epsilon_{\text{real}} \tag{IX-25}$$

will prove sufficiently accurate for our immediate purposes. Two loops that carry out these calculations along the top and bottom and along the edges of the mesh comprise the turbulent boundary condition routine.

The routine to compute the total viscosities assigns values to the three turbulence viscosity arrays (anuk, anuki, and anukj) by using K and eps values at the desired positions to calculate ν_t as in Eq. (IX-16) and adding the molecular viscosity. K's and ϵ' s at cell walls are calculated by averaging. The three equations for the total viscosities are

$$(\text{anuK})_{i,j} = \frac{K_{i,j}^{\frac{3}{2}}}{\epsilon_{i,j}} + \nu_{\text{molecular}} \tag{IX-26}$$

$$(\text{anuKi})_{i+1/2,j} = \frac{\left(\frac{K_{i,j}+K_{i+1,j}}{2}\right)^{\frac{3}{2}}}{\left(\frac{\epsilon_{i,j}+\epsilon_{i+1,j}}{2}\right)} + \nu_{\text{molecular}} \tag{IX-27}$$

$$(\text{anuKj})_{i,j+1/2} = \frac{\left(\frac{K_{i,j}+K_{i,j+1}}{2}\right)^{\frac{3}{2}}}{\left(\frac{\epsilon_{i,j}+\epsilon_{i,j+1}}{2}\right)} + \nu_{\text{molecular}} \cdot \tag{IX-28}$$

These should be carried out every time step, just before the program enters the routine for turbulence transport.

The turbulence transport equations are calculated by computing Eqs. (IX-20) and (IX-21) at every point on the mesh. For this calculation local arrays of variables are employed to represent each term in the equations. Equations (IX-20) and (IX-21) are then written as

$$\frac{\partial K}{\partial t} = -Kt1 + Kt2 + Kt3 - \epsilon \tag{IX-29}$$

and

$$\frac{\partial \epsilon}{\partial t} = -\epsilon t1 + \epsilon t2 + \epsilon t3 - \epsilon t4\,, \tag{IX-30}$$

where $Kt1 \equiv \frac{\partial K u_i}{\partial x_i}$, $Kt2 \equiv \frac{\partial}{\partial x_k}\left((\nu + \nu_t)\frac{\partial K}{\partial x_k}\right)$, $Kt3 \equiv (\nu+\nu_t)\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\frac{\partial u_i}{\partial x_j}$, $\epsilon t1 \equiv \frac{\partial \epsilon u_i}{\partial x_i}$, $\epsilon t2 \equiv \frac{\partial}{\partial x_k}\left(\frac{\nu+\nu_t}{\sigma_\epsilon}\frac{\partial K}{\partial x_k}\right)$, $\epsilon t3 \equiv C_{\epsilon 1}\frac{\epsilon}{K}(\nu+\nu_t)\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\frac{\partial u_i}{\partial x_j}$, and $\epsilon t4 \equiv C_{\epsilon 2}\frac{\epsilon^2}{K}$.

K and ϵ are calculated using the finite-difference versions of Eqs. (IX-29) and (IX-30):

$$K^{n+1}_{i,j} = K^n_{i,j} + dt(-Kt1^n_{i,j} + Kt2^n_{i,j} + Kt3^n_{i,j} - \epsilon^n_{i,j}) \tag{IX-31}$$

and

$$\epsilon^{n+1}_{i,j} = \epsilon^n_{i,j} + dt(-\epsilon t1^n_{i,j} + \epsilon t2^n_{i,j} + \epsilon t3^n_{i,j} - \epsilon t4^n_{i,j})\,, \tag{IX-32}$$

which leaves us the issue of how each of the terms in these equations is calculated.

The advective terms in these equations ($Kt1$ and $\epsilon t1$) are calculated by first computing donor-cell arrays as was done in the two-dimensional heat-flow equation: Two arrays, *idnr* and *jdnr*, are calculated. *idnr* is calculated from the u velocities at position $i+1/2, j$ and is zero if the flow is from left to right and one if the flow is from right to left. *jdnr* is calculated from the v velocities at position $i, j+1/2$ and is zero if the flow is upwards and one if the flow is downwards. These two arrays are used to calculate donor cell in a double-lookup fashion.

In finite-difference form $Kt1$ and $\epsilon t1$ are

$$Kt1^n_{i,j} = \frac{(uK)^n_{i+1/2,j} - (uK)^n_{i-1/2,j}}{dx} + \frac{(vK)^n_{i,j+1/2} - (vK)^n_{i,j-1/2}}{dy} \tag{IX-33}$$

and

$$\epsilon t1^n_{i,j} = \frac{(u\epsilon)^n_{i+1/2,j} - (u\epsilon)^n_{i-1/2,j}}{dx} + \frac{(v\epsilon)^n_{i,j+1/2} - (v\epsilon)^n_{i,j-1/2}}{dy}\,, \tag{IX-34}$$

where each K and ϵ at a cell wall is calculated using the donor-cell technique. In code form these equations appear as

```
Kt1(i,j) =(u(i,j)*K(i+idnr(i,j),j) -
& u(i-1,j)*K(i-1+idnr(i-1,j),j))/dx +
& (v(i,j)*K(i,j+jdnr(i,j)) -
& v(i-1,j)*K(i,j-1+jdnr(i-1,j))/dy

Epst1(i,j) =( u(i,j)*Eps(i+idnr(i,j),j) -
& u(i-1,j)*Eps(i-1+idnr(i-1,j),j) )/dx +
& (v(i,j)*Eps(i,j+jdnr(i,j)) -
& v(i-1,j)*Eps(i,j-1+jdnr(i-1,j) )/dy
```

