UNIVERSITY OF MINNESOTA

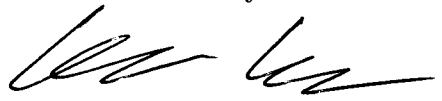
This is to certify that I have examined this copy of a doctoral thesis by

Michael J. Wright

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Graham V. Candler

Name of Faculty Advisor



Signature of Faculty Advisor

6/11/97

Date

GRADUATE SCHOOL

A Family of Data-Parallel Relaxation Methods for the Navier-Stokes Equations

A THESIS
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Michael James Wright

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

June 1997

Acknowledgments

I am grateful to the many friends and colleagues who have made this dissertation possible. First and foremost my adviser, Professor Graham Candler, for his constant support, encouragement, and enthusiasm during the evolution of this work. Jim Weilmuenster and Chris Riley coordinated the efforts at NASA Langley to test and benchmark the methods. Tom Magruder and Dr. Alexander Smits at Princeton University provided experimental results for the double-cone shock interactions. I would also like to thank my fellow graduate students and friends at the University of Minnesota, especially Dr. Deepak Bose and Dr. Joe Olejniczak, who proved through their support and advice that two (or even six!) heads definitely are better than one. Finally, I would like to thank my parents for their encouragement and faith.

This work was supported by the NASA Langley Research Center Contract NAG-1-1498. This work was also sponsored in part by the Army High Performance Computing Research Center under the auspices of the Department of the Army, Army Research Laboratory cooperative agreement number DAAH04-95-2-0003 / contract number DAAH04-95-C-0008, the content of which does not necessarily reflect the position or the policy of the government, and no official endorsement should be inferred. Computer time on the Cray T3E was provided by the Minnesota Supercomputer Institute.

Abstract

The use of parallel supercomputers for the solution of the Navier-Stokes equations is attractive, since such machines typically have more memory and computational power than conventional supercomputers. However, most current computational fluid dynamics (CFD) methods are not amenable to implementation on parallel machines, due to inherent data-dependencies. In this dissertation a new set of parallel implicit CFD algorithms are designed for the solution of the compressible Navier-Stokes equations on structured meshes. These new methods are inherently efficient on both data-parallel and message-passing computers. In order to accomplish this, the derivation of several commonly used serial and vector algorithms is examined, and the limitations of these methods are discussed. Then, modifications are proposed which allow the methods to be parallelized effectively. The result is a family of data-parallel relaxation methods, consisting of three new numerical algorithms. The performance of each new method is examined on a set of test problems, and their relative merits are discussed. Each new method exhibits a high parallel efficiency on a variety of platforms, and all show promise for the large-scale simulation of compressible flows.

Shock interaction flows provide a rigorous test for these new methods, because they can be extremely complicated, and require a large number of grid points to simulate. Therefore, the second part of this dissertation focuses on the solution of a series of shock interactions. First, inviscid interactions on double-wedge geometries are simulated using very fine grids, so that the underlying compressible gasdynamics can be better understood. Five different shock interactions are identified, including one which has previously not been classified. At high Mach number many of these interactions exhibit an underexpanded jet which is trapped against the body surface, creating large amplitude steady-state pressure variations. Tran-

sition criteria between the various types of interaction are discussed. Then, a series of computations are made to simulate a series of experiments on shock interactions over double-cones are performed at Princeton University. Schlieren images taken during the experiments compare very well with the laminar CFD calculations.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivations

In recent years there has been an increasing demand for the solution of larger and more complex computational fluid dynamics (CFD) problems. As aerospace budgets have dwindled, there has been considerable interest in using full three-dimensional vehicle simulations to replace wind tunnel or flight tests in the preliminary design phase. This would potentially allow a much faster turn around time at a much lower cost. However, the number of grid points that are required to adequately simulate such 3-D flows results in an extremely large-scale problem that is beyond the capabilities of most conventional serial and vector supercomputers.

The numerical simulation of nonequilibrium reacting flow problems is an even more computationally and memory intensive task, due to the large number of equations which must be solved at each grid point. For example, even for the relatively simple case of reacting non-ionizing air over a hypersonic vehicle there are at least ten conservation equations which must be solved at each grid point (five species mass equations, three momentum equations, and two energy equations). More complex flowfields, such as combustion simulations, can require the solution of a much larger equation set. In addition, the calculation of the source terms which govern the thermo-chemical state of the gas can be very intensive. The solution of this extended equation set quickly becomes very expensive on conventional supercomputers, which effectively limits the size and complexity of problems which have been attempted.

In addition, the large disparity between the different length scales in high Reynolds number flows and the time scales in chemically reacting flows can result in a very stiff equation set. Therefore, it is usually necessary to use some sort of implicit time advancement method so that steady-state solutions may be obtained in a reasonable time. However, the use of an implicit method can introduce further complexities into the numerical algorithm, due to required matrix inversions and proper handling of implicit boundary conditions.

Parallel supercomputers have long been viewed as a potential solution to the computational bottleneck which has occurred in large-scale CFD. Parallel machines typically boast a very large theoretical peak floating point performance, although inter-processor communication, which is usually much slower than on-processor computation, limits the operational performance for most real problems. Massively-parallel computers have been used with varying degrees of success on a variety of related problems. However, it is very difficult to implement most implicit methods on these machines. An exact implicit method requires the inversion of a large sparse matrix at each time step, which involves a great deal of inter-processor communication, and therefore reduces the operational performance of the machine. In order to better understand the challenge of implementing an implicit algorithm on a parallel machine, we first briefly review the major types of parallel supercomputers.

1.2 Types of Parallel Computers

Parallel computers are typically classified according to the way the memory is accessed by the processors. In a distributed memory machine, each processor has its own local memory, which can be accessed very quickly. However, if some piece of information is required that resides on another processor's memory, it must be retrieved through inter-processor communication. This communication is typically much slower than a local memory access, and thus the situation can arise that a processor remains idle while waiting for a piece of data to be received from another processor. This can form a bottleneck to the overall performance of a parallel

algorithm. The total time spent on inter-processor communication is given by

$$t_c = t_s + \frac{m}{b_m}$$

where t_c is the total time spent on communication, t_s is the latency, or startup time, m is the message size, and b_m is the communication bandwidth. Each separate communication request will incur its own startup time; therefore, for an efficient algorithm it is essential to minimize both the amount of data that is transferred between processors and the total number of communication requests.

In contrast, a true shared memory machine has a single memory location, and all processors can access that memory with equal speed. This would seem at first to be a much better design, since the problem of inter-processor communication is eliminated, and a serial or vector algorithm could be implemented on such a machine with minimal modification. However, in practice there are several problems with true shared memory machines. Most importantly, as the number of processors increases it becomes increasingly difficult to design an efficient inter-connection network (ICN) to connect each processor with all memory locations. As the ICN becomes more complicated there is increased chance of memory conflicts, which can occur when more than one processor attempts to access the same memory address. In addition, contention becomes a problem, which means that memory requests can be slowed as other requests make use of the same part of the ICN. The contention problem is analogous to a traffic jam, which occurs when too many cars attempt to use the same part of the highway system. These issues limit the number of processors that can be efficiently used in a true shared memory machine. This problem can be avoided somewhat by designing the machine such that the memory is physically distributed but logically shared. In this way, each processor has its own local memory, but can also directly access the memory of any other processor without explicit communication. However, with this architecture access times are typically much larger for non-local memory accesses, resulting in the same bottleneck that occurs for distributed memory machines. This work assumes

that the target machine has a distributed memory, but the resulting algorithms are equally effective on logically shared memory machines.

There are two main programming styles on distributed memory parallel machines. In the data-parallel programming style the data is spread evenly across the available processors by the compiler, and each processor executes the same source code in lock-step with all other processors. This means that proper load balancing will be ensured, as long as the the compiler has properly distributed the data. However, during portions of the computation that operate on a subset of the problem, such as the calculation of boundary conditions, all processors that do not contain a part of the subset simply remain idle while the others perform the computation. Thus the calculation of boundary conditions can upset the load balance. This programming style is commonly referred to as Single-Instruction Multiple-Data (SIMD) programming. Since the compiler handles the data distribution process, the interprocessor communication is somewhat hidden. The programmer does not explicitly request that a communication routine take place. Rather, the compiler determines when non-local data is being accessed, and generates its own communication requests. This simplifies the coding process somewhat, at the expense of flexibility. This programming style is very structured in nature, and is thus well suited to the type of structured mesh CFD applications discussed in this research. In addition, algorithms that have been designed to adhere to this rigid data-parallel style are typically easy to port to many parallel platforms, since they exhibit regular and organized communication patterns.

The other distributed memory programming style is called message-passing, and is much less rigid in design. In the message-passing style the user must explicitly break the problem up and distribute it to the various processors, in a process called domain decomposition. This process is illustrated schematically in Fig. (1.1). In this figure, each sub-domain of the problem is assigned to a different processing node, as indicated by the numbers. It is now the user's responsibility to distribute

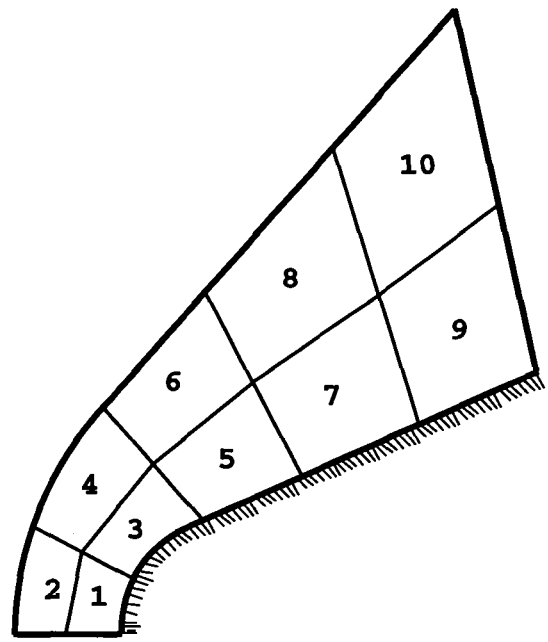


Fig. 1.1 Schematic diagram of a typical explicit domain decomposition of the computational grid around a cylinder-wedge blunt body. Each individual sub-domain is given a number representing the processor on which that portion of the problem will be placed.

the data in a way that load balancing is ensured. In this sense, the decomposition shown in Fig. (1.1) may not be the most efficient one if the wall boundary conditions are complex, since only half of the processors contain a portion of the wall boundary. This process is made even more complex, because now each processor is allowed to work independently of all others, even to the point of executing entirely different source code, if desired. This programming style is commonly referred to as Multiple-Instruction Multiple-Data (MIMD) programming. Inter-processor communication is required only when data must be shared across the boundaries of the sub-domains. Unlike the data-parallel programming model, the programmer must explicitly declare all necessary communication routines, telling the computer what information must be sent, as well as the source and destination processors for each such "message". It is also the programmer's responsibility to ensure that the messages have been properly received before the data are required, and that the execution of the code proceeds at the same rate on each of the nodes. While this

programming style requires more input from the programmer, it is also much more flexible than data-parallel programming, since the programmer now has explicit control over when communication takes place, and how the problem is distributed among the processors.

Finally, parallel computers are classified according to the degree of parallelism. Coarse-grained parallel machines, such as the Cray C90, consist of a small number of powerful processors. Fine-grained machines, such as the MasPar MP-2, can include many thousands of small processing nodes. Most current parallel supercomputers are somewhere in between these extremes. Such medium-grained machines consist of anywhere from tens to hundreds of processors, and are typically physically distributed memory machines that offer support of the message-passing programming style. Support for the data-parallel programming style is becoming less common, and is now usually offered only through third party software, such as High Performance Fortran (HPF).

1.3 Parallel Machines Used in This Work

In this section we introduce the two primary parallel supercomputers used in this work. Some specific information is given about the machine configuration and hardware. Then we briefly discuss implementation issues on each machine, and point out what is required for an algorithm to run effectively in such an environment.

1.3.1 Thinking Machines CM-5

The first machine for which results will be presented is the 896 processor Thinking Machines Connection Machine 5 (CM-5), located at the University of Minnesota Army High Performance Computing Research Center (AHPCRC). This machine is regularly configured into smaller power-of-two sized partitions with between 32 and 512 processors, which makes it readily accessible to a variety of problem sizes. Each of the smaller partitions is time-shared, which means that multiple jobs can be running at any given time, provided they all fit into the available memory. This

means that the elapsed (wall clock) time and the execution time of a code are not necessarily the same. On the other hand, the 512-processor machine runs in dedicated mode, and is intended only for large jobs. Each processor of the CM-5 has 32 Megabytes (MB) of memory and four vector units, each with a peak theoretical performance of 32 Megaflops (MF). A Megaflop is defined as 1 million floating point operations per second. Thus, the 512-processor machine has a total of 16 Gigabytes (GB) of available memory and a peak theoretical performance of 64 Gigaflops (GF). Each CM-5 partition is controlled by a front-end machine, which is essentially a sparc-station. This front-end performs serial operations in the code, handles I/O routines, and acts to synchronize the individual processors. The processors themselves are arranged in a fat-hypertree interconnection network. This network has the same geometry as a basic tree, but with an increasing number of wires closer to the root. The major advantage of this ICN is that it offers a fairly "flat" bandwidth for all inter-processor communication. This means that the physical separation of the processors has little or no effect on the time it takes to send a message between them. Unfortunately, this ICN is also very slow relative to the processor speed, due primarily to the large communication latency and small bandwidth of only 5 MB/sec. Thus, for an efficient parallel algorithm it is essential to minimize both the number of communication routines and the total amount of data that must be sent between processors.

The CM-5 is a Multiple-Instruction Multiple-Data (MIMD) distributed memory computer, and supports both the message-passing and data-parallel programming styles. However, we present results here only for the data-parallel programming style. The CM-5 supports both Fortran and C programming languages, although we use only the Fortran compilers. The "flavor" of Fortran used on the CM-5 is a data-parallel language developed at Thinking Machines, called CM-Fortran (CMF). This language is essentially a superset of the Fortran 90 standard, with additional extensions to ease the writing of data-parallel code. The main difference

between coding in CMF and Fortran 77 (F77) is that in CMF all do-loops are implicit, and thus all temporary variables must be arrays. This is illustrated by the following code segments, which perform identical tasks. The first is written in F77 and the second in CMF:

Fortran 77

```
do i = 1,il
do j = 1,jl
    t = p(i,j)/(gc*rho(i,j))
    c(i,j) = sqrt(gm*gc*t)
enddo
enddo
```

CM-Fortran

```
t(1:il,1:jl) = p(1:il,1:jl)/(gc*rho(1:il,1:jl))
c(1:il,1:jl) = sqrt(gm*gc*t(1:il,1:jl))
```

In both examples, `gc` and `gm` are scalars, while `p`, `rho`, and `c` are stored as arrays. The variable `t` is a temporary variable used in the computation of `c`, but not stored. In F77, this variable is merely a scalar which is overwritten each time through the loop. In CMF, on the other hand, the do-loops are implicit, and `t` is promoted to an array of the same size as `c`. This is important because in the data-parallel programming style the arrays are spread across all of the available processors, and all processors execute each statement sequentially in lock-step by operating on that portion of the array which is local. Therefore, on a data-parallel machine the computer will calculate all elements of `t` *first*, and then compute all elements of `c`. CMF is thus ideally suited to this data-parallel programming style. The major drawback of CMF is that by promoting all loop temporaries to arrays the memory storage requirements of the code can increase considerably. Note that if the operations in CMF occur over the entire array, the index notation can be replaced by a simple place-holder colon (:), or even eliminated entirely from the command. The resulting code fragment shown above then becomes

```
t = p/(gc*rho)
```

```
c = sqrt(gm*gc*t)
```

where `t,p,rho,` and `c` are all arrays of dimension (il, jl). Although in general the user does not know how the data have been distributed across the processors, there are a few things that are known. First, all conformable arrays will by default be distributed in the same way. This means that, if arrays `a` and `b` are conformable, the (i, j) element of each will always lie on the same processor. Therefore, a code segment such as that shown above will not generate any communication, as long as the arrays `t,p,rho,` and `c` are all conformable. In addition, the layout of arrays on the processors can be controlled somewhat by the use of compiler directives.

Inter-processor communication occurs when the program accesses non-local data. For example, the following code fragment would result in inter-processor communication on the CM-5:

```
tb(i,j) = 0.5*(t(i,j) + t(i+1,j))
```

since the i and $i+1$ elements of an array are in general not located on the same processor. The equivalent statement in CMF is generated using an F90 intrinsic function, the circular shift (CSHIFT). The CSHIFT function shifts the data elements in an array along a specified dimension by a specified number of indices. The code segment is then

```
tb = 0.5*(t + cshift(t,dim=1,shift=-1))
```

Note that in the data-parallel programming style, we are not declaring an explicit communication routine. We are merely requesting data that the machine determines is non-local. The compiler then constructs the communication request, in a process that is transparent to the user. Since the CSHIFT command generates a very structured nearest-neighbor communication pattern, it can be highly optimized. This is done on the CM-5, where communication generated by the CSHIFT command is much faster than a general router communication, such as that generated by the first code fragment above. Therefore, for good performance on the

CM-5, it is important that the communication generated by the algorithm be of this relatively fast nearest-neighbor style whenever possible. The CM-5 is thus ideally suited to the types of structured mesh CFD applications developed in this work, which require contributions at point (i, j) from only the four nearest-neighbors (six in three-dimensions).

1.3.2 Cray T3E

The second machine for which results will be presented is the Cray T3E, located at the Minnesota Supercomputer Center. This machine is currently configured with 68 processors, of which 64 are available for user applications. Eight of these 64 processors are reserved for short interactive jobs, while the other 56 accept batch jobs only. Unlike the CM-5, this machine is not time-shared, so the wall clock time and the execution time should be similar. Users can request any number of nodes between 1 and 56, however single node jobs are run on one of two command processors, which are time-shared and also provide the user interface to the machine. The processors themselves are Digital Equipment Corporation Alpha EV5 chips, which are capable of 600 MF peak performance. This is a cache-based RISC microprocessor with pipelined functional units. Each chip has 128 MB of local memory. The processors are connected in a 3-D torus by a high speed network capable of a payload bandwidth of 480 MB/sec. This is a much higher bandwidth than the CM-5, which implies that inter-processor communication will not be as much of a performance issue on this machine. In fact, the most important performance issue on the T3E is the single processor performance of the algorithm.