$Kt2$ and $\epsilon t2$ are written in finite-difference form as

$$Kt2^n_{i,j} = \frac{(\nu + \nu_{t(i+1/2,j)})(K_{i+1,j} - K_{i,j}) - (\nu + \nu_{t(i-1/2,j)})(K_{i,j} - K_{i-1,j})}{dx^2} + \frac{(\nu + \nu_{t(i,j+1/2)})(K_{i,j+1} - K_{i,j}) - (\nu + \nu_{t(i,j-1/2)})(K_{i,j} - K_{i,j-1})}{dy^2} \tag{IX-35}$$

$$\epsilon t2^n_{i,j} = \frac{1}{\sigma_\epsilon}\left[\frac{(\nu + \nu_{t(i+1/2,j)})(\epsilon_{i+1,j} - \epsilon_{i,j}) - (\nu + \nu_{t(i-1/2,j)})(\epsilon_{i,j} - \epsilon_{i-1,j})}{dx^2} + \frac{(\nu + \nu_{t(i,j+1/2)})\left(\epsilon_{i,j+1} - \epsilon_{i,j}\right) - (\nu + \nu_{t(i,j-1/2)})\left(\epsilon_{i,j} - \epsilon_{i,j-1}\right)}{dy^2}\right]. \tag{IX-36}$$

In code form, this equation is written as follows:

```
Kt2(i,j)=((anuki(i,j)*(K(i+1,j)-K(i,j))
& + anuki(i-1,j)*(K(i-1,j)-K(i,j)))/(dx*dx))
& + ((anukj(i,j)*(K(i,j+1)-K(i,j))
& + anukj(i,j-1)*(K(i,j-1)-K(i,j)))/(dy*dy))

Epst2(i,j)=((anuki(i,j)*(Eps(i+1,j)-Eps(i,j))
& + anuki(i-1,j)*(Eps(i-1,j)-Eps(i,j)))/(dx*dx))
& + ((anukj(i,j)*(Eps(i,j+1)-Eps(i,j))
& + anukj(i,j-1)*(Eps(i,j-1)-Eps(i,j)))/(dy*dy))
& * (1/sige)
```

$Kt3$ is calculated by first expanding to obtain

$$Kt3 = (\nu + \nu_t)\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\frac{\partial u_i}{\partial x_j} = (\nu + \nu_t)\left[\left(\frac{\partial u}{\partial x} + \frac{\partial u}{\partial x}\right)\frac{\partial u}{\partial x} + \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)\frac{\partial u}{\partial y} + \left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}\right)\frac{\partial v}{\partial x} + \left(\frac{\partial v}{\partial y} + \frac{\partial v}{\partial y}\right)\frac{\partial v}{\partial y}\right]. \tag{IX-37}$$

This reduces to

$$Kt3 = (\nu + \nu_t)\left[2\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)^2 + 2\left(\frac{\partial v}{\partial y}\right)^2\right]. \tag{IX-38}$$

In finite-difference form, this equation appears as

$$Kt3^n_{i,j} = (\nu + \nu^n_{t(i,j)})\left[2\frac{\left(u^n_{i+1/2,j} - u^n_{i-1/2,j}\right)^2}{dx^2} + \left(\frac{u^n_{i,j+1} - u^n_{i,j-1}}{2\,dy} + \frac{v^n_{i+1,j} - v^n_{i-1,j}}{2\,dx}\right)^2 + 2\frac{\left(v^n_{i,j+1/2} - v^n_{i,j-1/2}\right)^2}{dy^2}\right], \tag{IX-39}$$

where u at i and v at j are the average of $u_{i+1/2}$ and $u_{i-1/2}$ and the average of $v_{j+1/2}$ and $v_{j-1/2}$ respectively. Note that the middle term involves differences taken across a distance of $2\,dx$ and $2\,ddy$, due to the positions at which u and v are defined. In code form Eq. (IX-39) appears as the the following:

```
  Kt3(i,j)= anuK(i,j) * ((u(i,j)-u(i-1,j))**2)/(dx*dx) +
& ((v(i,j)-v(i,j-1))**2)/(dy*dy) +
& (((u(i,j+1)+u(i-1,j+1)-u(i,j-1)-u(i-1,j-1))/(4*dy)) +
& ((v(i+1,j)+v(i+1,j-1)-v(i-1,j)-v(i-1,j-1))/(4*dx)))**2
```

$\epsilon t3$ is calculated using $Kt3$, where

$$\epsilon t3^n_{i,j} = C_{\epsilon 1}\frac{\epsilon^n_{i,j}}{K^n_{i,j}}Kt3^n_{i,j}\,. \tag{IX-40}$$

In code form, this equation is

```
  Epst3(i,j) = ce1 * (Eps(i,j)/K(i,j)) * Kt3(i,j)
```

$\epsilon t4$ is simply calculated as

$$\epsilon t4^n_{i,j} = C_{\epsilon 2}\frac{\left(\epsilon^n_{i,j}\right)^2}{K^n_{i,j}} , \tag{IX-41}$$

or, in code form,

```
Epst4(i,j) = ce2 * Eps(i,j)*Eps(i,j)/K(i,j) .
```

This term is used along with the other ϵ terms to calculate the array of turbulence dissipation rates.

The turbulence transport routine is made up of three major loops: the first to calculate donor-cell arrays; the second to calculate Kt1, Kt2, Kt3, Epst1, Epst2, Epst3, and Epst4; and a third to combine these terms using Eqs. (IX-31) and (IX-32).

This routine, along with the turbulent boundary condition routine and the routine to calculate total viscosities, makes up the $K - \epsilon$ turbulence model. The implementation of these three procedures, along with the use of total viscosity wherever molecular viscosity appears in the momentum and turbulence transport equations, is all that is necessary to create a code that calculates turbulence transport.

D. Turbulence Transport and the Karman Vortex Street

Our incompressible code with equations of turbulence transport can be applied to the Karman vortex street problem. In order to compare our results with those obtained in Chapter VII, we can use the same set of parameters, namely: a flow passage 50 (cm) long and 15 (cm) wide, flow around the obstacle at a rate of 1.5 (cm/s), an initial flow rate at the right of 1 (cm/s), and a variable fluid viscosity.

The graphs presented in this section use these parameters on a grid of dimensions ibar = 25 and jbar = 15. Initial K values and input K values are set to 0.01 (cm^2/s^2), whereas initial and input ϵ values are set such that the TURBULENCE SCALE is equal to 2 (cm). The turbulence scale is a measure of the size of the turbulent fluctuations; it is represented by an s and is calculated as

$$s = \frac{K^{3/2}}{\epsilon} . \tag{IX-42}$$

A typical turbulence scale at low Reynolds numbers is about one to one half the size of a major feature, such as the obstacle. At high Reynolds numbers the turbulence scale is more on the order of one-fifth to one-tenth of the size of a major feature.

In this first set of graphs, anu is set to 1; resulting in a system with a Reynolds number of 5. Graphs appear at a time of 100 seconds.

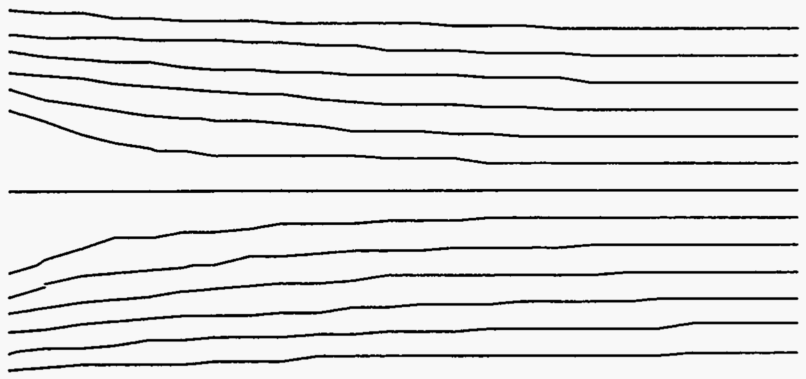


Figure IX-4

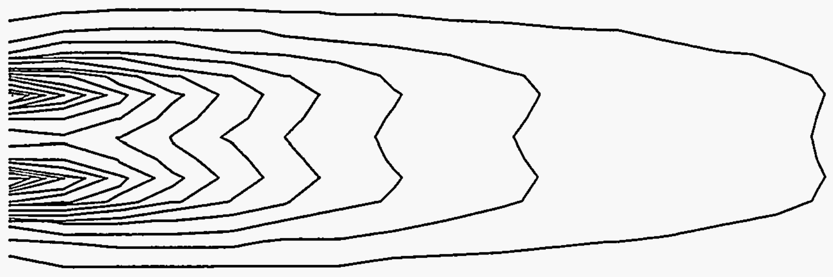


Figure IX-5

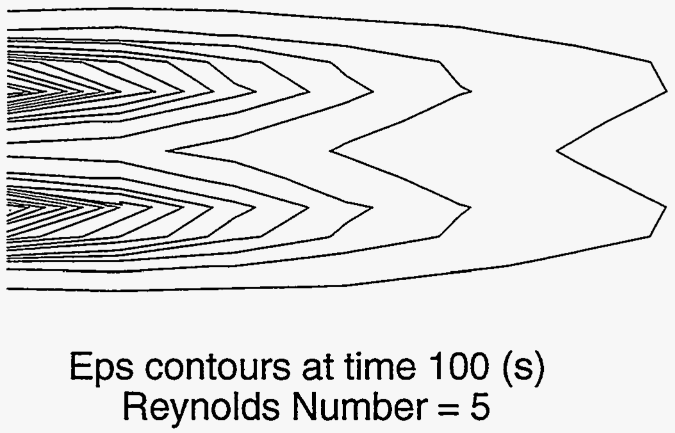


Figure IX-6

Notice that in these graphs, there is no flow separation, and turbulent fluctuations are confined to the region where turbulence is being directly pumped into the system. Turbulent kinetic energy cannot be sustained in this nonfluctuating system and is therefore dissipated.

At a viscosity of 0.2, the system begins to develop stationary vortices as can be seen in Fig. 7.

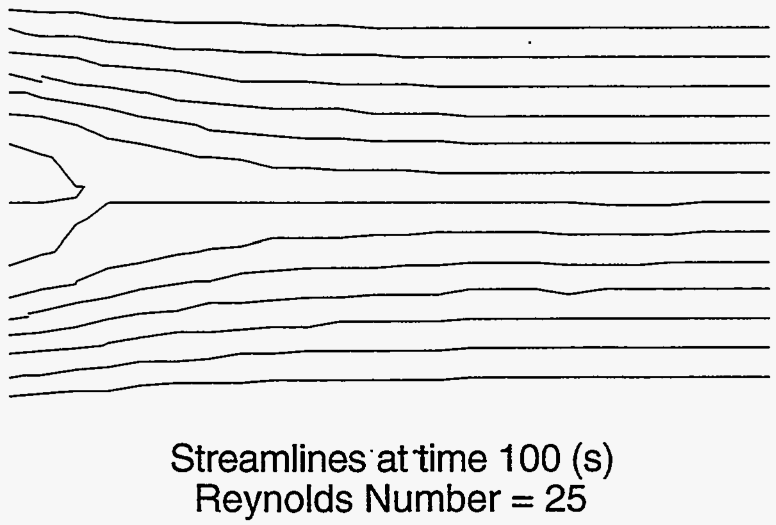
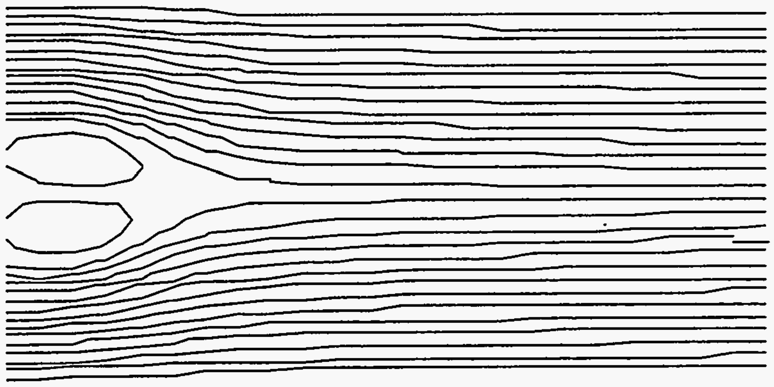