The T3E is also a MIMD supercomputer. The memory of this machine is logically shared but physically distributed, as discussed in Section 1.2. However, due to the relatively long access times for non-local data retrieves, the machine is usually programmed using message-passing as if the memory were logically distributed. Cray does provide a shared memory programming language (SHMEM), although this was not used in this research. There is currently no support for the data-parallel

programming style on the T3E. The machine supports both F90 and C programming languages. For a message-passing code we have found that F77 can be more highly optimized than F90; therefore we use F77 for most of the code, and include F90 constructs only when they offer a performance improvement. This is not difficult, since the F90 standard is a superset of the F77 programming language.

Since we are now programming in message-passing mode, the programmer must explicitly break the problem into smaller domains, and assign them to the processors in a way that ensures load balancing. For a medium-grained parallel machine such as the T3E, it is to our advantage to break the problem across only one grid dimension, if possible. This will increase the efficiency by reducing the amount of inter-processor communication required. Once the problem has been broken into sub-domains, computation in each domain can occur in parallel. However, whenever non-local data is required the programmer must include explicit communication requests in the program. In this research, these requests are written using the Message-Passing Interface (MPI) standard. One of the advantages of MPI is that it allows the programmer to hide communication latency with the use of non-blocking data sends and receives. The following code segment demonstrates schematically how this process works:

```
send boundary data
perform computations that do not require boundary data
        ⋮
wait for receipt of boundary data
perform remaining computations
```

The programmer first "posts" a send request that tells the computer what data should be sent, as well as the source and destination processors of each message. This command is only a send *request*, and executes immediately. Special communications hardware on the processor then begins to process the message and collect the necessary data. While this is happening, a series of computations that do not re-

quire the boundary data are performed. Then, a receive is posted, and the machine is instructed to wait until the message has been received. Finally, computations that require the boundary data can be performed. In this way, if enough computation can be performed between the send and receive, the time spent waiting for message receipt can be reduced to near zero. However, to take advantage of these non-blocking sends and receives it is necessary that the communication occur in a very structured and well defined manner, and that data-dependencies are minimized so that most of the problem can be computed while the messages are being sent. Therefore, an algorithm that has been optimized for a data-parallel programming style, which requires structured communication and few data-dependencies, will be very efficient in a message-passing environment.

1.4 Scope of the Current Research

The traditional solution to the problem of implementing an implicit method on a parallel computer has been to choose an effective serial algorithm, and implement it in parallel using some sort of explicit domain decomposition. This approach has been used with some success [Simon (1992)], but many of the most effective serial algorithms contain inherent data-dependencies, and cannot be implemented effectively in parallel without modification.

Another approach would be to design a new implicit algorithm that could take advantage of the nearest-neighbor communication pattern of a structured mesh approach. Such an algorithm would be inherently well suited to a data-parallel environment without explicit domain decomposition. The algorithm would then in principle be efficient and easy to implement in either data-parallel or message-passing mode, as discussed above, and would thus be portable to a wide variety of computer architectures.

This research involves the development of a family of such data-parallel algorithms. Each method has its own advantages and disadvantages as will be discussed

below. However, all of the new methods developed here are highly efficient in both data-parallel and message-passing environments, and have been used successfully on a variety of flow simulations. There are three primary research areas in which these new methods will be useful. First, the simulation of very large problems, such as full three-dimensional vehicle simulations. This application is of great practical interest in the preliminary design phase of a hypersonic vehicle, where a large number of flight conditions must be simulated in the minimum possible time. The methods developed here are currently in use at NASA Langley to perform preliminary aerodynamic calculations on the proposed X-33 concept. Second, complex chemical models can be implemented, which require a large number of conservation equations, and therefore a large amount of memory to properly simulate.

Finally, with these new methods we are able to perform detailed simulations of complex physical processes at a resolution not previously possible. One of the interesting applications in this area is detailed shock interaction calculations. Shock interactions, such as those between the bow shock of a hypersonic vehicle and the shock waves generated by a wing or control surface, are a vital design consideration, due to the potentially high localized heat transfer rates in the interaction region. For this reason a large amount of work has been done in an attempt to classify and predict shock interaction phenomena. However, these flows can be extremely complex, and numerical studies have been unable to fully resolve the structure of the interaction region for many cases. Therefore, we will see that the underlying fluid dynamics of these interactions is not well understood. The final two chapters of this thesis describe work that has been done to simulate shock interaction flows on double-wedge and double-cone geometries. Chapter 8 details the results of a study of the parameter space of these interactions. An inviscid flow assumption is used in this Chapter, so that the underlying gas-dynamics of the interactions can be explored. Chapter 9 discusses the results of an ongoing project with Princeton University to study double-cone interactions both experimentally and numerically.

Chapter 2

Mathematical Formulation

2.1 Introduction

This chapter introduces the set of partial differential equations that describes the dynamics of a compressible flow. The equation of state and the models used for the transport of mass, momentum, and energy are also developed. Two sets of equations are presented, one for a thermally and calorically perfect gas, and another for reacting air in thermochemical nonequilibrium. Which model is used depends on the physics of the problem it is desired to solve. For example, air at one atmosphere behaves essentially as a perfect gas for temperatures below 800 K [Anderson (1989)]. Above this temperature the vibrational modes of the gas become partially excited and must be accounted for. At temperatures above 2500 K the oxygen molecules begin to dissociate, followed by nitrogen at about 4000 K. Above 9000 K ionization becomes important. In general these "onset" temperatures decrease with decreasing pressure.

2.2 Perfect Gas Flows

For flows in the continuum regime, that is, flows in which the mean free path is small compared to the length scale of interest, the Navier-Stokes equations are the governing equations of fluid mechanics. The Navier-Stokes equations are presented here for the case of a perfect gas. The equation set will be presented in conservation form, which will make it easier to discretize the problem in preparation for a numerical solution. The equations are written using index notation, in which a repeated index implies summation.

The mass conservation, or continuity, equation is

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_j}(\rho u_j) = 0, \tag{2.1}$$

where ρ is the density and u_j is the mass averaged velocity.

There are also n total momentum equations, where n is the number of spatial dimensions in the problem

$$\frac{\partial}{\partial t}(\rho u_i) + \frac{\partial}{\partial x_j}(\rho u_i u_j + p\delta_{ij}) + \frac{\partial}{\partial x_j}(\tau_{ij}) = 0, \tag{2.2}$$

where p is the thermodynamic pressure, δ_{ij} is the Krönecker delta, and τ_{ij} is the stress tensor. The form for the stress tensor is derived assuming a Newtonian fluid that obeys the Stokes hypothesis, and is given by

$$\tau_{ij} = -\mu(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}) + \frac{2}{3}\mu\frac{\partial u_k}{\partial x_k}\delta_{ij}, \tag{2.3}$$

where μ is the kinematic viscosity. For perfect air flow the kinematic viscosity is given by a Sutherland formula [White (1991)]

$$\mu = \frac{1.458 \times 10^{-6} T^{3/2}}{T + 110.3}, \tag{2.4}$$

where T is in degrees Kelvin, and μ is in SI units (kg/m/s).

The final equation is the total energy equation, given by

$$\frac{\partial E}{\partial t} + \frac{\partial}{\partial x_j}((E+p)u_j) + \frac{\partial q_j}{\partial x_j} + \frac{\partial}{\partial x_j}(\tau_{ij}u_i) = 0, \tag{2.5}$$

where E is the total energy per unit volume and q_j is the heat conduction vector, which is assumed to be given by Fourier's Heat Law

$$q_j = -\kappa\frac{\partial T}{\partial x_j}. \tag{2.6}$$

κ is the thermal conductivity, which is found by assuming a constant Prandtl number, Pr

$$Pr = \frac{\mu c_p}{\kappa}, \tag{2.7}$$

where c_p is the constant pressure specific heat and $Pr = 0.72$ for air.

This equation set is sufficient to describe the dynamics of many compressible supersonic and even hypersonic flows, as long as the temperature remains below the "onset" values discussed in the previous section.

2.3 Reacting Air Flows

For high temperature flows, the standard mass, momentum, and energy equations are not sufficient to describe the thermochemical state of the gas. The temperatures associated with these flows can lead to a large number of chemical and energy relaxation processes which are not important at lower temperatures. In addition these processes can occur on a time scale similar to the flow time, leading to a large amount of thermal and chemical nonequilibrium in the flow. In order to simulate these flows all of these finite rate processes must be accurately modeled. Therefore an extended set of equations must be used to account for the chemical reactions and energy relaxation taking place.

This section presents the conservation equations governing the reacting air flows that are discussed in this work. We restrict ourselves to non-ionized flows. In addition, we assume that the translational and rotational states of the gas are in equilibrium at a common temperature T, while the vibrational energy can be characterized by a separate temperature T_v. Based on these assumptions, the models used for the chemical kinetics, exchange between energy modes of the gas, and the transport coefficients are discussed.

2.3.1 Governing Equations

The Navier-Stokes equations for reacting flows have been derived in many places, including Vincenti and Kruger (1986) and Clarke and McChesney (1975), and thus a replication of this procedure will not be included here. Lee (1986) and Candler (1988) have each provided an excellent summary of this work as it pertains to the numerical solution of flows in thermochemical nonequilibrium.

For a chemically reacting mixture of n_s species, of which n_d are diatomic and the rest are monatomic, there are n_s mass conservation equations of the form

$$\frac{\partial \rho_s}{\partial t} + \frac{\partial}{\partial x_j}(\rho_s u_j) + \frac{\partial}{\partial x_j}(\rho_s v_{sj}) = w_s, \tag{2.8}$$

where ρ_s is the density of species s, v_{sj} is the diffusion velocity of species s, and w_s is the source term representing the formation and destruction of species s due to chemical reactions. The form of the chemical source term will be discussed in section 2.3.4.

The total momentum equations are unchanged from the perfect gas case

$$\frac{\partial}{\partial t}(\rho u_i) + \frac{\partial}{\partial x_j}(\rho u_i u_j + p\delta_{ij}) + \frac{\partial}{\partial x_j}(\tau_{ij}) = 0, \tag{2.9}$$

The stress tensor τ_{ij} is defined as in Eq. (2.3), except that μ is now the kinematic viscosity of the mixture. The method used to calculate the mixture viscosity is discussed in section 2.3.3.

The total energy equation is given by

$$\frac{\partial E}{\partial t} + \frac{\partial}{\partial x_j}((E+p)u_j) + \frac{\partial}{\partial x_j}(q_j + q_{vj}) + \frac{\partial}{\partial x_j}(\tau_{ij}u_i) + \sum_{s=1}^{n_s} \frac{\partial}{\partial x_j}(h_s v_{sj}) = 0, \tag{2.10}$$

where h_s is the species enthalpy per unit volume, and q_{vj} is the vibrational heat conduction term. Both heat conduction terms are given by Fourier's Law

$$q_j = -\kappa \frac{\partial T}{\partial x_j}, \qquad q_{vj} = -\kappa_v \frac{\partial T_v}{\partial x_j}, \tag{2.11}$$

where κ and κ_v are the mixture thermal conductivities, discussed in section 2.3.3.

The final equation required is a conservation equation for the vibrational energy per unit volume, E_v. Note that it is possible to construct separate conservation equations for each individual vibrational energy mode of the gas, and that each mode would in principle be governed by a different temperature. However, we assume for this work that the energy exchange between vibrational modes of the gas is fast enough that all vibrational modes are in equilibrium with each other, and

are thus governed by a single vibrational temperature. Thus, only one vibrational energy conservation equation is required. For an excellent derivation of the more general case, refer to Olejniczak (1997). The vibrational energy equation is given as

$$\frac{\partial E_v}{\partial t} + \frac{\partial}{\partial x_j}(E_v u_j) + \frac{\partial}{\partial x_j}(q_{vj}) + \sum_{s=1}^{n_d} \frac{\partial}{\partial x_j}(\rho_s e_{vs} v_{sj}) = w_v, \tag{2.12}$$

where e_{vs} is the vibrational energy per unit mass of species s, and w_v is the source term for vibrational energy, which will be discussed in section 2.3.5.

2.3.2 Equations of State

The total energy of the gas per unit volume can be defined as

$$E = \sum_{s=1}^{n_s} \rho_s c_{vs} T + \frac{1}{2}\rho u_i u_i + E_v + \sum_{s=1}^{n_s} \rho_s h_s^\circ. \tag{2.13}$$

c_{vs} is the translational-rotational specific heat of species s, given by

$$c_{vs} = c_{\mathrm{vtr}\,s} + c_{\mathrm{vrot}\,s} \tag{2.14}$$

The translational and rotational specific heats are given by

$$\begin{aligned} c_{\mathrm{vtr}\,s} &= \frac{3}{2}\frac{R}{M_s}, \\ c_{\mathrm{vrot}\,s} &= \frac{R}{M_s} \qquad s \leq nd, \end{aligned} \tag{2.15}$$

where R is the universal gas constant and M_s is the species molecular weight. Thus the first term of Eq. (2.13) represents the energy due to random thermal translational-rotational motion of the molecules. The second term represents the kinetic energy of the gas. The third term is the contribution of the vibrational energy modes of the gas. The vibrational energy per unit volume of each species (E_{vs}) can be related to a vibrational temperature, T_v, assuming a Boltzmann distribution exists, using a harmonic oscillator expression

$$E_{vs} = \rho_s \frac{R}{M_s} \frac{\theta_{vs}}{\exp(\theta_{vs}/T_v) - 1}, \tag{2.16}$$

where θ_{vs} is the characteristic temperature of vibration for species s. Values of θ_{vs} are given in the Appendix. The total vibrational energy per unit volume is then obtained simply from

$$E_v = \sum_{s=1}^{n_d} E_{vs}. \tag{2.17}$$

Note that in the case where there is more than one diatomic species present it is not possible to solve Eq. (2.17) directly for T_v, and therefore the vibrational temperature for a given energy must be computed iteratively. The final term of Eq. (2.13) represents the chemical heat of formation associated with each species, which accounts for the energy stored in chemical bonds. The values of the species formation enthalpy, h_s°, are given in the Appendix. In this study only the ground electronic states of the chemical species are considered. Hence, there are no terms in Eq. (2.13) due to excited electronic states.

Finally, the pressure and translational-rotational temperature of the gas are related through the perfect gas law, where the thermodynamic pressure is determined from the partial pressures of each species using Dalton's Law

$$p = \sum_{s=1}^{n_s} \rho_s \frac{R}{M_s} T. \tag{2.18}$$

2.3.3 Species and Mixture Transport Properties

In this section we derive expressions for the species and mixture values of viscosity and thermal conductivity. The viscosity coefficient for each chemical species, μ_s, is determined using curve fits obtained by Blottner *et al.* (1971)

$$\mu_s = 0.1 \exp[(A_s \ln T + B_s) \ln T + C_s], \tag{2.19}$$

where A_s, B_s, and C_s are species dependent curve fit parameters given in the Appendix. These fits are valid up to 10,000 K, which is sufficient for this work. If the flows of interest include species not in the Blottner data or extend outside the valid temperature range, species viscosity coefficients can also be computed from kinetic theory using cross section data [Svehla (1962) & Gupta *et al.* (1990)].

The thermal conductivities for the translational, rotational, and vibrational energy modes are determined from an Eucken relation [Vincenti and Kruger (1986)], assuming that the transport of translational energy is correlated to the velocity of the molecules, but the transport of internal energies shows no such correlation. Based on this assumption, we have

$$\kappa_{ts} = \frac{5}{2}\mu_s c_{v\mathrm{tr}\,s}, \quad \kappa_{rs} = \mu_s c_{v\mathrm{rot}\,s}, \quad \kappa_{vs} = \mu_s c_{v\mathrm{vib}\,s}, \tag{2.20}$$

for the translational, rotational, and vibrational thermal conductivities, respectively. Since the vibrational modes of the gas are in general not fully excited, they do not contribute a full $\frac{1}{2}kT$ of energy per degree of freedom, and therefore the vibrational specific heat must be computed using

$$c_{v\mathrm{vib}\,s} = \frac{\partial e_v}{\partial T_v}. \tag{2.21}$$

Once the species viscosity and heat conduction coefficients have been determined, mixture values must be calculated. This is accomplished using Wilke's mixing rule [Wilke (1950)],

$$\mu = \sum_s \frac{X_s \mu_s}{\phi_s}, \quad \kappa = \sum_s \frac{X_s \kappa_s}{\phi_s}, \tag{2.22}$$

where

$$X_s = \frac{c_s M}{M_s}, \quad M = \Big(\sum_s \frac{c_s}{M_s}\Big)^{-1}, \tag{2.23}$$

$$\phi_s = \sum_r X_r \left[1 + \sqrt{\frac{\mu_s}{\mu_r}}\left(\frac{M_r}{M_s}\right)^{1/4}\right]^2 \left[\sqrt{8\Big(1 + \frac{M_s}{M_r}\Big)}\right]^{-1}, \tag{2.24}$$

where $c_s = \rho_s/\rho$ is the mass fraction of species s.

In general, mass diffusion in the flow is driven by gradients of concentration, pressure, and temperature. However, for most flows of interest, only the term due to the concentration gradient is significant. Thermal diffusion can become important within the shock layer, or in other regions where the temperature gradients become very large, but in these regions the continuum assumption generally becomes suspect [Boyd *et al.* (1995)], since the length scale over which the gradients occur is

comparable to the mean free path. Therefore, in these regions the Navier-Stokes equations themselves are invalid, and particle based methods (such as Direct Simulation Monte Carlo) should be used. Therefore, for this work the mass diffusion is assumed to be driven entirely by concentration gradients. The diffusion term can then be written based on Fick's law as

$$\rho_s v_{sj} = \rho \mathcal{D}_s \frac{\partial c_s}{\partial x_j}. \tag{2.25}$$

A single diffusion coefficient for all species, $\mathcal{D}$, is then derived assuming a constant Lewis number, Le, defined by

$$Le = \mathcal{D}\frac{\rho c_p}{\kappa}, \tag{2.26}$$

where c_p is the constant pressure translational-rotational specific heat, and the Lewis number is usually taken to be 1.4 for air. Multicomponent diffusion coefficients $\mathcal{D}_s$ can be calculated from the fits of Blottner *et al.* (1971) or Goopta *et al.* (1990), however this is much more complicated to implement in a numerical code, and has been shown to make only a small difference in most air flows of interest [Olejniczak *et al.* (1996b)]. The effects of multicomponent diffusion would need to be included, however, if very light species such as hydrogen or helium are present in the flowfield.