Figure IX-7

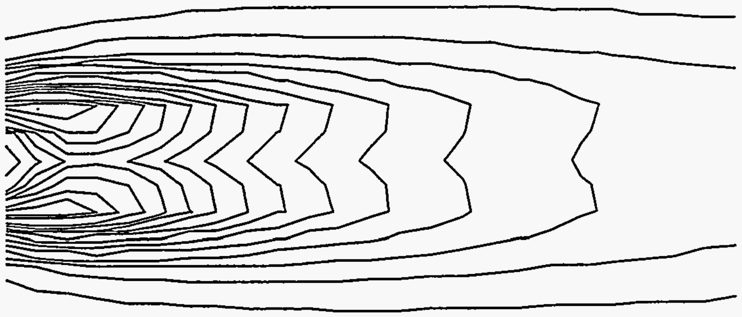
These vortices are better resolved in a run with a jbar of 30.



Streamlines at time 100 (s)
Reynolds Number = 25

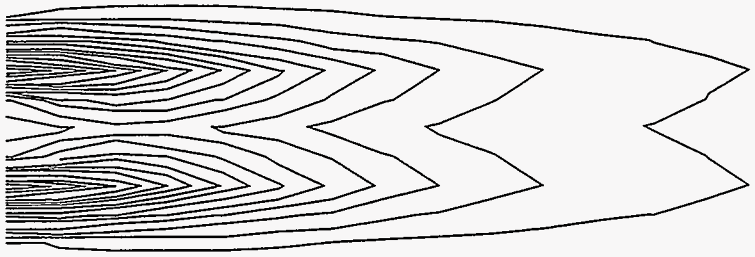
Figure IX-8

K and ϵ in a run using a jbar of 15 appear as follows.



K contours at time 100 (s)
Reynolds Number = 25

Figure IX-9

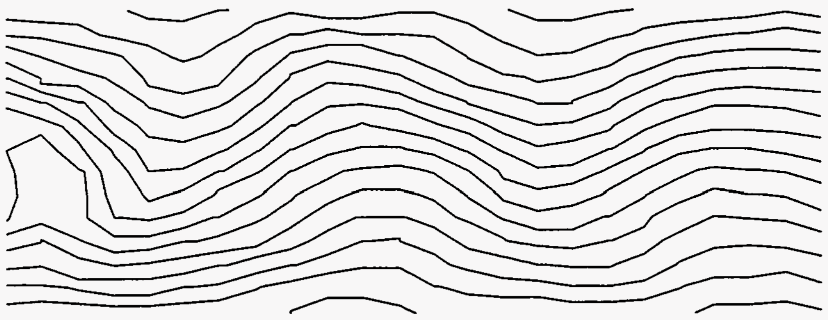


Eps contours at time 100 (s)
Reynolds Number = 25

Figure IX-10

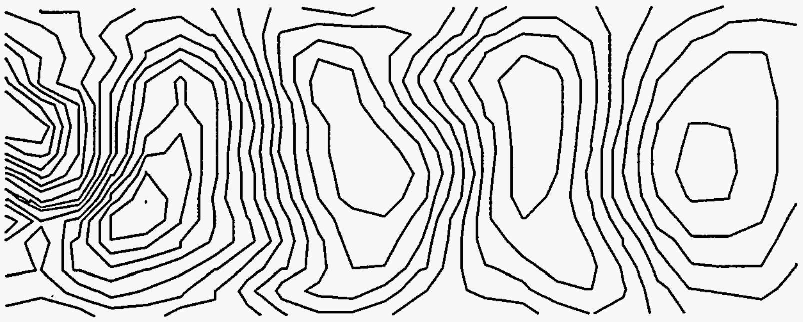
Again we see turbulence energy being dissipated by the system. Turbulence energy is much stronger where it is being pumped into the system than it is anywhere else. At this Reynolds number, however, turbulence energy reaches an area farther downstream than it did in the first example. The Reynolds number has not yet been increased to the level that turbulence is being generated by the system, but it is now sufficiently high for the input turbulence to persist for an extended period of time.

At a viscosity of 0.02 and a Reynolds number of 250, the system develops a Karman vortex street. This can be seen in the following two graphs taken at a time of 100 seconds.



Streamlines at time 100 (s)
Reynolds Number = 250

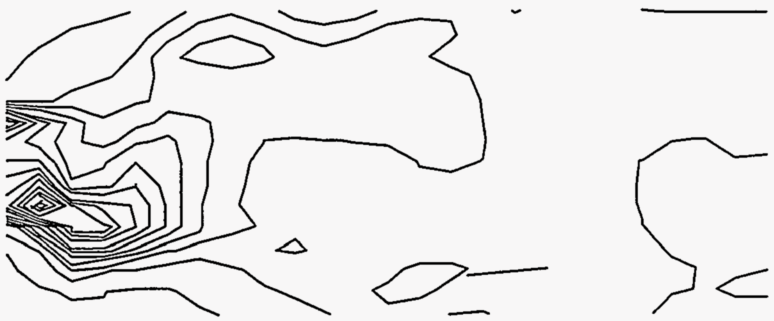
Figure IX-11



Streamlines at time 100 (s)
Reynolds Number = 250
Fluid Reference Frame

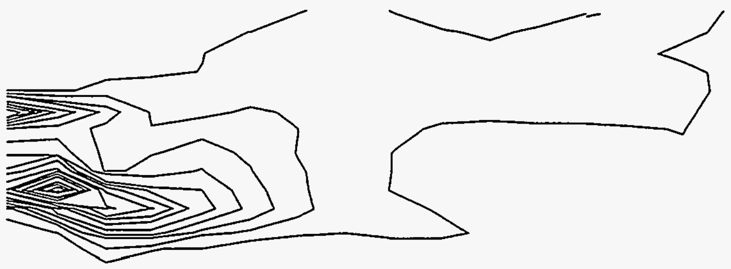
Figure IX-12

K and ϵ at the same time appear as the following



K contours at time 100 (s)
Reynolds Number = 250

Figure IX-13



Eps contours at time 100 (s)
Reynolds Number = 250

Figure IX-14

In these graphs, turbulence is once again strongest where it is being pumped in, but shearing forces farther down the street have created other regions of turbulence. The Reynolds number has been increased to a point where turbulence not only persists but is generated by the system.

The Karman vortex street can be simulated in another manner, by setting the initial K's and the K at the left to a higher value and allowing for the turbulence equations to represent not only the small fluctuations within the stream but the large changes in velocity of the stream itself The system then evolves into a situation where the turbulence kinetic energy is high, thereby resulting in a high-turbulence viscosity. This viscosity lowers the EFFECTIVE REYNOLDS NUMBER of the system, the Reynolds number as calculated using the sum of the molecular and turbulence viscosities. The effective reynolds number is in contrast to the MOLECULAR REYNOLDS NUMBER, which is calculated using only the molecular viscosity. At a low effective Reynolds number, the resolved system does not contained the Karman vortex street itself, but the kinetic energy contained in this stream is visible in the turbulence kinetic energies. Such a system is shown in the following set of plots, which are taken at a Reynolds number of 250 and an initial K and input K of 0.225 (cm^2/s^2).

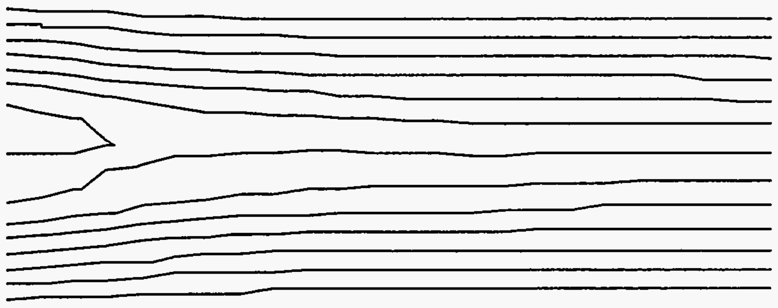

Streamlines at time 100 (s)
Molecular Reynolds Number = 250

Figure IX-15

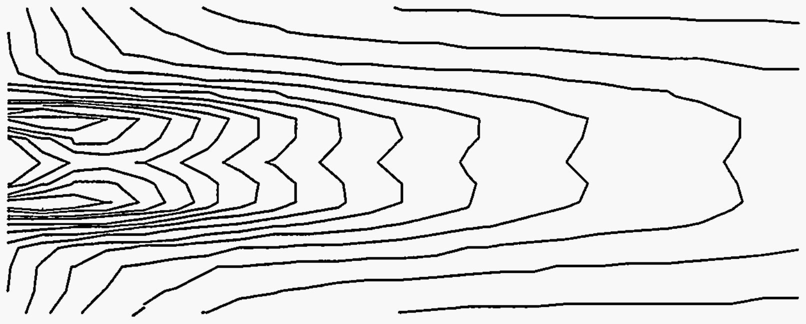

K contours at time 100 (s)
Molecular Reynolds Number = 250

Figure IX-16

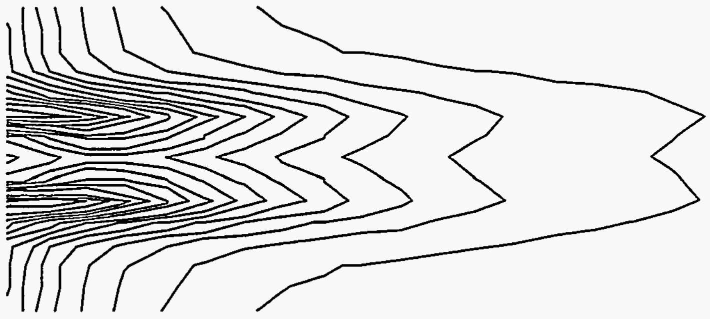
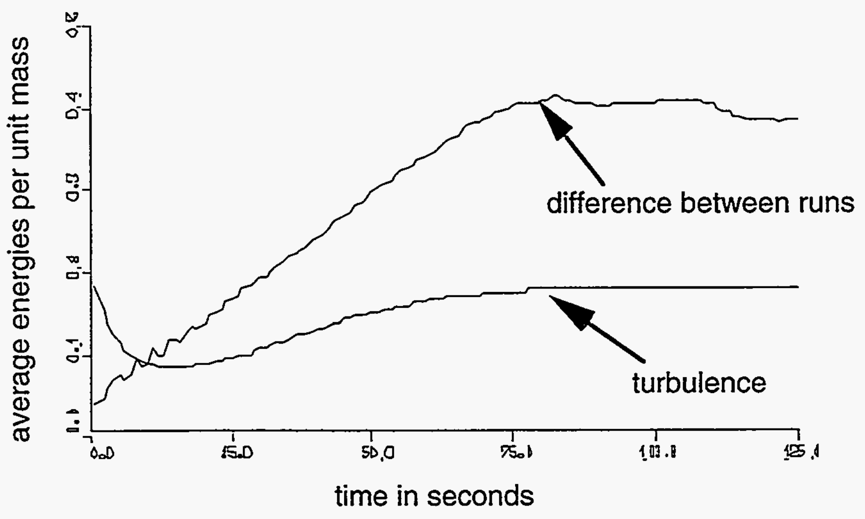

Eps contours at time 100 (s)
Molecular Reynolds Number = 250

Figure IX-17

In these plots, K values are about 0.17 (cm^2/s^2) whereas ϵ values are about 0.016 (cm^2/s^3). These values result in a turbulence viscosity of 0.16 (cm^2/s), corresponding to an effective Reynolds number of 30, which is consistent with the type of resolved flow that can be seen in Fig. IX-15.