2.3.4 Chemical Source Terms

Each species mass conservation equation has a source term (w_s) due to the formation and destruction of species s from chemical reactions. The sum of these source terms should of course be zero, since mass must be conserved. The form of this chemical source term is discussed in this section.

For high temperature air with no ionization, there are five important chemical reactions which must be considered, involving five distinct chemical species: N_2, O_2, NO, N, and O. These reactions are summarized here.

$$N_2 + M \rightleftharpoons 2N + M$$
$$O_2 + M \rightleftharpoons 2O + M$$
$$NO + M \rightleftharpoons N + O + M$$
$$N_2 + O \rightleftharpoons NO + N$$
$$NO + O \rightleftharpoons O_2 + N$$

In this reaction set the bi-directional arrows indicate that the reaction can proceed in either the forward or backward direction. All of the reactions are presented so that they are endothermic in the forward direction. The first three reactions represent collision induced dissociation of the diatomic species N_2, O_2, and NO respectively. In these reactions M is a collision partner, which provides the energy necessary to break the chemical bond but is otherwise unaltered in the reaction. The collision partner may be any of the species present in the flow. The reverse of these reactions is three body recombination, where the collision partner, now an energy absorber, is required for energy conservation. The final two reactions in the set are the Zeldovich exchange reactions. These Zeldovich reactions are the primary source of nitric oxide (NO) in a hypersonic flowfield.

Each of these reactions is governed by forward and backward rate coefficients, k_{fm} and k_{bm} respectively. Therefore we may write the rate of each reaction as a sum of the forward and backward rates

$$\begin{aligned}
\mathcal{R}_1 &= \sum_m \left[k_{b1m} \frac{\rho_N}{M_N} \frac{\rho_N}{M_N} \frac{\rho_m}{M_m} - k_{f1m} \frac{\rho_{N_2}}{M_{N_2}} \frac{\rho_m}{M_m} \right] \\
\mathcal{R}_2 &= \sum_m \left[k_{b2m} \frac{\rho_O}{M_O} \frac{\rho_O}{M_O} \frac{\rho_m}{M_m} - k_{f2m} \frac{\rho_{O_2}}{M_{O_2}} \frac{\rho_m}{M_m} \right] \\
\mathcal{R}_3 &= \sum_m \left[k_{b3m} \frac{\rho_N}{M_N} \frac{\rho_O}{M_O} \frac{\rho_m}{M_m} - k_{f3m} \frac{\rho_{NO}}{M_{NO}} \frac{\rho_m}{M_m} \right] \\
\mathcal{R}_4 &= k_{b4} \frac{\rho_{NO}}{M_{NO}} \frac{\rho_N}{M_N} - k_{f4} \frac{\rho_{N_2}}{M_{N_2}} \frac{\rho_O}{M_O} \\
\mathcal{R}_5 &= k_{b5} \frac{\rho_{O_2}}{M_{O_2}} \frac{\rho_N}{M_N} - k_{f5} \frac{\rho_{NO}}{M_{NO}} \frac{\rho_O}{M_O}.
\end{aligned} \tag{2.27}$$

Now the chemical source terms can be expressed in terms of the individual reaction

rates, $\mathcal{R}_s$

$$\begin{aligned} w_{N_2} &= M_{N_2}(\mathcal{R}_1 + \mathcal{R}_4) \\ w_{O_2} &= M_{O_2}(\mathcal{R}_2 - \mathcal{R}_5) \\ w_{NO} &= M_{NO}(\mathcal{R}_3 - \mathcal{R}_4 + \mathcal{R}_5) \\ w_N &= M_N(-2\mathcal{R}_1 - \mathcal{R}_3 - \mathcal{R}_4 - \mathcal{R}_5) \\ w_O &= M_O(-2\mathcal{R}_2 - \mathcal{R}_3 + \mathcal{R}_4 + \mathcal{R}_5). \end{aligned} \tag{2.28}$$

As a check we note that the sum of the source terms over all n_s species is identically zero, as required by conservation of mass, and that elemental conservation holds for nitrogen and oxygen.

Now we require expressions for the forward and backward rate coefficients in Eq. (2.27) above. These rate coefficients are in general affected by the level of thermal nonequilibrium in the flow, and thus can be represented as

$$k = Vk^*, \tag{2.29}$$

where k^* is the reaction rate in thermal equilibrium, and V is some as yet undetermined factor which accounts for the effects of thermal nonequilibrium. The functional form that V should take depends on the physical coupling between the chemistry and vibrational energy modes of the gas, and V can thus be referred to as the vibration-dissociation coupling factor. There are many vibration-dissociation coupling models in the literature; however, only a few are used regularly. These include the Marrone and Treanor (1963) CVDV model, the Macheret and Rich (1993) model, and the Park (1986) model. Of these, the Park model has been used most extensively, since it is the simplest to implement and has been shown to give adequate answers for a variety of flow conditions. However, none of these models satisfactorily address the issue of vibration-dissociation coupling under all flow conditions, and more research is required in this area. For the purposes of this work, only the Park model has been considered, and a brief discussion of this model is included here.

Park introduces an effective temperature, T_a, which is some function of the translational-rotational and vibrational temperatures, and assumes that the reaction rates are functions of this effective temperature. The forward rates are then typically expressed using a modified Arrhenius form, as in

$$k_{fm}(T_a) = C_{fm} T_a^{\eta_m} \exp\left(\theta_{dm}/T_a\right), \tag{2.30}$$

where the constants C_{fm}, η_m, and the reaction energy θ_{dm} have been determined by curve fits to experimental data over a given temperature range [Park (1988)] and are given in the Appendix. The backward rates are then calculated using the principle of detailed balance

$$k_{bm}(T_a) = \frac{k_{fm}(T_a)}{K_{\mathrm{eq}\,m}}. \tag{2.31}$$

The expression for the equilibrium constant, K_{eq}, is also a curve fit to experimental data, and is taken to be a function only of T

$$K_{\mathrm{eq}\,m} = C_m \exp(A_{1m} + A_{2m} Z + A_{3m} Z^2 + A_{4m} Z^3 + A_{5m} Z^4), \tag{2.32}$$

where $Z = 10,000/T$. The constants C_m and A_{nm} are given by Park (1985), and reproduced in the Appendix.

It is now necessary to define the effective temperature, T_a. Park recommends using the geometric average of the translational-rotational and vibrational temperatures for the forward rates of the dissociation reactions, as in

$$T_a = \sqrt{T T_v}. \tag{2.33}$$

However, the backward (recombination) rates are taken to be functions of T only. It has also been reasoned that the rates for the two exchange reactions should be functions only of the relative speed of the collision, and thus the effective temperature for the exchange reactions should also be simply T. This assumption is used in this work, although recent work by Bose and Candler (1996), using a quasi-classical

trajectory analysis of the first Zeldovich reaction, suggests that the rate is indeed dependent on the reactant vibrational energy.

2.3.5 Vibrational Energy Source Term

The source term in the vibrational energy equation (w_v) is due to the exchange of energy between the translational-rotational and vibrational modes of the gas, as well as the creation and destruction of diatomic molecules with some vibrational energy. The form of this source term is discussed in this section.

In general there are many energy exchange processes which are important, including exchange between the translational and vibrational modes ($V-T$) exchange between the rotational and vibrational modes ($V-R$), and exchange between various vibrational modes of the separate species ($V-V$). However, under the simplifying assumptions outlined in this chapter, $V-V$ exchanges are taken to be fast enough to equilibrate all of the vibrational modes of the gas to the same vibrational temperature T_v. In addition, $V-T$ and $V-R$ exchanges can be lumped into a single exchange rate, signified by Q_{T-V}. For the conditions considered here, the rate of energy exchange can be described by the Landau-Teller model [Vincenti and Kruger (1986)], which is expressed as

$$Q_{T-V\,\mathrm{L-T}} = \rho_s \frac{e^*_{vs} - e_{vs}}{<\tau_{s\,\mathrm{L-T}}>}, \tag{2.34}$$

where e^*_{vs} is the vibrational energy per unit mass of species s evaluated at the local translational temperature, and $<\tau_{s\,\mathrm{L-T}}>$ is the molar averaged Landau-Teller relaxation time, given by Lee (1985)

$$<\tau_{s\,\mathrm{L-T}}> = \frac{\sum_r X_r}{\sum_r X_r/\tau_{sr\,\mathrm{L-T}}}, \tag{2.35}$$

where X_r has been defined previously in Eq. (2.23), and $\tau_{sr\,\mathrm{L-T}}$ is the Landau-Teller inter-species relaxation time, given by Millikan and White (1963) as

$$\begin{aligned} \tau_{sr\,\mathrm{L-T}} &= \frac{1}{p} exp[A_{sr}(T^{-1/3} - 0.015\mu_{sr}^{1/4}) - 18.42], \qquad (p \text{ in atm}) \\ A_{sr} &= 1.16 \times 10^{-3} \mu_{sr}^{1/2} \theta_{vs}^{4/3}, \\ \mu_{sr} &= M_s M_r/(M_s + M_r). \end{aligned} \tag{2.36}$$

The second portion of the vibrational energy source term is due to the fact that as diatomic molecules are created or destroyed in chemical reactions, they carry with them some vibrational energy. The form of this term is dependent on the vibration-dissociation coupling model used, since we must know the reaction rates as a function of the vibrational states of the product and reactant molecules. For this work we assume that, on the average, molecules are created and destroyed with a vibrational energy given by the local value of T_v. Therefore, the rate of vibrational energy production due to the creation or destruction of diatomic species s is just

$$Q_{vs\,\mathrm{chem}} = w_s e_{vs}. \tag{2.37}$$

The total contribution to the vibrational source term due to chemical reactions can be obtained by summing over all diatomic species, and the terms due to energy exchange and chemical reactions are then summed to give the final form of the vibrational energy source term

$$w_v = Q_{T-V\,\mathrm{L-T}} + \sum_{s=1}^{n_d} w_s e_{vs}. \tag{2.38}$$

Chapter 3

Numerical Method

3.1 Introduction

This chapter discusses the numerical approach used to discretize and solve the system of non-linear partial differential equations developed in the previous chapter. The finite volume formulation is introduced, and the methods used to evaluate the inviscid and viscous flux terms are presented. Finally, the basic starting equation used to advance the solution forward in time is derived.

3.2 Finite Volume Formulation

This section outlines the discretization of the equation set using the finite volume technique. The conservation equations presented in Chapter 2 must first be converted to a general curvilinear coordinate system, using the transformation

$$\begin{aligned} \frac{\partial}{\partial \xi} &= \frac{\partial}{\partial x}\frac{\partial x}{\partial \xi} + \frac{\partial}{\partial y}\frac{\partial y}{\partial \xi}, \\ \frac{\partial}{\partial \eta} &= \frac{\partial}{\partial x}\frac{\partial x}{\partial \eta} + \frac{\partial}{\partial y}\frac{\partial y}{\partial \eta}, \end{aligned} \tag{3.1}$$

where ξ is defined as the body-tangential and η the body-normal direction. After this transformation, the governing equations can be written in conservation form as

$$\frac{\partial U}{\partial t} + \frac{\partial \tilde{F}}{\partial \xi} + \frac{\partial \tilde{G}}{\partial \eta} = W, \tag{3.2}$$

where U is the vector of conserved quantities, $\tilde{F}$ and $\tilde{G}$ are the flux vectors in the ξ and η directions, and W is the vector of source terms. In this curvilinear coordinate system $\tilde{F}$ and $\tilde{G}$ are functionally equivalent. Details of this transformation can be

found in Candler (1988) or Hirsch (1991). The resulting flux vectors can be split into convective (inviscid) and viscous parts

$$\tilde{F} = F + F_v, \qquad \tilde{G} = G + G_v. \tag{3.3}$$

Under the approximations listed in the previous chapter, the conserved quantity, flux, and source term vectors for a two-dimensional thermochemical nonequilibrium air flow can be written as

$$U = \begin{pmatrix} \rho_1 \\ \rho_2 \\ \vdots \\ \rho_s \\ \rho u \\ \rho v \\ E_v \\ E \end{pmatrix}, \quad F = G = \begin{pmatrix} \rho_1 u' \\ \rho_2 u' \\ \vdots \\ \rho_s u' \\ \rho u u' + p s_x \\ \rho v u' + p s_y \\ E_v u' \\ (E+p)u' \end{pmatrix}$$

$$F_v = G_v = -\begin{pmatrix} \rho \mathcal{D}(\frac{\partial c_1}{\partial x} s_x + \frac{\partial c_1}{\partial y} s_y) \\ \rho \mathcal{D}(\frac{\partial c_2}{\partial x} s_x + \frac{\partial c_2}{\partial y} s_y) \\ \vdots \\ \rho \mathcal{D}(\frac{\partial c_s}{\partial x} s_x + \frac{\partial c_s}{\partial y} s_y) \\ \tau_{xx} s_x + \tau_{xy} s_y \\ \tau_{xy} s_x + \tau_{yy} s_y \\ (\rho \sum_{nd} e_{vs} \mathcal{D} \frac{\partial c_s}{\partial x} + q_{vx}) s_x + \\ (\rho \sum_{nd} e_{vs} \mathcal{D} \frac{\partial c_s}{\partial y} + q_{vy}) s_y \\ (\tau_{xx} u + \tau_{xy} v + q_x + q_{vx} + \rho \sum_{ns} h_s \mathcal{D} \frac{\partial c_s}{\partial x}) s_x + \\ (\tau_{xy} u + \tau_{yy} v + q_y + q_{vy} + \rho \sum_{ns} h_s \mathcal{D} \frac{\partial c_s}{\partial y}) s_y \end{pmatrix}, \quad W = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_s \\ 0 \\ 0 \\ w_v \\ 0 \end{pmatrix}, \tag{3.4}$$

where s_x, s_y are the local direction cosines of the curvilinear coordinate system, and u' is the normal velocity component, defined as

$$u' = u s_x + v s_y. \tag{3.5}$$

A similar result is obtained for perfect gas flows. In the finite volume formulation we solve Eqs. (3.2) by integrating them over an arbitrary control volume V. Green's

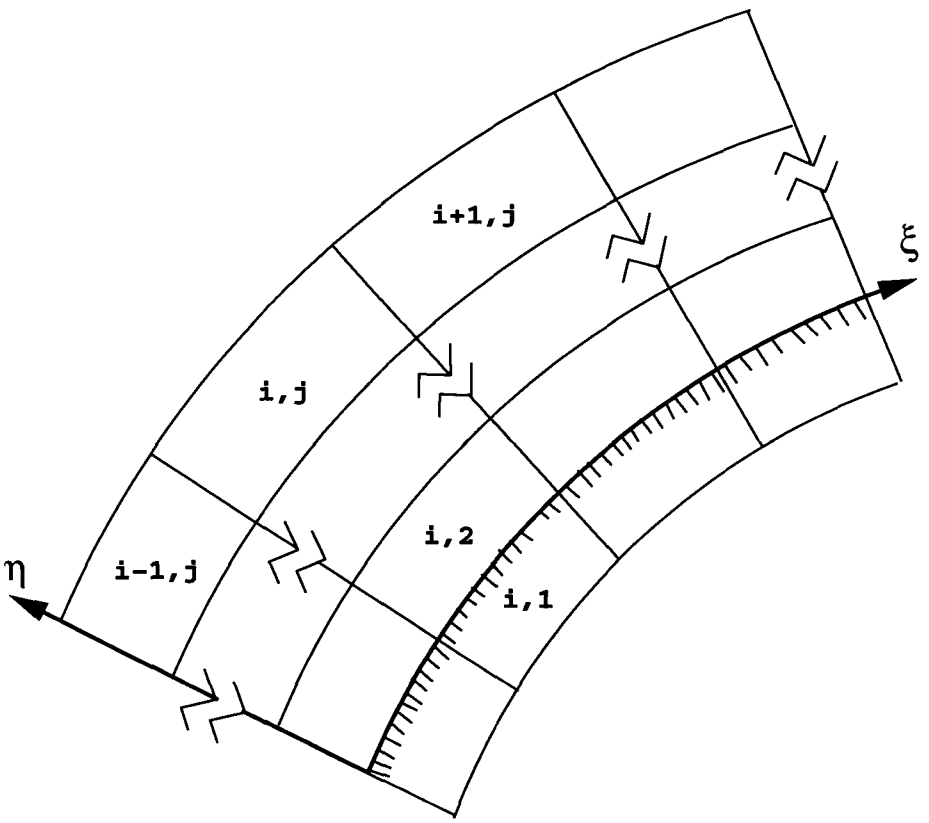


Fig. 3.1 Schematic diagram of a typical two-dimensional finite volume grid. A solid wall is located between $j=1$ and $j=2$.

Theorem is then used to convert the volume integral containing the flux vectors into a surface integral, resulting in

$$\frac{\partial U}{\partial t} + \frac{1}{V}\int_s \tilde{F} \cdot dS = W, \tag{3.6}$$

where V is the volume, $\tilde{F} \cdot dS$ is the total flux through the surface S, and U and W are assumed constant within the control volume. Now, we discretize the problem by dividing the computational space into a series of volume elements. This is typically done by laying out a regular curvilinear mesh over the solution domain, resulting in a structured grid on which, for two-dimensional problems, each computational cell is a four-sided polygon. A sample portion of such a structured grid is shown in Fig. (3.1). In the figure, the grid directions are defined such that i increases in the ξ-direction and j increases in the η-direction. In the finite volume method flow variables are stored at the cell centers, rather than the nodes. Data at the

cell faces is then found by some sort of averaging procedure. Because of this, an extra line of cells, called "dummy" cells, is usually included along the boundaries of the grid. These dummy cells are used to store the necessary boundary conditions for the problem. In Fig. (3.1) there is a solid wall boundary at $j = 3/2$, and the $j = 1$ cells are the dummy cells located inside the solid wall. The interior of the computational domain then begins at $j=2$.