It is hoped that the difference between the resolved kinetic energies in this and in another calculation that has no equations of turbulence transport will be comparable to the turbulent kinetic energy. This hypothesis can be tested by recording the resolved kinetic energy and turbulent energy per unit mass of a turbulence transport calculation and the resolved kinetic energy per unit mass of the second calculation at equal time intervals. The difference between the two resolved kinetic energies can then be compared with the turbulence kinetic energy.

Figure 18 shows a comparison of these two values for the parameters used in Figs. IX-15–17.



Comparisons of Fluctuating Energies

Figure IX-18

In this figure, we can see that the turbulent kinetic energy is in fact much less than the difference between the two resolved kinetic energies. This difference is caused by the fact that the turbulence equations are only able to model the fluctuations in the shear layer that occurs behind the object. They are able to simulate the vortex street but underestimate its effect in the top and bottom portions of the flow passage. In the resolved vortex street, fluctuations in the center of the graph have large and immediate effects on the areas in the top and bottom of the graph; these areas can be said to be CORRELATED. This correlation, which can be clearly seen in Fig. IX-12, fails to be accurately represented by the $K-\epsilon$ turbulence model.

The problem is that the $K-\epsilon$ model is a SINGLE POINT TURBULENCE MODEL, meaning that it relies on the values of quantities directly surrounding a single point to generate the turbulence values at that point. This type of model is in contrast with the SPECTRAL TURBULENCE MODELS that are presently being developed. Such models establish correlations between different regions in a fluid and attempt to simulate turbulence in a much more detailed and accurate manner. The development of spectral turbulence models is one of the many areas in which research is being done in the field of computational fluid dynamics.

In this book we have examined the basics of finite-difference methods for numerical fluid dynamics. The equations that we have studied, the terminology we have used, and the techniques we have examined have been used for many years; yet the field is one in which many developments are still being made. Finite-difference codes have given us the ability to mathematically represent and study the behavior of physical systems with increasing accuracy and complexity. Their research and development remains an active and exciting field for those interested in the application of mathematics and computing to the study of the world around us.

Glossary

Adiabatic system (VII-C). A system that contains no processes that either absorb or generate heat. In an adiabatic system, both pressure and internal energy are functions of density.

Advective flux (V-A). Flux that occurs as a result of the motion of fluid from one region to another. An example of this type of flux is heat convection.

Artificial viscosity (VI-E). An additional diffusion term that is added to the finite-difference momentum equation in order to counteract the negative diffusion that is intrinsic to this approximation of a partial-differential equation.

Benard problem (VIII-C). A problem involving a long, thin flow passage that is heated at the bottom, cooled at the top, and insulated along the sides. The Nusselt number can be calculated in this system in relation to the Rayleigh number.

Boundary conditions (II-B). Equations that represent the external conditions that act on a system.

Boussinesq approximation for heat flow (VIII-B). The assumption in an incompressible fluid code that all terms of the momentum equation can be modeled at constant density except the buoyancy term. The Boussinesq approximation for heat flow is not to be confused with the Boussinesq approximation for the Reynolds stress tensor. These are completely independent concepts.

Boussinesq approximation for the Reynolds stress tensor (IX-B). The approximation of the Reynolds stress tensor as

$$-\nu_t\left(\frac{\partial \bar{u}_i}{\partial x_j}+\frac{\partial \bar{u}_j}{\partial x_i}\right)+\frac{2}{3}K\delta_{i,j}\ ,$$

where K is the turbulent kinetic energy and ν_t is the turbulent viscosity. The Boussinesq approximation for the Reynolds stress tensor is not to be confused with the Boussinesq approximation for heat flow.

Cartesian tensor notation (IX-A). A notation that makes use of subscripts to express the general directionality of a quantity without explicitly stating that the quantity is in a particular x-, y-, or z-direction. This notation is helpful in simplifying the equations of fluid motion and aids in the complex derivations that are used to generate turbulence transport models, as well as many other models in physics.

Cell (II-B). An element of finite size that is used to represent the conditions at an arbitrary position in a system. A cell is also called a zone.

Centered flux (V-B). An advective expression that uses the average of the values of the quantities on both sides of the advective surface as the value of the advected quantity.

Coefficient of heat conductivity (II-B). A quantity that is proportional to the rate at which a given material conducts heat across a temperature gradient. Its units are $\frac{\text{(energy)}}{\text{(length)(time)(temperature)}}$.

Compressible fluid (IV-A). A fluid that is moved at speeds comparable to its sound speed, causing it to change its density.

Conservation (II-A). The concept that mass, momentum, and energy are never destroyed, only change form or move from one region to another.

Contact Discontinuity (IV-E). Any fluid discontinuity that moves with its fluid elements, such as the fluid interface in a shock tube.

Correlation (IX-D). An interdependence between quantities not necessarily located adjacent to each other.

Courant condition (IV-C). A numerical stability condition that occurs as a result of the finite-difference approximation of the momentum equation. The Courant condition is

$$\frac{|v|dt}{dx} < 1, \qquad \text{(IV-36)}$$

where v is $(|u| + c_{\text{sound}})$.

Donor-cell flux (V-B). An advective expression that uses the upstream value of the advected quantity.

Effective Reynolds number (IX-D). The Reynolds number as calculated using the sum of the molecular and turbulence viscosities. See Reynolds number.

Error Function. A function that often results as a solution to a partial-differential equation, the error function is defined as

$$erf(x) \equiv \frac{2}{\sqrt{\pi}} \int_0^x e^{-x^2} dx \ .$$

The error function can be calculated using the table at the end of Section III-E. It is also called the probability integral.

Eulerian fluid-mechanics code (IV-A). A code in which zones remain fixed in space. In this type of code, fluids move in and out of zones at various rates, causing the mass contained in a particular zone to change as the simulation progresses. All physical quantities are fluxed between cells, but the positions of the cells remain the same.

Explicit solving method (III-C). A solving method in which values at each new time cycle are calculated directly from the values at the previous time cycle. This is in contrast to the implicit solving method.

Fictitious zones (II-D). Finite-difference zones that exist beyond the normal boundaries of a system and are used in representing boundary conditions. Fictitious zones are also called ghost zones.

Fluid (IV-A). A material that is infinitely deformable or malleable. A fluid may resist moving from one shape to another but resists the same amount in all directions and in all shapes.

Flux (II-A). The amount of a quantity passing through a unit area in a unit time.

Ghost zones (II-D). Zones that exist beyond the normal boundaries of a system and are used in representing boundary conditions. They are also called fictitious zones.

Hot spot (VIII-B). A section of wall that contains a prescribed temperature boundary condition in an otherwise insulated system.

Implicit solving method (III-C). A solving method in which values at a new time cycle are calculated based on the rate of change of values at this new time step. Values at the old time step are used only indirectly. This is in contrast to the explicit method.

Incompressible fluid (IV-A). A fluid that moves at far subsonic speeds and does not change its density.

Infinite-strength shock (IV-E). A shock that moves at a speed that is large compared to the sound speed of the fluid ahead of the shock.

Insulated boundary condition (VIII-B). A boundary condition in which there is no heat fluxed across the wall. It is achieved by specifying a zero temperature gradient across the wall.

Isotropic (VII-B). The quality of not varying as a function of direction.

$K - \epsilon$ turbulence model (IX-B). A turbulence representation that contains transport equations for the turbulent kinetic energy per unit mass (K) and the dissipation rate of that turbulence (ϵ).

Karman vortex street (VII-C). A type of turbulent fluid flow that occurs in systems in which a fluid within an appropriate range of velocities and viscosities flows around an object. The Karman vortex street is a fluctuating stream with alternating eddies that is caused by the shedding of vortices. It is also sometimes called the Von Karman vortex street.

Kinematic viscosity (VII-B). The normal molecular viscosity of a fluid; the kinematic viscosity is produced in gases by the fluctuating departures of the velocities of the molecules from some mean value. In liquids it is caused primarily by the intermolecular forces.

Kronecker symbol (IX-B). A second order tensor that is designated as $\delta_{i,j}$. The Kronecker symbol is one if i equals j and zero otherwise.

Lagrangian derivative (V-C). An expression for the rate of change of a quantity along the motion of a fluid. It is equal to $\frac{\partial q}{\partial t} + \frac{\partial q}{\partial x}\frac{\partial x}{\partial t}$ where q is the quantity that is changing. The Lagrangian derivative is denoted as $\frac{dq}{dt}$ as opposed to $\frac{\partial q}{\partial t}$.

Lagrangian fluid-mechanics code (IV-A). A fluid code in which the positions of zones vary between time steps. As fluids are compressed and decompressed, the zones move accordingly, maintaining an equal mass throughout the simulation. In a Lagrangian calculation, the energies, momenta, and positions of the zones change from time step to time step; only the mass contained by each zone is held fixed.

Mach number (VIII-E). The ratio of the velocity of a shock to the sound speed ahead of that shock. The Mach number is defined as

$$M \equiv \frac{v}{c_{\text{sound}}} \, .$$

Mach stem (VIII-E). A shock that is formed between a shock that hits an obstacle and the resulting reflected shock. A Mach stem is always perpendicular to the obstacle.

Mass Matching (IV-E). In a Lagrangian calculation, the decreasing of the initial volumes of the denser zones and the increasing of the initial volumes of the less dense zones in a manner such that the masses of all zones are equal.

Mean flow (IX-B). The steady part of a fluid flow; the part of a fluid flow that is not considered turbulence.

Modeling (IX-B). The approximation of a true transport equation with a more simple equation that retains the properties of the original equation but is not algebraically equivalent.

Molecular Reynolds number (IX-D). The Reynolds number as calculated using only the molecular (kinematic) viscosity. See Reynolds number.

Natural Convection (VIII-C). The circulating motion of fluid between regions of different temperatures due to the difference in the fluid density at each of these temperatures.

Navier-Stokes Equations (IV-B). A general term for the equations that describe the motion of fluids.

Nonadvective flux (V-A). Flux in addition to the advective flux that occurs when quantities diffuse from one area to another. Examples of this sort of flux are pressure flux in the momentum equation and work flux in the energy equation.

Nusselt Number (VIII-C). The ratio between the total heat flux in a system and the heat flux due only to conduction:

$$Nu = \frac{\text{Total Flux}}{\text{Conductive Flux}} \,. \tag{VIII-26}$$

Obstacle (VIII-A). An object that prevents fluid from flowing through a specified subregion.

Prescribed-temperature boundary condition (VIII-B). A boundary condition in which the wall exists at a prescribed temperature. For this condition the temperature gradient across the wall is chosen such that the temperature at the wall remains at a specified value.

Polytropic equation of state (IV-D). An equation that relates pressure to density and internal energy in an ideal gas. The polytropic equation of state is

$$p = (\gamma - 1)\rho I \,. \tag{IV-23}$$

Polytropic gas constant. A variable that represents the ratio of specific heats in an ideal gas. The Polytropic gas constant is designated by a γ.

Probability Integral. See error function.