For a 2-D volume element i, j we can represent the time rate of change of U as the sum of the fluxes through the four cell faces and the source of U within the volume, as in

$$\frac{\partial U_{i,j}}{\partial t} = -\frac{1}{V_{i,j}}(\tilde{F}_{i+1/2,j}S_{i+1/2,j} - \tilde{F}_{i-1/2,j}S_{i-1/2,j} + \tilde{G}_{i,j+1/2}S_{i,j+1/2} - \tilde{G}_{i,j-1/2}S_{i,j-1/2}) + W_{i,j}, \tag{3.7}$$

The $\pm 1/2$ indices on the flux and surface area terms indicate that they should be evaluated at the appropriate cell face. This expression represents the discretized form of the conservation equations. An analogous set of expressions can easily be derived for three-dimensional flow, where in that case the volume elements are six-sided polyhedra.

3.3 Evaluation of the Fluxes

There are many different methods in the literature for evaluating the right-hand side of Eq. (3.7). The solution is complicated by the fact that the inviscid fluxes are hyperbolic, while the viscous fluxes are elliptic in nature. For this reason the viscous fluxes are typically evaluated using simple central differencing [Hirsch (1991)], while a more sophisticated approach is required for the evaluation of the inviscid flux. Most modern solution methods for the inviscid flux terms take advantage of the fact that there are definable characteristic directions in the flow, and thus the required derivatives should be taken following these characteristic directions. This is commonly referred to as upwind biasing. One such method that has been widely used due to its simplicity and robustness is Steger-Warming (1981)

flux-vector splitting. A brief derivation of this method is included here not only because it is the primary method used in this research, but also because some of the concepts introduced will be useful in the following chapters when the implicit time advancement algorithms developed in this work are presented.

3.3.1 Inviscid Fluxes

First, we focus on the inviscid problem. We will return to the evaluation of the viscous fluxes in the next section. The first step in the evaluation of the inviscid fluxes is to use the fact that they are homogeneous in the vector of conserved quantities U, such that

$$F(\lambda U) = \lambda F(U), \tag{3.8}$$

where λ is any scalar. This can easily be shown from the governing equations developed in Chapter 2. From this it is possible to linearize the flux vector, using

$$\begin{aligned} F &= \left(\frac{\partial F}{\partial U}\right)U = AU, \\ G &= \left(\frac{\partial G}{\partial U}\right)U = BU, \end{aligned} \tag{3.9}$$

where A and B are the inviscid flux Jacobian matrices in the ξ and η directions, respectively.

We then split the fluxes into positively-moving and negatively-moving components using an upwind biasing scheme. In the Steger-Warming formulation the fluxes are split according to the sign of the eigenvalues of the Jacobians. Therefore, we must calculate the eigenvalues and eigenvectors of A. In order to do this we first break the Jacobian matrices into components that will be easier to work with

$$A = \frac{\partial U}{\partial V}\frac{\partial V}{\partial U}\frac{\partial F}{\partial V}\frac{\partial V}{\partial U}, \tag{3.10}$$

where V is a vector of primitive variables, introduced purely as a convenience to simplify the computation of A. The choice of V is not unique, but for the inviscid fluxes it is convenient to use

$$V = \left(\rho_1, \rho_2, \ldots, \rho_s, u, v, e_v, p\right)', \tag{3.11}$$

where e_v is the total vibrational energy per unit mass. $\frac{\partial U}{\partial V}$ and $\frac{\partial V}{\partial U}$ are the transformation matrices between primitive and conserved variables, and are denoted by S^{-1} and S respectively. This transformation is made because the remaining matrix $\frac{\partial V}{\partial U}\frac{\partial F}{\partial V}$ is much easier to diagonalize than A. The diagonalization itself is now straightforward, using basic linear algebra

$$\frac{\partial V}{\partial U}\frac{\partial F}{\partial V} = R^{-1}\Lambda R, \tag{3.12}$$

where Λ is the diagonal matrix of eigenvalues of the system, and R^{-1} and R are the left and right eigenvector matrices. These required matrices are given in the Appendix for a perfect gas flow, and the results are easily extended to reacting flows as well.

The flux vector can now be split into positively-moving and negatively-moving components, based on the sign of the eigenvalues. The split fluxes F_+ and F_- are defined by

$$\begin{aligned} F_+ &= S^{-1}R^{-1}\Lambda_+ RSU = A_+ U, \\ F_- &= S^{-1}R^{-1}\Lambda_- RSU = A_- U, \end{aligned} \tag{3.13}$$

where Λ_+ and Λ_- are the split eigenvalue matrices consisting of only the positive and negative eigenvalues, respectively. A_+ and A_- are the corresponding positive and negative Jacobians. The total inviscid flux vector through each cell face can then be expressed as the sum of positively and negatively moving components, as in

$$\begin{aligned} F_{i+\frac{1}{2},j} &= F_{+i+\frac{1}{2},j} + F_{-i+\frac{1}{2},j}, \\ G_{i,j+\frac{1}{2}} &= G_{+i,j+\frac{1}{2}} + G_{-i,j+\frac{1}{2}}. \end{aligned} \tag{3.14}$$

The split fluxes at each cell face are calculated using

$$\begin{aligned} F_{+\,i+\frac{1}{2},j} &= A_{+\,i,j} U_{i,j}, \\ F_{-\,i+\frac{1}{2},j} &= A_{-\,i+1,j} U_{i+1,j}, \end{aligned} \tag{3.15}$$

where A and U are evaluated based on data at the upwind cell center. However, in practice we usually evaluate the Jacobian matrices at the cell faces rather than the

upwind cell centers, as in

$$\begin{aligned} F_{+i+\frac{1}{2},j} &= A_{+i+\frac{1}{2},j} U_{i,j}, \\ F_{-i+\frac{1}{2},j} &= A_{-i+\frac{1}{2},j} U_{i+1,j}. \end{aligned} \tag{3.16}$$

This modification of the original Steger-Warming method was proposed by MacCormack and Candler (1989), and has been shown to greatly reduce the amount of numerical dissipation in the method. While this modified approach works well in regions of weak gradients, additional dissipation is required to capture strong gradients, such as shock waves. Therefore, a hybrid approach is usually used, in which the Jacobian matrices are evaluated using a pressure weighted average of quantities in the adjacent cells. In this way the method will smoothly switch from modified to true Steger-Warming in regions of high pressure gradients. If we then substitute the split fluxes into Eq. (3.7), we get the standard upwind finite volume representation of the inviscid problem

$$\begin{aligned} \frac{\partial U_{i,j}}{\partial t} = -\frac{1}{V_{i,j}} \Big\{ & \left(A_{+i+\frac{1}{2},j} S_{i+\frac{1}{2},j} U_{i,j} - A_{+i-\frac{1}{2},j} S_{i-\frac{1}{2},j} U_{i-1,j} \right) \\ & - \left(A_{-i-\frac{1}{2},j} S_{i-\frac{1}{2},j} U_{i,j} - A_{-i+\frac{1}{2},j} S_{i+\frac{1}{2},j} U_{i+1,j} \right) \\ & + \left(B_{+i,j+\frac{1}{2}} S_{i,j+\frac{1}{2}} U_{i,j} - B_{+i,j-\frac{1}{2}} S_{i,j-\frac{1}{2}} U_{i,j-1} \right) \\ & - \left(B_{-i,j-\frac{1}{2}} S_{i,j-\frac{1}{2}} U_{i,j} - B_{-i,j+\frac{1}{2}} S_{i,j+\frac{1}{2}} U_{i,j+1} \right) \Big\} + W_{i,j}. \end{aligned} \tag{3.17}$$

The Steger-Warming method as formulated has only first-order spatial accuracy, although it is possible to achieve second-order accuracy in regions of weak gradients with a minor modification. While this is sufficient for many problems, a higher order accurate scheme may be preferred for certain applications. In general schemes with higher order spatial accuracy are less dissipative, and can produce a more resolved solution on a given computational mesh. Therefore, if it is desired to resolve flow features that can be easily smeared by the numerical dissipation present in the scheme, such as weak shear layers, a high order accurate scheme may be required. Unfortunately, while there are a great many high order accurate schemes available, most suffer from numerical difficulties when used to simulate flows in

which strong shocks are present. In certain cases these schemes require so many extra time steps to achieve a steady-state solution that an equivalent solution can be obtained in less time on a more resolved grid using a first-order method [Olejniczak *et al.* (1996a)]. Several second-order accurate methods have been used in this research when required, including a Harten-Yee (1989) upwind total variation diminishing (TVD) method, and a symmetric TVD method proposed by Gnoffo (1990).

3.3.2 Viscous Fluxes

As discussed earlier, the elliptic nature of the viscous fluxes makes them much simpler to compute than the inviscid component. The viscous flux vector outlined in Eq. (3.4) is simply evaluated at the required cell faces using central differencing, resulting in a flux through each face of the form

$$F_{v\,i+\frac{1}{2},j} S_{i+\frac{1}{2},j}. \tag{3.18}$$

The contribution of the viscous fluxes is then summed through each of the four cell faces (six in three-dimensional flow) and added to Eq. (3.17) above. A detailed discussion of the formulation of these fluxes and the calculation of the required derivatives can be found in Hirsch (1991).

3.4 Time Advancement

Up to this point we have made no effort to evaluate the time derivative on the left-hand side of Eq. (3.17). This term must also be discretized in some manner in order to time-march the solution toward a steady-state answer. The simplest form of time advancement that is used is first-order forward Euler differencing, in which the time derivative is expressed

$$\frac{\partial U_{i,j}}{\partial t} \simeq \frac{U_{i,j}^{n+1} - U_{i,j}^{n}}{\Delta t} = \frac{\Delta U_{i,j}^{n}}{\Delta t}. \tag{3.19}$$

In this expression, the superscript denotes the time level of the solution, which is advanced from the current time n to the next level $n+1$. Expressions for the

time derivative may of course be obtained to any order of accuracy, using Runge-Kutta or other differencing methods. The resulting expression is then substituted directly into Eq. (3.17) and the solution is marched forward in time by discrete time steps. For the Euler method detailed above, the explicit formulation of the inviscid problem becomes

$$\begin{aligned}\Delta U^n_{i,j} = -\frac{\Delta t}{V_{i,j}}\Big\{&\left(A_{+i+\frac{1}{2},j}S_{i+\frac{1}{2},j}U_{i,j} - A_{+i-\frac{1}{2},j}S_{i-\frac{1}{2},j}U_{i-1,j}\right)\\ &-\left(A_{-i-\frac{1}{2},j}S_{i-\frac{1}{2},j}U_{i,j} - A_{-i+\frac{1}{2},j}S_{i+\frac{1}{2},j}U_{i+1,j}\right)\\ &+\left(B_{+i,j+\frac{1}{2}}S_{i,j+\frac{1}{2}}U_{i,j} - B_{+i,j-\frac{1}{2}}S_{i,j-\frac{1}{2}}U_{i,j-1}\right)\\ &-\left(B_{-i,j-\frac{1}{2}}S_{i,j-\frac{1}{2}}U_{i,j} - B_{-i,j+\frac{1}{2}}S_{i,j+\frac{1}{2}}U_{i,j+1}\right)\Big\}^n + \Delta t W^n_{i,j},\end{aligned} \tag{3.20}$$

where $\Delta U^n_{i,j}$ is the explicit residual, or change of solution vector. The solution is then updated to the next time level using

$$U^{n+1}_{i,j} = U^n_{i,j} + \Delta U^n_{i,j}. \tag{3.21}$$

This is called an explicit method because the change of solution vector on the left-hand side of Eq. (3.20) can be obtained explicitly by evaluating the right-hand side at time level n. In other words, the solution at any point at the new time level $n+1$ is not dependent on the solution of other points at time level $n+1$.

While explicit time advancement methods can be useful for the solution of unsteady problems, where time accuracy is important, they are not effective for the simulation of most steady-state flows. This is because the numerical stability of the method imposes a maximum stable time step, which will in general be much smaller than the time required to reach a steady-state solution. This time step is based on the length of time it takes information to traverse a computational cell, and therefore will be dependent not only on the fluid dynamics, but also on the mesh spacing. For the one-dimensional Euler equations, the maximum stable time step can be shown to be

$$\Delta t_{max} = \frac{\Delta x}{|u| + a}.$$

Unfortunately, no proof of stability exists for the two- or three-dimensional case, although a similar definition is typically used. In practice, some fraction of Δt_{max} is typically chosen, and the actual time step is given by

$$\Delta t = CFL\,\Delta t_{max},$$

where CFL is the Courant-Fredereichs-Lewy number. Therefore, if it is desired to use an explicit method to reach a steady-state solution, a large number of iterations (time steps) will be required, especially on a highly resolved grid.

The use of an implicit method can allow much larger time steps to be taken. Therefore, a dramatic time-savings is possible in the solution of steady-state problems with the use of such implicit methods. In a true implicit method, the solution at any point in the grid is dependent on the solution of all other points at the new time level $n+1$. Therefore, in order to implement an implicit time advancement method it is necessary to evaluate the fluxes at time level $n+1$. The fully implicit upwind formulation of the inviscid problem can be expressed as

$$\begin{aligned}\delta U^n_{i,j} = -\frac{\Delta t}{V_{i,j}}(&F_{+i+1/2,j}S_{i+1/2,j} - F_{+i-1/2,j}S_{i-1/2,j} + \\ &F_{-i+1/2,j}S_{i+1/2,j} - F_{-i-1/2,j}S_{i-1/2,j} + \\ &G_{+i,j+1/2}S_{i,j+1/2} - G_{+i,j-1/2}S_{i,j-1/2} + \\ &G_{-i,j+1/2}S_{i,j+1/2} - G_{-i,j-1/2}S_{i,j-1/2})^{n+1} + \Delta t\, W^{n+1}_{i,j},\end{aligned} \tag{3.22}$$

where $\delta U^n_{i,j}$ is the change in the solution between time levels n and $n+1$. Now the fluxes and the source term vector are linearized in time and space

$$\begin{aligned}F^{n+1}_{i,j} &= F^n_{i,j} + \Big(\frac{\partial F}{\partial U}\Big)^n_{i,j}(U^{n+1}_{i,j} - U^n_{i,j}) + \mathcal{O}(\Delta t^2) \\ &\simeq F^n_{i,j} + A^n_{i,j}\delta U^n_{i,j} \\ W^{n+1}_{i,j} &\simeq W^n_{i,j} + C^n_{i,j}\delta U^n_{i,j},\end{aligned} \tag{3.23}$$

where $C^n_{i,j}$ is the source term Jacobian. This linearization is applied to the split fluxes in the same upwind biased manner described above, and the fully implicit

upwind formulation of the problem can be expressed by substituting the linearized fluxes into Eq. (3.22)

$$\begin{aligned}\delta U^n_{i,j} + \frac{\Delta t}{V_{i,j}}\Big\{ &\left(A_{+i+\frac{1}{2},j} S_{i+\frac{1}{2},j} \delta U_{i,j} - A_{+i-\frac{1}{2},j} S_{i-\frac{1}{2},j} \delta U_{i-1,j}\right) \\ &-\left(A_{-i-\frac{1}{2},j} S_{i-\frac{1}{2},j} \delta U_{i,j} - A_{-i+\frac{1}{2},j} S_{i+\frac{1}{2},j} \delta U_{i+1,j}\right) \\ &+\left(B_{+i,j+\frac{1}{2}} S_{i,j+\frac{1}{2}} \delta U_{i,j} - B_{+i,j-\frac{1}{2}} S_{i,j-\frac{1}{2}} \delta U_{i,j-1}\right) \\ &-\left(B_{-i,j-\frac{1}{2}} S_{i,j-\frac{1}{2}} \delta U_{i,j} - B_{-i,j+\frac{1}{2}} S_{i,j+\frac{1}{2}} \delta U_{i,j+1}\right)\Big\}^n \\ &+\Delta t\, C^n_{i,j} \delta U^n_{i,j} = \Delta U^n_{i,j},\end{aligned} \tag{3.24}$$

For the solution of viscous flows, Eq. (3.24) must be modified to include the contribution of the appropriate implicit viscous terms. Following the methods of Tysinger and Caughey (1991), or Gnoffo (1990), we can linearize the viscous flux vectors F_v and G_v, assuming that the transport coefficients are locally constant, to obtain

$$F_v^{n+1} \simeq F_v^n + \frac{\partial}{\partial \xi}(L\delta U)^n, \quad G_v^{n+1} \simeq G_v^n + \frac{\partial}{\partial \eta}(N\delta U)^n, \tag{3.25}$$

where the viscous Jacobians L and N are evaluated in such a way that they are functions of the vector of conserved quantities U, and not the derivatives of U. Note that in general F_v will be a function of U, U_ξ, and U_η, and thus the viscous Jacobians in Eq. (3.25) are only approximate. However, these expressions have been used effectively in a variety of algorithms. The viscous Jacobians are reproduced in the Appendix for perfect gas flows, but again the extension to chemically reacting flows is straightforward. With these definitions Eq. (3.24) will be unchanged if we simply replace the Euler Jacobians A and B with $\tilde{A}$ and $\tilde{B}$, where

$$\begin{aligned}\tilde{A}_+ &= A_+ - L, \qquad \tilde{A}_- = A_- + L, \\ \tilde{B}_+ &= B_+ - N, \qquad \tilde{B}_- = B_- + N.\end{aligned} \tag{3.26}$$

In principle, Eq. (3.24) can be solved directly for $\delta U^n_{i,j}$. However, with all of the off-diagonal terms on the left-hand side of the equation, the entire flowfield is fully coupled. Therefore, the entire left-hand side of the equation must be solved in

a fully implicit fashion, using a matrix inversion. A direct solution of this equation would thus require the formation and inversion of a large block banded matrix, which is too numerically intensive for practical problems, and would be extremely difficult to parallelize effectively. Most implicit methods, including those derived in this work, seek to make some simplifications to Eq. (3.24) which simplify the left-hand side and make the solution procedure more tractable. This is the focus of the following chapters.