Rarefaction Wave (IV-E). A wave that occurs in a region of high density when a barrier is removed between that region and a region of lower density.

Rayleigh number (VIII-C). A dimensionless number that relates the magnitudes of the buoyancy and viscous forces in a system. In the Benard problem, the Rayleigh number is calculated as

$$Ra = -\frac{gh^3\beta\Delta T}{\nu\sigma} \,, \tag{VIII-27}$$

where g is the acceleration of gravity (defined as negative if downward), h is the height of the passage, ΔT is the difference in temperatures between the top and the bottom of the passage, ν is the viscosity of the fluid, σ is the thermometric conductivity of the fluid, and β is $\frac{1}{T_0}$ the inverse of the reference temperature.

Reynolds number (VII-E). The Reynolds number is a dimensionless quantity that compares the advective versus the diffusive properties of a system. It can be used to predict the tendency of a system towards turbulence. For the Karman vortex street problem, the Reynolds number is calculated as

$$Re = \frac{h_{\mathrm{obs}} u_\infty}{\nu} , \tag{VII-94}$$

where h_{obs} is the height of the obstacle, u_∞ is the velocity of the fluid far away from the obstacle, and ν is the viscosity of the fluid. As the Reynolds number increases, the system is likely to become more turbulent.

Reynolds stress tensor (IX-B). A second order tensor that serves as a measure of the turbulence of a system. The Reynolds stress tensor is equal to the ensemble average of the product of the fluctuations in fluid velocities in two directions:

$$R_{i,j} \equiv \overline{u_i' u_j'} ,$$

where u_i' and u_j' are first order tensors representing the fluctuations in velocities in the i- and j-directions.

Shear force (IX-B). A force similar to friction that is caused by flows at different velocities rubbing against each other.

Shock (IV-E). A rapid transition between two states that moves relative to the fluid. It is also called a shock front.

Shock Front (IV-E). Same as a shock.

Shock Tube (IV-E). A tube containing two fluids, usually gasses, of different densities that are used to study the properties of shocks and rarefactions

Single-Point Turbulence Model (IX-D). A turbulence model that relies on the values of quantities directly surrounding a single point to generate the turbulence values at that point. Such a model contains no correlations.

Spectral Turbulence Model (IX-D). A turbulence model that establishes correlations between different regions in a fluid. Turbulence values at any given point are calculated in conjunction with these correlations rather than using only the values adjacent to that point.

Staggered mesh (VII-B). A fluid dynamics computational mesh in which some variables exist at cell walls and others exist at cell centers.

Streamlines (VII-D). Lines that indicate the path along which the fluid is flowing.

Strouhal Number (VII-E). A dimensionless number that relates the period of the stream to the size of the object and the rate of the flow. The Strouhal number is a dimensionless quantity that is calculated as

$$St = \frac{h_{obs}}{u_\infty \tau_{\text{street}}} , \tag{VII-96}$$

where h_{obs} is the height of the obstacle, u_∞ is the velocity of the fluid far away from the obstacle, and τ is the period of the street. In a Karman vortex street, the Strouhal number has been experimentally observed to be approximately 0.2.

Taylor-series expansion (VI-C). An expansion that uses Taylor's theorem. In a Taylor-series expansion, a function $f(x + dx)$ becomes

$$f(x) + \frac{dx}{1} f'(x) + \frac{dx^2}{2!} f''(x) + \frac{dx^3}{3!} f'''(x) + \dots ,$$

where f', f'', f''', etc., are the first, second, third, etc., derivatives of the function f.

Tensor order (IX-A). A measure of the number of directional dimensions associated with a quantity. A scalar, for example, has a tensor order of zero, indicating that it has no directionality associated with it. A vector, having a single direction, is a quantity with a tensor order of one. The Reynolds stress tensor, the product of two vectors, has a tensor order of two. Higher order tensors exist with a number of directions equal to their tensor order.

Thermometric conductivity (II-B). Notated by σ, this quantity is equal to the coefficient of heat conductivity of a material divided by its density and specific heat $\left(\frac{k}{\rho b}\right)$. Its units are those of an area per unit time.

Time cycle counter (II-B). An integer that represents the number of time cycles that have been calculated in a simulation.

Truncation error analysis (VI-A). A method that can be applied to determine the error of finite-difference approximations. Truncation error analysis involves using a Taylor series expansion on a finite-difference approximation and comparing the resulting equation with the original partial-differential equation.

Turbulence (IX-B). The fluctuating portion of a fluid flow. The part of a fluid flow that is not considered mean flow.

Turbulence scale (IX-D). A measure of the size of turbulent fluctuations. The turbulence scale is denoted by an s and is calculated as

$$s = \frac{K^{3/2}}{\epsilon} \,. \tag{IX-42}$$

Turbulent kinetic energy (IX-B). The kinetic energy that is present in turbulent fluctuations. Turbulent kinetic energy is often measured as turbulent kinetic energy per unit mass which is denoted by K.

Turbulent viscosity (IX-B). Viscosity that results from turbulent fluctuations in a fluid. It is denoted as ν_t.

Unconditionally unstable (V-B). Unstable regardless of the parameters that are chosen.

Vortices (VII-E). Areas in a fluid flow where fluid is not moving along with the main flow but rather circling in an eddy.

Wall function (IX-C). A function that is used to calculate K and ϵ values at the ghost zones in problems where the turbulent conditions at the boundaries are important.

Zone (II-A). An element of finite size that is used to represent the conditions at an arbitrary position in a system. A zone is also called a cell.

Acknowledgments

We would like to thank Margaret Findley for typsetting this work and creating the figures, as well as Eric Harstad, Denise Hunter, and Patricia Mendius for help in editing, and Thomas Adams and XHM for making the publication of this work possible. We would also like to thank thank T-3, XHM, and Los Alamos National Laboratory for continuing to support research programs for students in the sciences.