Chapter 4

Data-Parallel Lower Upper Relaxation Method

4.1 Introduction

Equation (3.24) represents the standard implicit upwind formulation of the Navier-Stokes equations, and serves as the starting point for a variety of implicit schemes, including the new methods derived in this research. However, a direct solution of this equation would require a large block banded matrix inversion during each time step, which would be very memory and computationally expensive, and would be difficult to parallelize effectively. Therefore, most existing implicit methods make some simplifications to Eq. (3.24) which make the problem more tractable. The type of simplifications that will result in a successful numerical algorithm depends not only on the physics of the problems it is desired to solve, but also on the machine on which the solution is performed. Therefore, it is not unusual for an algorithm that gives good results on a serial or vector machine to be inefficient when implemented on a parallel computer.

However, the advent of parallel computers is still relatively recent, while serial and vector machines have been used for decades. Therefore there are many algorithms currently in use that have been designed to be efficient on a serial or vector architecture. Because of this, the traditional approach to programming on a parallel machine has been to choose an effective serial algorithm, and then implement it in parallel using domain decomposition. This approach can be effective in some cases [Simon (1992)]. However, care must be taken during the domain decomposition to ensure that each processor has an equal amount of computational work, or some processors will be idle during a portion of the solution process, leading to poor

parallel efficiency. This load balancing procedure can become complicated for some implicit methods. In addition, while the interior of each sub-domain can theoretically be computed in parallel, some type of inter-processor communication will be required to share data at the boundaries of each sub-domain. For some algorithms these boundary updates can require frequent and irregular communication, which again results in processor idle time and poor parallel efficiency. Finally, some of the most successful implicit methods rely on a Gauss-Seidel update pattern, in which a series of sweeps are made through the flowfield, using the latest available data during the update at each computational cell. This is extremely difficult to implement in parallel, since the solution at each grid point is dependent on the solution at neighboring points. This inherent data dependency means that the solution process must take place in a serial manner.

The mixed results obtained by using this domain decomposition approach make it clear that a method which is effective on a serial machine may not be a good choice on a parallel platform. Therefore, a better approach would be to develop a new algorithm which would be inherently parallel. This new algorithm should be designed so that it is amenable to implementation on a data-parallel machine without an explicit domain decomposition. Such an algorithm would then be readily portable to a wide variety of parallel architectures, since it is relatively easy to run a data-parallel code in a message-passing environment, while the reverse is generally not true.

From Eq. (3.24) we see that in principle it should be possible to develop a method of solution that is amenable to a data-parallel environment, at least for structured meshes, since the terms on the left hand side of Eq. (3.24) are evaluated only at point (i, j) and its four nearest neighbors. An analogous equation can be easily derived for three-dimensional flows; in that case point (i, j, k) will have six nearest neighbors. In order to achieve this goal, we first examine an effective serial algorithm, and look for modifications that can be made which allow the method to

be efficiently parallelized. This procedure is discussed in the next section.

4.2 Lower-Upper Symmetric Gauss-Seidel Method

The Lower-Upper Symmetric Gauss-Seidel (LU-SGS) method of Yoon and Jameson (1987,1988) is a logical choice for implicit time advancement, because it makes some simplifications to the implicit equation which diagonalize the problem and allow steady-state solutions to be obtained with a dramatically reduced number of iterations over an explicit method, without a substantial increase in the computational cost per iteration. In addition, the extension to three-dimensional flows is straightforward, unlike many implicit methods. This implicit operator has been well tested, and in fact is already used in a number of commercially available CFD codes, including INCA, OVERFLOW [Kandula and Buning (1994)], and TURNS [Srinivasan and Baeder (1993)]. Although the method as formulated does not lend itself to implementation on a parallel machine, we will see that it is possible to make some modifications to the algorithm that make it inherently data-parallel.

The central idea in the LU-SGS method is to make an approximation to the implicit Euler Jacobians which simplifies the resulting equation. Following the approach of Yoon and Jameson, we replace the true split Euler Jacobians with

$$A_+ = \frac{1}{2}(A + \rho_A I), \qquad A_- = \frac{1}{2}(A - \rho_A I), \tag{4.1}$$

where ρ_A is the spectral radius of the Jacobian A, and I is the identity matrix. The spectral radius is defined as the magnitude of the largest eigenvalue $|u'|+a$, where u' is the velocity component normal to the cell surface, and a is the local frozen sound speed. This approximation ensures that A_+ will be positive-definite, while A_- will be negative definite, which will increase the diagonal dominance of the equation set. Note that we make this change only to the implicit Jacobians, so the steady-state solution will not be affected. With this simplification differences between the positive and negative Jacobians become diagonal matrices. For example,

$$A_+ - A_- = \rho_A I. \tag{4.2}$$

If we neglect the viscous terms for now and move the off-diagonal terms in Eq. (3.24) to the right-hand side, the resulting implicit equation becomes

$$\begin{aligned}
&\left\{I + \lambda_A I + \lambda_B I + \Delta t\, C\right\}^n_{i,j} \delta U^n_{i,j} = \Delta U^n_{i,j} \\
&+\frac{\Delta t}{V_{i,j}} A^n_{+i-\frac{1}{2},j} S_{i-\frac{1}{2},j} \delta U^n_{i-1,j} - \frac{\Delta t}{V_{i,j}} A^n_{-i+\frac{1}{2},j} S_{i+\frac{1}{2},j} \delta U^n_{i+1,j} \\
&+\frac{\Delta t}{V_{i,j}} B^n_{+i,j-\frac{1}{2}} S_{i,j-\frac{1}{2}} \delta U^n_{i,j-1} - \frac{\Delta t}{V_{i,j}} B^n_{-i,j+\frac{1}{2}} S_{i,j+\frac{1}{2}} \delta U^n_{i,j+1},
\end{aligned} \tag{4.3}$$

where $\lambda_A = \frac{\Delta t S_I}{V} \rho_A$ and $\Delta U^n_{i,j}$ is the change of the solution due to the explicit flux evaluation at time level n. Note that we have assumed in Eq. (4.3) that adjacent cell face areas on the diagonal can be approximated by the appropriate cell-centered values, as in

$$S_{i+\frac{1}{2},j} \simeq S_{i-\frac{1}{2},j} \simeq S_I, \qquad S_{i,j+\frac{1}{2}} \simeq S_{i,j-\frac{1}{2}} \simeq S_J. \tag{4.4}$$

By moving the off-diagonal terms to the right-hand side of the equation we have in effect decoupled them from the diagonal. This means that their effect is included only after the matrix inversion on the left-hand side is performed. With this approximation the left hand side of Eq. (4.3) has been reduced to a diagonal matrix, with the exception of the source term Jacobian C. Therefore, if some way can be found to approximate the source term Jacobian by a diagonal matrix, it becomes trivial to solve for $\delta U^n_{i,j}$ via a scalar inversion.

There have been several efforts to develop an efficient way to diagonalize C, but most methods are either very complex or suffer from poor convergence. The obvious way to approximate C would be to replace it with a diagonal matrix containing the spectral radius, in a manner similar to that used for the Euler Jacobians. However, this is not used in practice, primarily because the eigenvalues of C are extremely difficult to determine analytically. Eberhardt and Imlay (1990) proposed to approximate C as

$$C \simeq -\text{diag}\left(\frac{1}{\tau_1}, \cdots, \frac{1}{\tau_s}, 0, 0, 0, \frac{1}{\tau_v}, 0\right), \tag{4.5}$$

where τ_s is a characteristic chemistry time scale for species s and τ_v is a time scale for vibrational relaxation. This method was combined with the use of reaction rate limiters to achieve good convergence rates for many reacting flows. However, for flows with stiff chemistry this method suffers from slow convergence due to the fact that each chemical species has a different chemical time scale, and is thus allowed to evolve at a different rate. For example, for the five-species air model used in this work, it can be shown that since N_2 dissociation has a much higher activation energy than O_2 dissociation, we have

$$\tau_{N_2} \gg \tau_{O_2}.$$

This disparity means that the diagonal element of C will be much smaller for N_2 than O_2, and therefore N_2 will adjust much more quickly to convection induced changes. For highly reactive flowfields this can lead to a violation of elemental conservation [Hassan *et al.* (1993)].

In order to alleviate this problem Hassan *et al.* proposed a relatively simple modification to the Eberhardt model that improves the stability and convergence of the method. The first step involves a change of the U-vector to ensure elemental conservation at each time step. The modified U becomes

$$U = (\tilde{\rho}_N,\ \tilde{\rho}_O,\ \rho_{NO},\ \rho_N,\ \rho_O,\ \rho u,\ \rho v,\ E_v,\ E)', \tag{4.6}$$

where $\tilde{\rho}_N$ and $\tilde{\rho}_O$ are the elemental densities of nitrogen and oxygen. For the five species air model, the elemental densities are given by

$$\begin{aligned} \frac{\tilde{\rho}_N}{M_N} &= 2\frac{\rho_{N_2}}{M_{N_2}} + \frac{\rho_{NO}}{M_{NO}} + \frac{\rho_N}{M_N}, \\ \frac{\tilde{\rho}_O}{M_O} &= 2\frac{\rho_{O_2}}{M_{O_2}} + \frac{\rho_{NO}}{M_{NO}} + \frac{\rho_O}{M_O}. \end{aligned} \tag{4.7}$$

Now, since elemental conservation must hold during chemical reactions, the source terms for the elemental conservation equations are zero, and the approximation for C becomes

$$C \simeq -\mathrm{diag}\left(0,\ 0,\ \frac{1}{\tau_{NO}},\ \frac{1}{\tau_N},\ \frac{1}{\tau_O},\ 0,\ 0,\ \frac{1}{\tau_v},\ 0\right). \tag{4.8}$$

The chemical time scale parameters are evaluated from density derivatives of the source terms and take the functional form

$$\frac{1}{\tau_s} = \sqrt{\sum_{r=1}^{n_{el}} \left(\frac{\partial w_s}{\partial \tilde{\rho}_r}\right)^2 + \sum_{r=n_{el}+1}^{n_s} \left(\frac{\partial w_s}{\partial \rho_r}\right)^2}, \tag{4.9}$$

where n_{el} is the number of elemental conservation equations in the modified system. The vibrational relaxation time scale is given simply by

$$\frac{1}{\tau_v} = \left|\frac{\partial w_v}{\partial E_v}\right|. \tag{4.10}$$

This method is slightly more complex than the standard solution procedure, due primarily to the more complicated pressure and energy derivatives which result from the change of variables. However, the convergence rate for thermochemical nonequilibrium flowfields is significantly improved, and in fact can become comparable to that for perfect gas simulations. In addition, the change of variables is made only in the implicit portion of the solution, which reduces the complexity of the code considerably. The major drawback of this approach is that it is not a universal solution. The required modification of the U-vector is chosen to ensure elemental conservation and to minimize the disparity between chemical time scales. Therefore, the modification is dependent on the chemistry model used. While this method works well for the air flows discussed here, a similar approach may not be effective for combustion simulations, where there are many more chemical species, and the chemical time scales can vary widely.

For the solution of viscous flows a similar simplification must be made to the viscous Jacobians L and N, which were given in Eq. (3.25). Unlike the source term Jacobian, the spectral radii of the viscous Jacobians are readily evaluated, and are given by [Tysinger and Caughey (1991)]

$$\rho_L = \rho_N = \frac{\kappa}{\rho c_v}\left(\frac{S}{J}\right), \tag{4.11}$$

where J is the determinant of the transformation matrix between cartesian and curvilinear coordinates. This approach has been used successfully by several authors

[Candler and Olynick (1992) & Hassan *et al.* (1993)] in conjunction with the LU-SGS method.

The resulting implicit equation is then

$$\begin{aligned}\Big\{I+\lambda_A I + 2\lambda_L I + \lambda_B I + 2\lambda_N I + \Delta t\,\mathrm{diag}(C)\Big\}^n_{i,j}\delta U^n_{i,j} &= \Delta U^n_{i,j} \\ +\frac{\Delta t}{V_{i,j}}\tilde{A}^n_{+i-\frac{1}{2},j}S_{i-\frac{1}{2},j}\delta U^n_{i-1,j} - \frac{\Delta t}{V_{i,j}}\tilde{A}^n_{-i+\frac{1}{2},j}S_{i+\frac{1}{2},j}\delta U^n_{i+1,j} \\ +\frac{\Delta t}{V_{i,j}}\tilde{B}^n_{+i,j-\frac{1}{2}}S_{i,j-\frac{1}{2}}\delta U^n_{i,j-1} - \frac{\Delta t}{V_{i,j}}\tilde{B}^n_{-i,j+\frac{1}{2}}S_{i,j+\frac{1}{2}}\delta U^n_{i,j+1},\end{aligned} \tag{4.12}$$

where $\lambda_L = \frac{\Delta t S_I}{V}\rho_L$, and $\tilde{A}$ and $\tilde{B}$ are defined by Eq. (3.26). Note that if the computational grid is constructed so that η is the body-normal direction, the ξ-direction viscous Jacobians often will be much smaller than those in the η-direction, and can be eliminated from Eq. (4.12) with no effect on the convergence of the algorithm. This is equivalent to making a thin layer approximation on the implicit side only, and will not affect the steady-state solution.

The LU-SGS algorithm employs a pair of corner-to-corner sweeps through the flowfield using the latest available data for the off-diagonal terms to solve Eq. (4.12). The first sweep begins in the lower left corner of the computational grid and proceeds to the upper right corner. During the first sweep data in front of the current location are neglected, and the latest available data are used behind. This results in an intermediate solution vector δU^*, as in

$$\begin{aligned}\delta U^*_{i,j} = &\Big(I + \lambda^n_A I + 2\lambda^n_L I + \lambda^n_B I + 2\lambda^n_N I + \Delta t\,\mathrm{diag}(C^n)\Big)^{-1}_{i,j} \\ &\times\Big(\Delta U^n_{i,j} + \frac{\Delta t}{V_{i,j}}\tilde{A}^n_{+i-\frac{1}{2},j}S_{i-\frac{1}{2},j}\delta U^*_{i-1,j} + \frac{\Delta t}{V_{i,j}}\tilde{B}^n_{+i,j-\frac{1}{2}}S_{i,j-\frac{1}{2}}\delta U^*_{i,j-1}\Big).\end{aligned} \tag{4.13a}$$

The final solution is then obtained by sweeping in the reverse direction, neglecting data behind and using latest available data in front of the current location, as in

$$\begin{aligned}\delta U^n_{i,j} = \delta U^*_{i,j} - &\Big(I + \lambda^n_A I + 2\lambda^n_L I + \lambda^n_B I + 2\lambda^n_N I + \Delta t\,\mathrm{diag}(C^n)\Big)^{-1}_{i,j} \\ &\times\frac{\Delta t}{V_{i,j}}\Big(\tilde{A}^n_{-i+\frac{1}{2},j}S_{i+\frac{1}{2},j}\delta U^n_{i+1,j} + \tilde{B}^n_{-i,j+\frac{1}{2}}S_{i,j+\frac{1}{2}}\delta U^n_{i,j+1}\Big).\end{aligned} \tag{4.13b}$$

This method has been shown to be efficient on a serial machine, and can be vectorized with long vector lengths by ordering the data along the grid diagonal. This approach has several other advantages which make it attractive for the computation of large problems. First, since the term on the left-hand side which operates on $\delta U_{i,j}$ is a diagonal matrix, the system can be solved without the costly matrix inversions required by more exact methods. In addition, the approximate Jacobians have a much simpler form than the true split matrices, and may easily be computed on the fly as needed instead of stored. This means that the memory storage requirements will scale linearly with the number of equations, as opposed to the quadratic scaling which is observed for more exact methods. Therefore larger problems can be solved with the same available memory.

Another advantage of the LU-SGS method is that it is inherently stable. In fact, it can be shown that the method is unconditionally stable for linear problems. This inherent stability makes the method insensitive to small changes in implementation, and allows several further approximations which significantly reduce the amount of CPU time and memory required with only a small decrease in convergence rate. First, it is possible to use stored face-centered values for surface normals and cosines on the diagonal, which eliminates the need to either store cell-centered values or recompute them at every time step. Also, the implicit boundary conditions may be treated simply by setting the boundary δU's to zero before beginning the implicit portion of the algorithm. This greatly reduces the time spent on boundary updates, which can be significant in a parallel algorithm.

However, using Gauss-Seidel sweeps to solve Eq. (4.12) is inefficient on a parallel computer, because not all of the available processors can be kept busy during each sweep. For example, if we examine the forward sweep in Eq. (4.13a), each (i, j) point cannot begin the computation until points $(i–1, j)$ and $(i, j–1)$ have completed the step. This data-dependence means that during the sweeping process only a diagonal band of cells can be active at any one time, while the rest are idle. This process

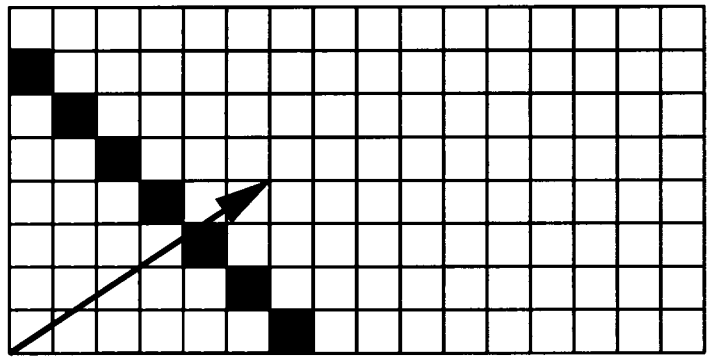


Fig. 4.1 Schematic diagram of the Gauss-Seidel sweeping process used by the LU-SGS method. Only those cells within the shaded area have enough information to perform the solution update at a given time.

is illustrated schematically in Fig. (4.1). As the forward sweep proceeds, only the band of computational cells represented by the shaded area have all the information they need to perform the update, therefore any processor not containing one (or more) of these elements must remain idle.

For example, on a three-dimensional problem having a grid size of n^3 implemented on a machine having n^2 processors, less than one half of the processors can be kept active on average during the sweeps [Candler *et al.* (1994)]. Some improvement can be gained on coarse-grained parallel machines by laying out the data among the processors along the grid diagonals [Barszcz *et al.* (1993)], but only at the cost of frequent and irregular boundary updates. The challenge of efficiently implementing the LU-SGS sweeps in parallel is well illustrated by examining performance results obtained by computer vendors on the Lower-Upper (LU) NAS 1 Parallel Benchmark pseudo application [Bailey *et al.* (1993)]. This application has the same data-dependencies as the LU-SGS method, and parallel efficiencies are typically one half of that obtained on the other two pseudo applications (BT and SP) which are Alternating Direction Implicit (ADI) based schemes [Bailey *et al.* (1994)].

A method to overcome this inefficiency has been proposed by Wong *et al.* (1995). In this approach, domain decomposition is used to break the prob-

lem into a number of sub-domains (one per processor), and each sub-domain is allowed to perform its own local LU-SGS sweeps in parallel. The boundaries between the processors are then updated explicitly at the end of each iteration using inter-processor communication. While this method does show promise on coarse-grained parallel machines, Wong shows that the convergence rate of the method is dependent on the number of processors used, and deteriorates rapidly as the number of sub-domains becomes large. This is simply due to the fact that the ratio of boundary cells to interior cells increases with the number of processors. In the limit as the number of processors approaches the number of grid points, all of the cells become boundary cells, and this approach fails completely. From this we see that, while the LU-SGS method has several advantages that make it attractive, significant modifications are required to reduce or eliminate the data dependencies that are inherent in Eq. (4.13) and make the method amenable to implementation in a massively parallel environment.

4.3 Data-Parallel Lower-Upper Relaxation Method

Ideally the solution at time level $n+1$ should not depend on the solution of neighboring points at the new time level. This would allow each grid point to perform its update independently, and would thus ensure that all of the processors are active at all times *on any parallel computer* during the solution of Eq. (4.12). In this way it should be possible to achieve a significant fraction of the peak theoretical floating point performance of the machine. This can be accomplished for the LU-SGS method outlined above by replacing the Gauss-Seidel sweeps with a series of pointwise relaxation steps using the following algorithm. First, the implicit off-diagonal terms are neglected and the explicit residual, $\Delta U^n_{i,j}$, is divided by the diagonal operator to obtain $\delta U^{(0)}$

$$\delta U^{(0)}_{i,j} = \left\{ I + \lambda^n_A I + 2\lambda^n_L I + \lambda^n_B I + 2\lambda^n_N I + \Delta t \, \mathrm{diag}(C)^n \right\}^{-1}_{i,j} \Delta U^n_{i,j}.$$

Then a series of k_{max} relaxation steps are made using

$$\begin{aligned}
&\text{for } k = 1,\ k_{max} \\
&\delta U_{i,j}^{(k)} = \Big\{ I + \lambda_B^n I + 2\lambda_L^n I + \lambda_B^n I + 2\lambda_N^n I + \Delta t\, \mathrm{diag}(C)^n \Big\}_{i,j}^{-1} \Big\{ \Delta U_{i,j}^n + \frac{\Delta t}{V_{i,j}} \Big(\\
&\tilde{A}^n_{+i-\frac{1}{2},j} S_{i-\frac{1}{2},j} \delta U_{i-1,j}^{(k-1)} - \tilde{A}^n_{-i+\frac{1}{2},j} S_{i+\frac{1}{2},j} \delta U_{i+1,j}^{(k-1)} + \\
&\tilde{B}^n_{+i,j-\frac{1}{2}} S_{i,j-\frac{1}{2}} \delta U_{i,j-1}^{(k-1)} - \tilde{B}^n_{-i,j+\frac{1}{2}} S_{i,j+\frac{1}{2}} \delta U_{i,j+1}^{(k-1)} \Big) \Big\} \\
&\text{then} \\
&\delta U_{i,j}^n = \delta U_{i,j}^{(k_{max})}.
\end{aligned} \tag{4.14}$$

With this approach, all data required for each relaxation step have already been computed during the previous $(k-1)$ step. Therefore, the entire relaxation step can be performed simultaneously in parallel without any data dependencies, and all communication can be handled by optimized nearest-neighbor routines. In addition, since the same pointwise calculation is performed on each computational cell, load balancing will be ensured as long as the data are evenly distributed across the processors. The resulting diagonal Data-Parallel Lower-Upper Relaxation (DP-LUR) algorithm should be very efficient on a parallel supercomputer, and, aside from the required nearest-neighbor communication, the method should approach the peak operational performance of the machine. In addition, since the method is entirely matrix free, it is memory efficient, which permits the solution of very large problems. Finally, the extension of the DP-LUR method to three-dimensional flows is straightforward. For the 3-D case there are simply two additional spectral radii on the diagonal, and two additional nearest-neighbor off-diagonal terms on the right-hand side.

4.4 Performance of the DP-LUR Method

The diagonal DP-LUR method has been tested on two- and three-dimensional geometries, with an emphasis on evaluating the convergence properties and parallel performance of the new method. The primary test case shown here is the Mach 15 flow over a cylinder-wedge blunt body at 69 km altitude, with Reynolds numbers

based on freestream conditions and the body length varying from 3×10^4 to 3×10^7. A sample 128×128 grid for this problem is shown in Fig. (4.2), and temperature contours for a typical flow solution are shown in Fig. (4.3). The three-dimensional computations presented in this section are performed on multiple planes of the same 2-D grids, which makes it easy to directly compare the convergence properties of the two- and three-dimensional implementations of the method. However, the method has been used with good results on several true three-dimensional flows [Levin *et al.* (1996) & Bibb *et al.* (1997)].

The boundary layer resolution is measured with the wall variable

$$y_+ = \frac{\rho y u_*}{\mu},$$

where u_* is the friction velocity, given in terms of the wall stress, τ_w, by $u_* = \sqrt{\tau_w/\rho}$. For a well resolved boundary layer the mesh spacing is typically chosen so that $y_+ \leq 1$ for the first cell above the body surface. The grid for each case is then exponentially stretched from the wall to the outer boundary, which results in approximately 70 points in the boundary layer at all Re for the baseline 128×128 grids.

Unless noted otherwise, the results presented here are based on modified first order Steger-Warming flux vector splitting for the numerical fluxes [MacCormack and Candler (1989)]. However, it is important to note that the derivation of the implicit algorithm is general, and it can therefore be used with many flux evaluation methods. In order to test this, results have also been obtained using a Harten-Yee upwind non-MUSCL TVD scheme [Yee (1989)]. All implicit cases were run with an essentially infinite time step ($CFL \approx 10^6$), unless noted otherwise. However, due to the approximate nature of the implicit operator, the CFL number does not represent the true elapsed time of each iteration. Thus, no improvement in convergence is obtained by taking still larger time steps.

The implicit off-diagonal viscous Jacobians in Eq. (4.14) facilitate the diffusion of viscous effects through the flowfield, and therefore proper treatment of these

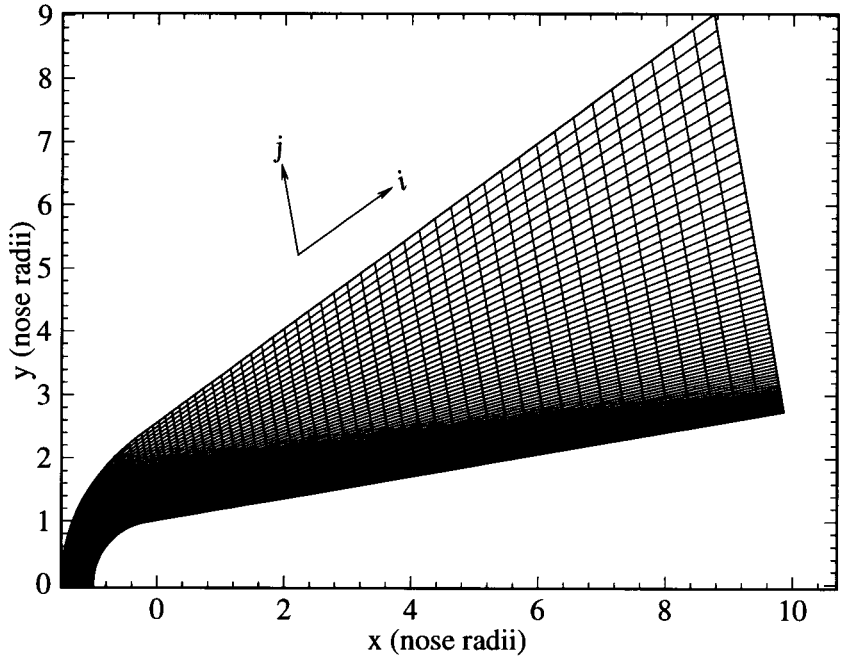


Fig. 4.2 Sample 128 × 128 cylinder-wedge grid. Every fourth grid point shown.

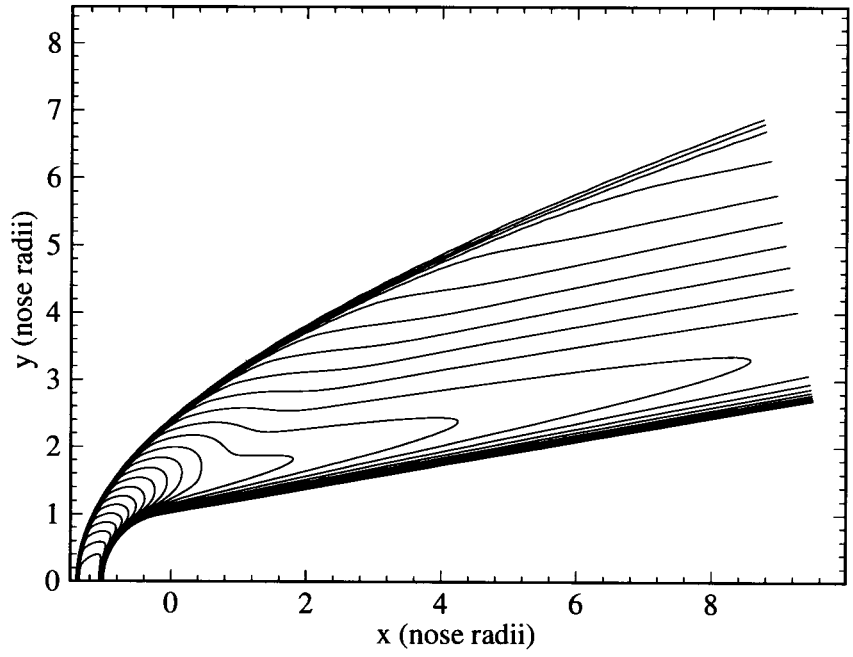


Fig. 4.3 Temperature contours for a typical flow solution. 2-D cylinder-wedge blunt body in perfect air at $M_\infty = 15$ and $Re = 3 \times 10^4$. 128 × 128 grid with $y_+ = 1$ for the first cell above the body.

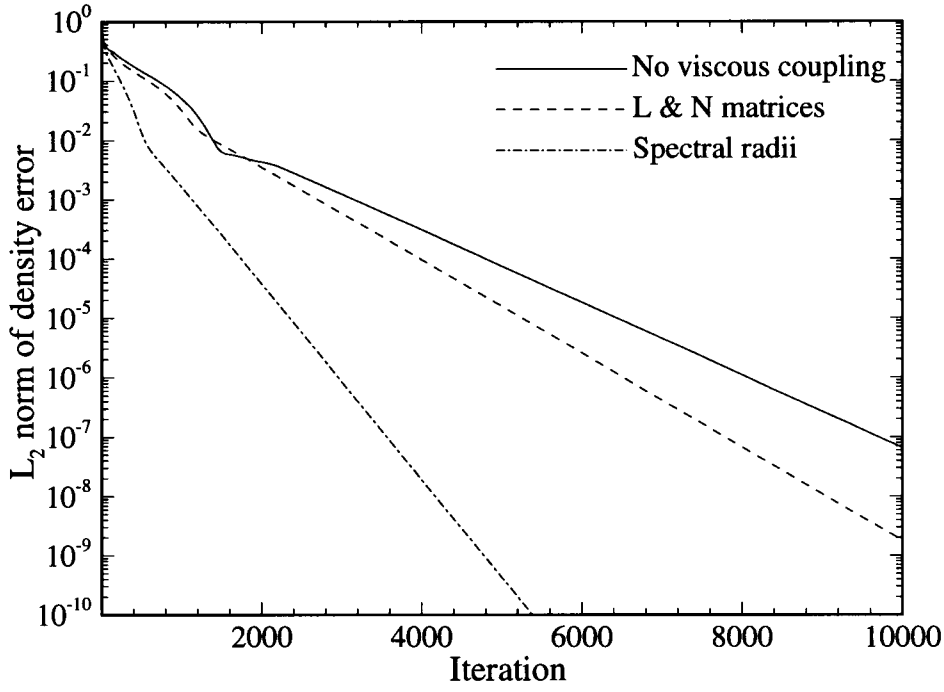


Fig. 4.4 Convergence histories for the diagonal DP-LUR method showing the effect of the off-diagonal viscous coupling terms. 2-D cylinder-wedge blunt body in perfect air at $M_\infty = 15$ and $Re = 3 \times 10^4$. 128×128 grid with $y_+ = 1$.

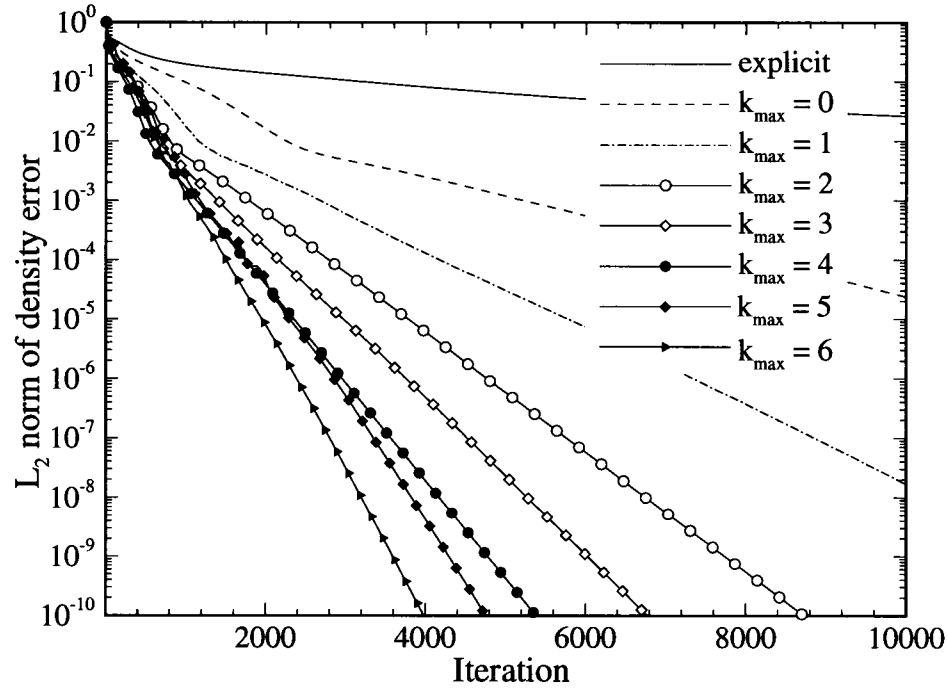
terms can be very important to the convergence and stability of the algorithm. There are two possible ways to handle these off-diagonal terms; the full L and N matrices can be used as shown above, or all viscous Jacobians can be approximated by their spectral radii, as in

$$L \simeq \rho_L I, \qquad N \simeq \rho_N I.$$

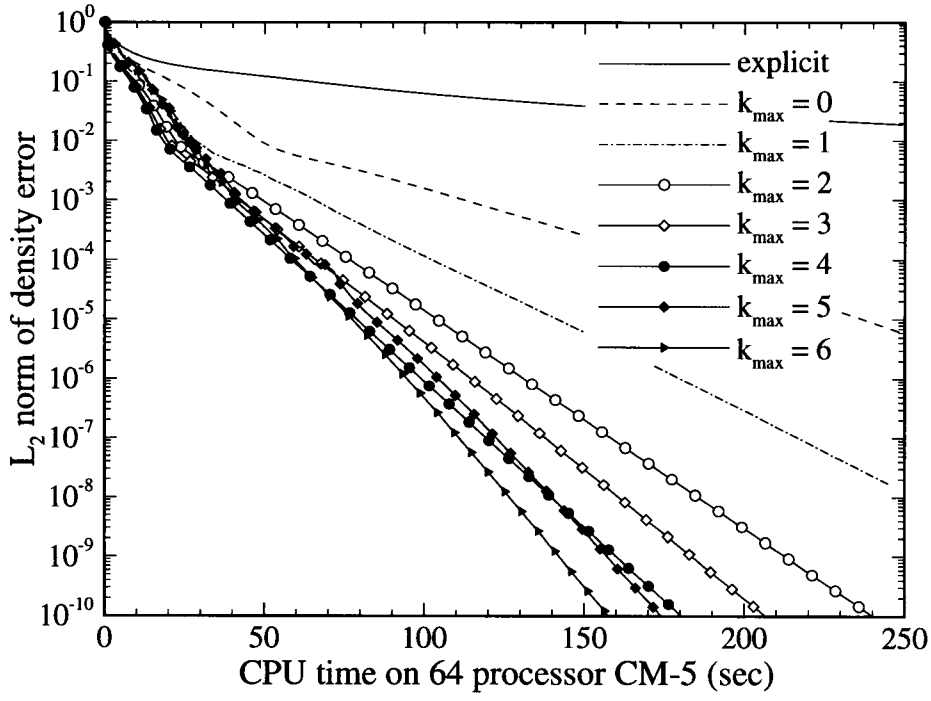
Figure (4.4) plots the convergence histories for these two approaches for a $Re = 3 \times 10^4$ flow. As expected, the coupling terms greatly improve the convergence rate of the algorithm, and in fact are necessary for stability at some flow conditions. For all cases run to date the spectral radius approximation is the most efficient approach; therefore it will be used for all of the diagonal DP-LUR results presented below. This result is not surprising, since it is consistent with the level of approximation made on the diagonal.

The effect of the number of relaxation steps (k_{max}) on convergence is shown in Fig. (4.5a) for the 2-D cylinder-wedge at $Re = 3 \times 10^4$ and $y_+ = 1$. The explicit solution was obtained using a first-order Euler method with the maximum stable timestep ($CFL = 0.1$). We see that even $k_{max} = 0$ is a significant improvement over the explicit method. However, the performance of the DP-LUR method improves when the off-diagonal terms are included ($k_{max} > 0$), and further increases with increasing k_{max}. Figure (4.5b) plots the convergence histories for the same case versus computer time on a 64 processor CM-5. Because each relaxation step has a small cost compared to the evaluation of the fluxes, we see that increasing k_{max} also improves the cost effectiveness of the method. However, it should be noted that this improvement diminishes rapidly for large values of k_{max}. For the case shown in Fig. (4.5b), the improvement in cost effectiveness between $k_{max} = 2$ and $k_{max} = 4$ is about 25%, while the improvement from $k_{max} = 4$ to $k_{max} = 6$ is about 12%, and the improvement from $k_{max} = 6$ to $k_{max} = 8$ (not shown) is only 6%. It is theoretically possible to use any number of relaxation steps; however our experience has shown that using values of k_{max} greater than 4 can cause the algorithm to become less stable for more complex problems, which reduces the maximum allowed CFL number and slows the overall convergence rate. Therefore, all of the results presented here are run at $k_{max} = 4$, which has proven to give the best combination of numerical stability and cost effectiveness for a variety of problems.

The performance of the DP-LUR method is compared to the LU-SGS method in Fig. (4.6a). We see that the LU-SGS method, which consists of a single forward and backward sweep from corner to corner through the flowfield, performs better than DP-LUR with $k_{max} = 2$. However, with $k_{max} = 4$ the DP-LUR method performs better than LU-SGS. This result is surprising because the Gauss-Seidel sweeps of the LU-SGS method should allow information to travel across the entire grid during each implicit time step, while with the DP-LUR method data can only

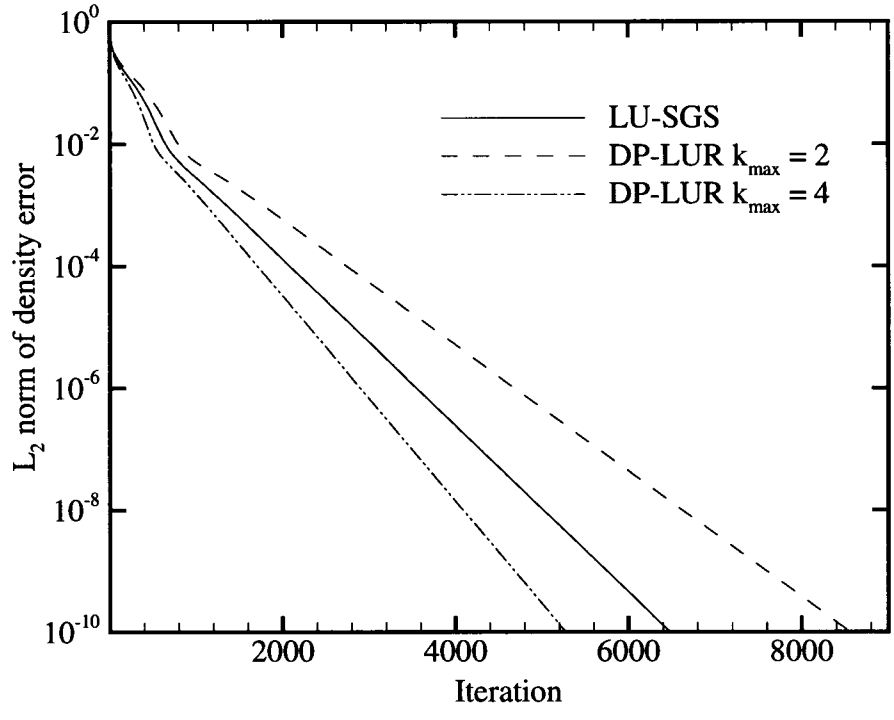


a)

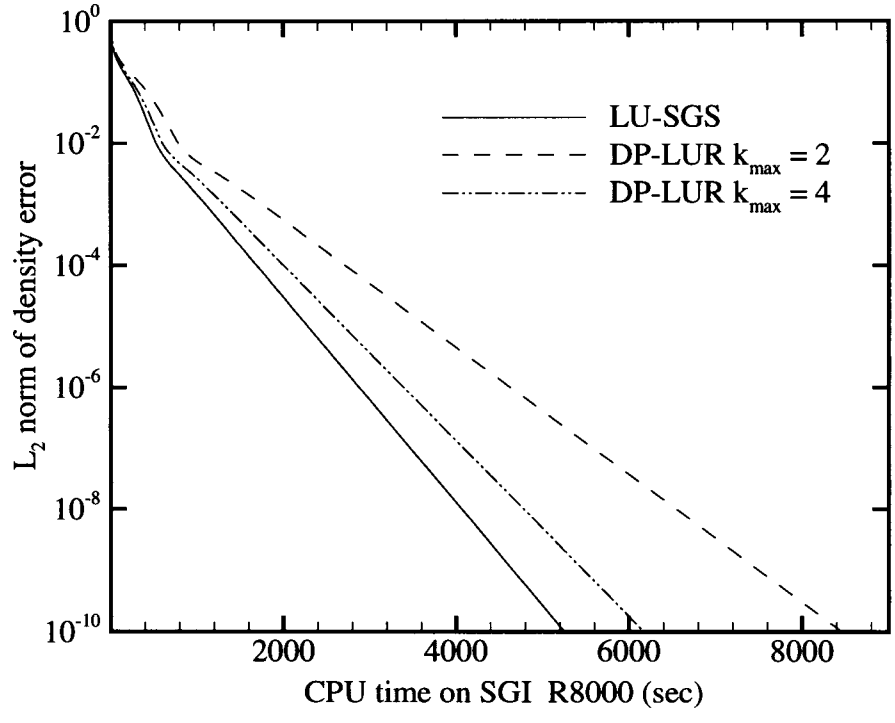


b)

Fig. 4.5 a) Convergence histories and b) CPU times on a 64 processor CM-5, for the diagonal DP-LUR method showing influence of k_{max}. 2-D cylinder-wedge blunt body in perfect air at $M_\infty = 15$ and $Re = 3 \times 10^4$. 128×128 grid with $y_+ = 1$.



a)



b)

Fig. 4.6 a) Convergence histories and b) CPU times on a single processor SGI R8000 machine for the DP-LUR and original LU-SGS methods. 2-D cylinder-wedge blunt body in perfect air at $M_\infty = 15$ and $Re = 3 \times 10^4$. 128×128 grid with $y_+ = 1$.

travel k_{max} grid points per time step.

A direct comparison of the CPU time required by each method on a parallel machine is meaningless, because the LU-SGS method is inefficient when implemented in parallel, as discussed earlier. However, some idea of the relative effectiveness can be obtained by a comparison of the amount of computational work required by each of the methods. In Table (4.1) we examine the number of floating point operations required to perform a single implicit iteration for each method, as compared to the floating point operations required for the Euler and viscous flux evaluation. In this table, arithmetic is counted as a single floating point operation, while functions such at square root and exponentiation are counted as eight.

Function	Floats
Fluxes	1151
LU-SGS	275
DP-LUR $k_{max} = 2$	396
DP-LUR $k_{max} = 4$	726

Table 4.1 Number of floating point operations required for each portion of the 2-D perfect gas implementation of the LU-SGS and DP-LUR methods.

From Table (4.1) we see that the implicit portion of the code is in all cases smaller than the evaluation of the fluxes, which means that small increases in the amount of time spent in the implicit routine will not have a large effect on the total solution time. The DP-LUR method with $k_{max} = 2$ does require slightly more floating point operations than LU-SGS. This is because the contribution of all four off-diagonal terms is computed during each DP-LUR relaxation step, while only two off-diagonal terms are used during each LU-SGS sweep. However, it is apparent that the DP-LUR method with $k_{max} = 2$ should require a total amount of work comparable to that of the LU-SGS method, while the DP-LUR method with $k_{max} = 4$ will require about twice as much work.

It is also interesting to compare the CPU time required for the DP-LUR and LU-SGS methods on a single processor machine. This is shown in Figure (4.6b) for

the two algorithms implemented on a Silicon Graphics (SGI) R8000 workstation. We see that the LU-SGS method is the most cost effective approach on a single processor machine, converging in about 15% less time than DP-LUR with $k_{max} = 4$. However, this small performance penalty is more than made up for on a parallel computer, where the data-dependencies and irregular communication patterns of the LU-SGS method would make it extremely slow. This is a graphic example of the need to design algorithms specifically for a parallel computer.

Since the single processor performance of the LU-SGS method is slightly better than that of the DP-LUR method, Wissink *et al.* (1996) has shown that an interesting possibility arises on a coarse-grained parallel computer. By breaking the problem into sub-domains we could perform local LU-SGS sweeps in the interior of each sub-domain, and then use DP-LUR relaxation steps to update the boundaries between the domains. The convergence properties of this hybrid algorithm should then approach that of the DP-LUR method as the number of sub-domains increases, and approach that of the LU-SGS method as the number of sub-domains tends toward one. This hybrid approach has been developed by Wissink *et al.*, and has been shown to be effective for the solution of transonic rotor flows.

Figure (4.7) shows the effect of the method used to evaluate the numerical fluxes on the convergence rate of the diagonal DP-LUR method. Results are presented for the modified Steger-Warming approach and a Harten-Yee upwind TVD scheme. We see that for this case both methods result in good convergence properties. However, the TVD method is more sensitive to the size of the implicit time step, with a maximum stable CFL number of approximately 100. Because of this, the TVD method requires more iterations to reach a steady-state solution. This time step limitation is not surprising, since the DP-LUR method has only first-order spatial accuracy on the implicit side. This has no effect on the quality of the converged solution, but it can cause a slight degradation of the convergence rate when used with a high-order accurate flux evaluation, as seen here.

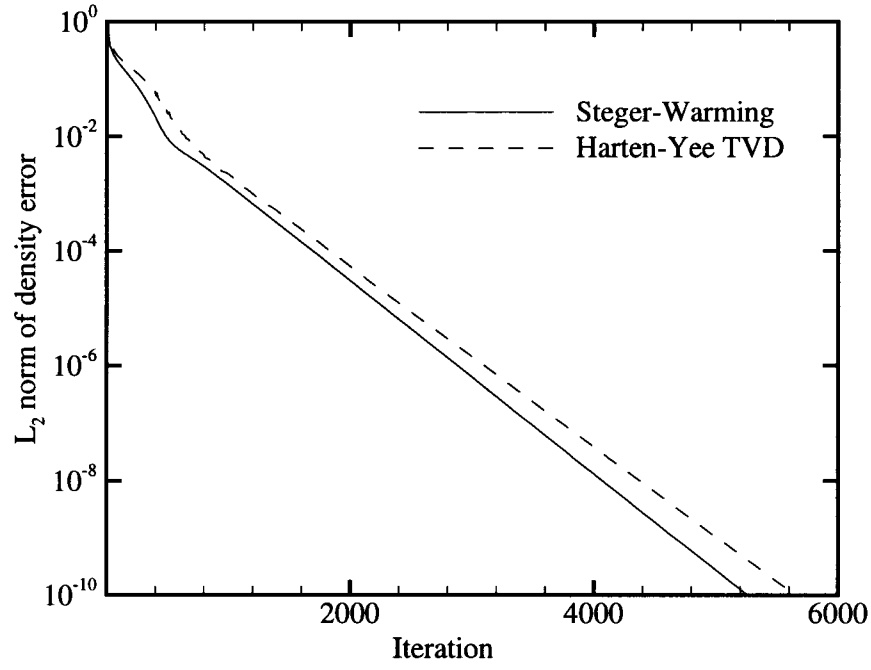


Fig. 4.7 Convergence histories for the diagonal DP-LUR method, showing the effect of the choice of the flux evaluation method. 2-D cylinder-wedge blunt body in perfect air at $M_\infty = 15$ and $Re = 3 \times 10^4$. 128 × 128 grid with $y_+ = 1$.

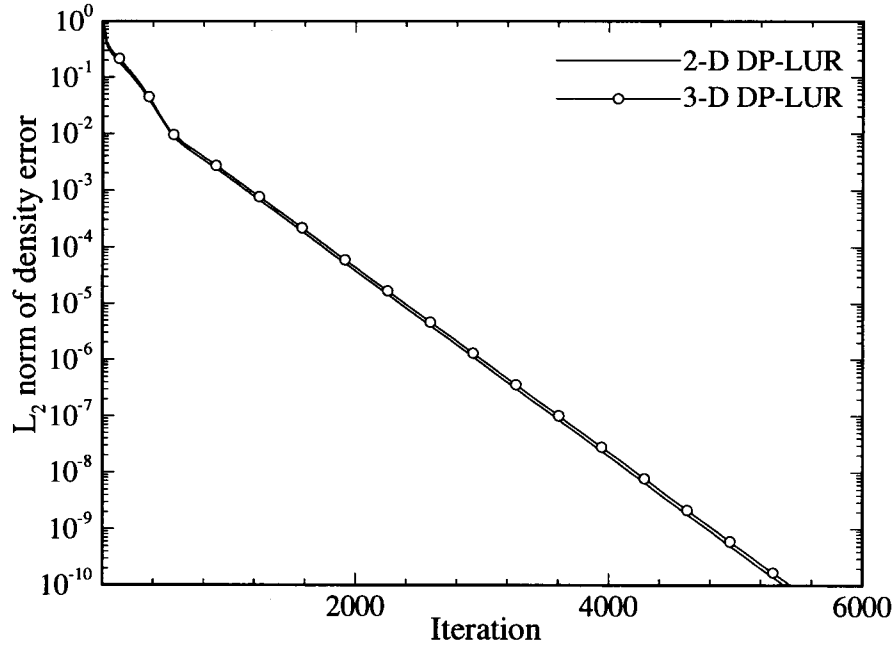


Fig. 4.8 Convergence histories for the two- and three-dimensional versions of the diagonal DP-LUR method. Cylinder-wedge blunt body in perfect air at $M_\infty = 15$ and $Re = 3 \times 10^4$. 128 × 128 grid with $y_+ = 1$.

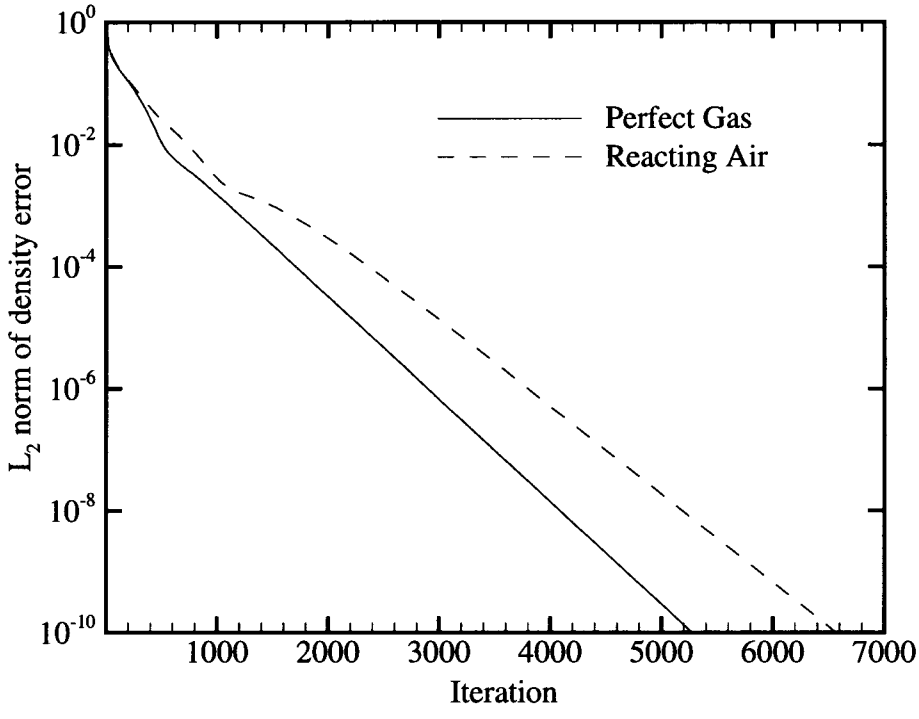


Fig. 4.9 Convergence histories for the diagonal DP-LUR method showing the effect of extension to reacting air problems. 2-D cylinder-wedge blunt body at $M_\infty = 15$, $Re = 3 \times 10^4$, and 69 km altitude. 128×128 grid with $y_+ = 1$.

Figure (4.8) plots the convergence histories of the two- and three-dimensional diagonal DP-LUR methods for the same case shown in Fig. (4.7). We see that there is essentially no difference in the convergence rate of the two implementations. This result holds for all cases tested in this work.

The DP-LUR method was next tested with the five species thermochemical nonequilibrium air model discussed in Chapter 3. All of the comparisons between different k_{max} values and between the DP-LUR and LU-SGS methods presented earlier are valid for the reacting flow solutions as well, and therefore are not repeated here. Figure (4.9) presents a comparison of convergence histories for a reacting air and perfect gas simulation on the same computational grid. We see that there is a slight degradation of the convergence rate of the reacting air case as compared to the perfect gas simulation. For this case the reacting air code requires about 25% more iterations to achieve a ten order of magnitude reduction in the density

error norm. Perfect gas and reacting air convergence rates were also compared for a variety of other flow conditions, with the general result that the amount of convergence degradation of the reacting air code was directly related to the stiffness of the equation set. This is not a surprising result, since we are approximating the source term Jacobian with a diagonal matrix. This approach should work well for flows dominated by fluid dynamic effects, since the flux Jacobians would then be much larger than the source term Jacobian. However, for flows with stiff source terms we would expect the convergence rate to be degraded as the diagonalized source term Jacobian begins to dominate the equation set. This problem with the diagonal DP-LUR method will be revisited in the next chapter.

Although this method is efficient for the solution of a variety of flowfields, it is well known [Candler *et al.* (1992) & Yoon and Kwak (1994)] that diagonalized methods of this type show significant degradation of the convergence rate on the high cell aspect ratio grids necessary to resolve the boundary layer of a high Reynolds number flow. This degradation is partially due to an over-stabilizing effect that is inherent in the Yoon and Jameson approximation to the Jacobian matrices. To illustrate this point, Fig. (4.10) shows the convergence histories of the algorithm for several perfect gas viscous flows with Reynolds numbers from 3×10^4 to 3×10^7. The 128×128 grid for each case was chosen so that $y_+ = 1$ for the first cell above the body, ensuring that the boundary layer is equally well resolved for all cases. Since the thickness of the boundary layer (δ) decreases with increasing Re [Anderson (1989)], as in

$$\frac{\delta}{x} \propto \frac{1}{\sqrt{Re}},$$

the maximum cell aspect ratio (CAR) of the grid must increase as well to meet the $y_+ = 1$ requirement. For a two-dimensional grid the cell aspect ratio is defined by the maximum ratio of the length to the width of a computational cell, as in

$$\mathrm{CAR} = \max\left(\frac{\Delta\xi}{\Delta\eta}, \frac{\Delta\eta}{\Delta\xi}\right).$$

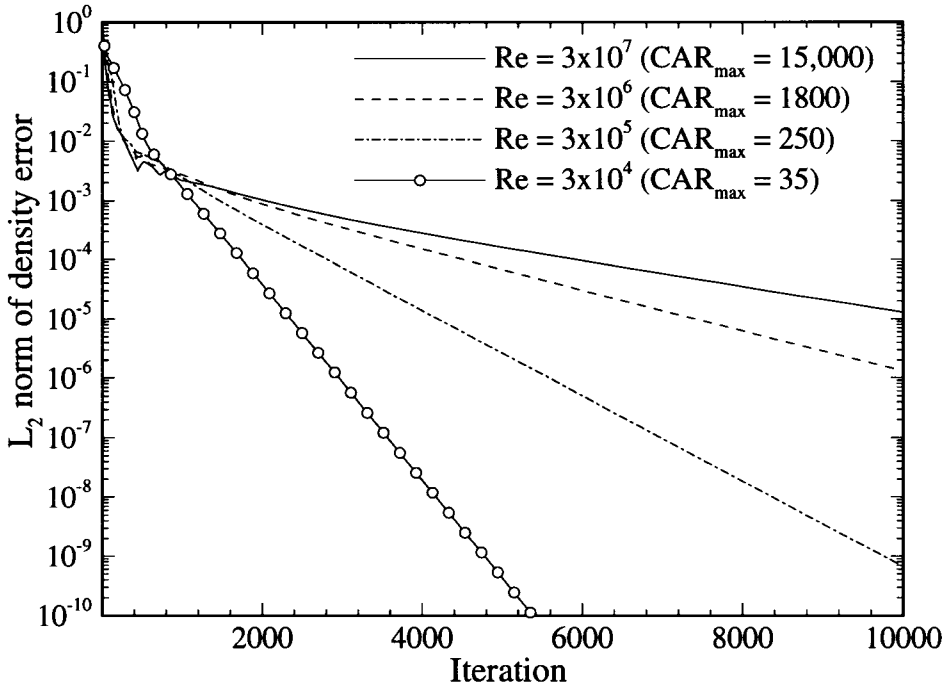


Fig. 4.10 Convergence histories for the diagonal DP-LUR method showing influence of Reynolds number. 2-D cylinder-wedge blunt body in perfect air at $M_\infty = 15$. 128×128 grid with $y_+ = 1$ for each case.

A similar definition is made for three-dimensional flows. For the test cases in Fig. (4.10), the maximum CAR ranges from 35 for $Re = 3 \times 10^4$ to about 15,000 for $Re = 3 \times 10^7$. From the figure we see that while the algorithm converges well for low Re (low CAR) flows, the convergence rate decreases rapidly as Re (and therefore CAR) increases.

The underlying reason for this convergence degradation can be seen easily by examining the diagonal term in Eq. (4.14). If we assume a rectangular grid for simplicity, with x tangential and y normal to the body, it can be seen that the scalar terms on the diagonal are inversely proportional to the mesh spacing, as in

$$\lambda_A = \frac{\Delta t\, S_I}{V}\rho_A \simeq \frac{\Delta t}{\Delta x}\rho_A.$$

Therefore, neglecting the source term Jacobian, the diagonal term in Eq. (4.14) becomes

$$\Big\{I + \lambda_A I + 2\lambda_L I + \lambda_B I + 2\lambda_N I\Big\} \simeq \Big\{I + \frac{\Delta t}{\Delta x}(\rho_A + 2\rho_L)I + \frac{\Delta t}{\Delta y}(\rho_B + 2\rho_N)I\Big\}.$$

From this it is apparent that as the cell aspect ratio in the boundary layer increases ($\Delta x \gg \Delta y$), the magnitude of the diagonal operator will be increasingly dominated by the terms inversely proportional to Δy. This will eventually cause the solution to become over-stabilized in the x direction, slowing the convergence rate. This result also applies to 3-D flows, except that there are now three relevant cell lengths, one normal and two tangential to the body. In all cases the amount of over-stabilization, and therefore the degradation of the convergence rate, is driven by the maximum cell aspect ratio. For most external flows the maximum CAR will be given approximately by the ratio of the mesh spacing in one of the body-tangential directions by the spacing in the body-normal direction for the first cell above the body surface. In addition, examination of flowfield solutions indicates that for the high CAR cases the cells nearest to the body evolve slowly compared to the rest of the flow. This means that even after the global error norm has fallen several orders of magnitude the solution may still be changing slowly near the body surface. This makes it difficult to determine when steady-state has been reached, and can lead to the use of incorrect flow solutions.

This problem is solely a function of the computational grid, and consequently will be observed for inviscid flows on high cell aspect ratio grids as well. This is demonstrated in Fig. (4.11), which compares the convergence of the viscous and inviscid algorithms for two of the cases in Fig. (4.10). In each case the inviscid calculation was performed on the same computational grid as the corresponding viscous case. As expected, the performance of the algorithm on high CAR grids is nearly independent of viscosity. However, there is a slight degradation in the convergence rate for the viscous problem, which becomes more pronounced as the Reynolds number decreases. This effect, which can also be seen in Fig. (4.11), is caused by two factors. First, the diagonalization made to the viscous Jacobian matrices is only an approximation, and thus we would expect to see the convergence degrade as the Reynolds number decreases and the viscous Jacobians become larger

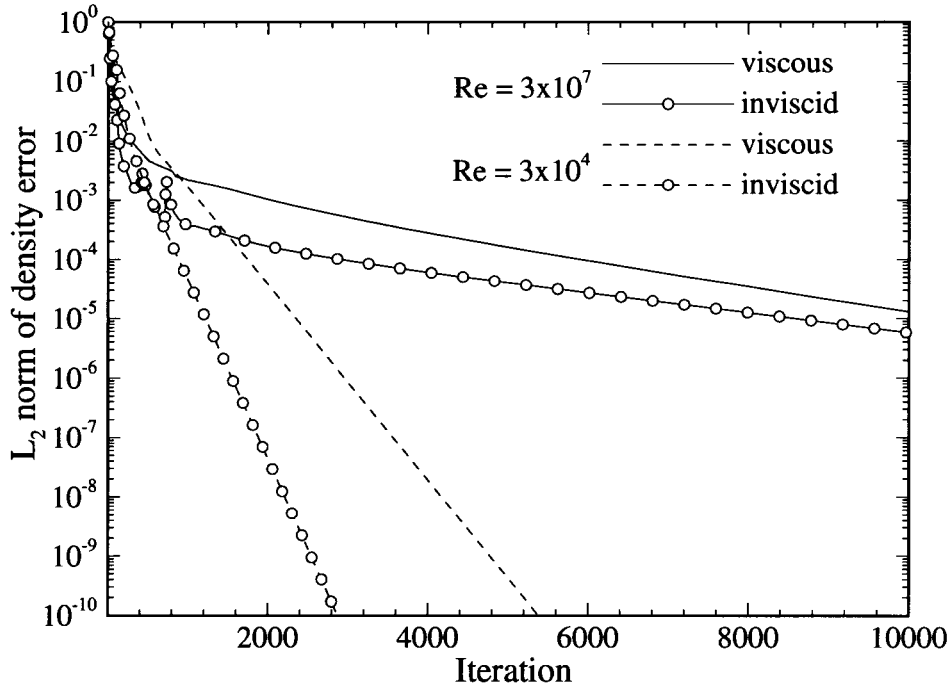


Fig. 4.11 Convergence histories for the viscous and inviscid implementations of the diagonal DP-LUR method. 2-D cylinder-wedge blunt body in perfect air at $M_\infty = 15$. 128×128 grid with $y_+ = 1$ for each viscous case.

as compared to the Euler Jacobians. Second, since the implicit off-diagonal terms are on the right-hand side of Eq. (4.14), their effect is only weakly coupled to the diagonal. This could cause a degradation of the convergence rate in the boundary layer of a viscous flow, where there are strong gradients in the body-normal direction. These factors will be explored in the following chapters.

There have been many methods proposed to resolve this high CAR convergence problem, including local time-stepping, modified Jacobian splitting [Obayashi and Guruswamy (1994)], the introduction of non-isotropic diagonal terms [Swanson and Turkel (1985) & Vatsa and Wedan (1990)], multigrid methods [Yoon and Kwak (1994) & Edwards (1994)], and using the diagonal method only as a preconditioner for a more exact implicit scheme [Obayashi (1988)]. However, none of these modifications fully address the fundamental problem with the diagonalized operator, and none deal with the difficulties of solving flows with stiff source terms. In order to

solve these problems it is necessary to return to the original implicit equation (3.24) and review the simplifications made, and how they affect the convergence properties of the method. This procedure, and the new implicit algorithm that results from this analysis, will be discussed in the next chapter.

Chapter 5

Full Matrix DP-LUR Method

5.1 Introduction

While the diagonal DP-LUR method developed in Chapter 4 has shown promise for the solution of many flows, we have seen that the formulation of the method makes it ineffective in two cases: the simulation of high Reynolds number flows, and the simulation of flows with complex chemistry. In order to address these problems it is necessary to return to the original implicit equation (3.24) and review the simplifications made in Chapter 4. Only by understanding the effect of these simplifications on the convergence properties of the method can a new method be developed to overcome these difficulties. The two primary simplifications made in Chapter 4 were to move all of the implicit off-diagonal terms in Eq. (3.24) to the right-hand side of the equation, and to approximate the Jacobian matrices by using their spectral radii. The first simplification will reduce the implicit coupling of the method, while the second has been shown to cause slow convergence on the high cell aspect ratio (CAR) grids required for the simulation of high Reynolds number flows. The obvious way to avoid poor performance on high aspect ratio grids would be to eliminate these simplifications and move the off-diagonal terms back to the left-hand side of Eq. (3.24). However, solution of the resulting equation would require the inversion of a large block banded matrix at every time step, which would be both computationally and memory intensive, and would destroy the data-parallel nature of the algorithm.

Another possibility arises if we recognize that the approximation given by Yoon and Jameson (1988) acts to ensure the numerical stability of the algorithm by in-

creasing the diagonal dominance of the system. This approximation has been shown to work well on low CAR grids. However, as the CAR increases the diagonal dominance of the system increases as well, eventually resulting in an over-stabilized algorithm that converges very slowly. A better approach for simulations on high cell aspect ratio grids would be to remove the Yoon and Jameson approximation and use the exact flux Jacobians on and off the diagonal for the Euler and viscous terms [Wright *et al.* (1996)]. If we also replace the diagonalized source term Jacobian with its exact counterpart, we also address the convergence degradation that is observed for reacting flows with stiff source terms, and eliminate the need to make a mechanism dependent modification to the U-vector. Therefore, the resulting algorithm should be applicable to all reacting flows, even those with complex chemical mechanisms. This is the central idea in the formulation of the full matrix DP-LUR method, discussed in the next section.

5.2 Full Matrix DP-LUR Method

The derivation of the exact upwind Euler Jacobians has been presented in Chapter 3, and will not be repeated here. The resulting Jacobians are given by

$$A_{\pm} = E_A^{-1}\Lambda_{A\pm}E_A, \qquad B_{\pm} = E_B^{-1}\Lambda_{B\pm}E_B, \tag{5.1}$$

where E_A^{-1} and E_A are the left and right eigenvector matrices, given in terms of the notation of Chapter 3 by $E_A^{-1} = S^{-1}C_A^{-1}$ and $E_A = C_A S$. $\Lambda_{A\pm}$ is the diagonal matrix of positive or negative eigenvalues. For the solution of the Navier-Stokes equations, the viscous Jacobians L and N are also required. These matrices have also been presented in Chapter 3. The Jacobian matrices can be substituted directly into Eq. (3.24), and the implicit off-diagonal terms moved to the right-hand side as

before to obtain our starting equation

$$\begin{aligned}
&\Big\{I + \frac{\Delta t}{V_{i,j}}\big[\tilde{A}_{+i+\frac{1}{2},j}S_{i+\frac{1}{2},j} - \tilde{A}_{-i-\frac{1}{2},j}S_{i-\frac{1}{2},j} \\
&+ \tilde{B}_{+i,j+\frac{1}{2}}S_{i,j+\frac{1}{2}} - \tilde{B}_{-i,j-\frac{1}{2}}S_{i,j-\frac{1}{2}}\big] + \Delta t\, C_{i,j}\Big\}^n \delta U^n_{i,j} = \Delta U^n_{i,j} \\
&+ \frac{\Delta t}{V_{i,j}}\tilde{A}^n_{+i-\frac{1}{2},j}S_{i-\frac{1}{2},j}\delta U^n_{i-1,j} - \frac{\Delta t}{V_{i,j}}\tilde{A}^n_{-i+\frac{1}{2},j}S_{i+\frac{1}{2},j}\delta U^n_{i+1,j} \\
&+ \frac{\Delta t}{V_{i,j}}\tilde{B}^n_{+i,j-\frac{1}{2}}S_{i,j-\frac{1}{2}}\delta U^n_{i,j-1} - \frac{\Delta t}{V_{i,j}}\tilde{B}^n_{-i,j+\frac{1}{2}}S_{i,j+\frac{1}{2}}\delta U^n_{i,j+1}.
\end{aligned} \tag{5.2}$$

The full matrix form of the DP-LUR method then follows logically if we use the same approach to solve Eq. (5.2) as shown in Eq. (4.14). The resulting algorithm would still require a large amount of communication or storage to implement in data-parallel, since all of the Jacobians should be evaluated at the indicated cell face. However, the method can be further modified by calculating all flux Jacobians at the upwind cell center with almost no degradation of the convergence rate. This means that the Jacobians can be computed with purely local data. In addition, if we assume that adjacent cell face areas that appear on the diagonal are approximately equal

$$S_{i+\frac{1}{2},j} \simeq S_{i-\frac{1}{2},j} \simeq S_I, \qquad S_{i,j+\frac{1}{2}} \simeq S_{i,j-\frac{1}{2}} \simeq S_J, \tag{5.3}$$

and recognize that

$$\Lambda_+ - \Lambda_- = |\Lambda|, \tag{5.4}$$

we can combine terms on the diagonal, and the method simplifies further. The resulting full matrix DP-LUR algorithm is then given by

$$\delta U^{(0)}_{i,j} = \Big\{I + \frac{\Delta t}{V_{i,j}}\Big(E_A^{-1}|\Lambda_A|E_A S_I - 2LS_I + E_B^{-1}|\Lambda_B|E_B S_J - 2NS_J\Big)^n - \Delta t\, C^n\Big\}^{-1}_{i,j}\Delta U^n_{i,j},$$

and the series of k_{max} relaxation steps

for $k = 1,\ k_{max}$

$$\begin{aligned}\delta U_{i,j}^{(k)} =& \Big\{ I + \frac{\Delta t}{V_{i,j}} \Big(E_A^{-1} |\Lambda_A| E_A S_I - 2LS_I + E_B^{-1} |\Lambda_B| E_B S_J - 2NS_J \Big)^n \\ & - \Delta t\, C^n \Big\}^{-1}_{i,j} \Big\{ \Delta U_{i,j}^n \\ & + \frac{\Delta t}{V_{i,j}} \tilde{A}^n_{+i-1,j} S_{i-\frac{1}{2},j} \delta U_{i-1,j}^{(k-1)} - \frac{\Delta t}{V_{i,j}} \tilde{A}^n_{-i+1,j} S_{i+\frac{1}{2},j} \delta U_{i+1,j}^{(k-1)} \\ & + \frac{\Delta t}{V_{i,j}} \tilde{B}^n_{+i,j-1} S_{i,j-\frac{1}{2}} \delta U_{i,j-1}^{(k-1)} - \frac{\Delta t}{V_{i,j}} \tilde{B}^n_{-i,j+1} S_{i,j+\frac{1}{2}} \delta U_{i,j+1}^{(k-1)} \Big\}\end{aligned} \tag{5.5}$$

then

$$\delta U_{i,j}^n = \delta U_{i,j}^{(k_{max})},$$

where S_I and S_J are evaluated using local face-centered metrics. Implicit boundary conditions are implemented simply by setting the boundary δU's to zero before beginning the implicit portion of the algorithm, as in the diagonal DP-LUR method. Although Eq. (5.5) is written for 2-D problems, the extension to three dimensions remains straightforward. The full matrix DP-LUR method is more computationally intensive than the diagonal version, since the true split Jacobian matrices have a more complex form than the approximate Jacobians used in the diagonal method. In addition, the term operating on $\delta U_{i,j}$ is now a $n_{eq} \times n_{eq}$ matrix which must be stored and inverted at every grid point, where n_{eq} is the number of conservation equations in the system. This also increases the memory requirements over the diagonal method, and implies that the memory required will now scale with the square of the number of equations in the system. However, the current implementation was designed to be memory efficient, and therefore the off-diagonal Jacobians required during the k_{max} relaxation steps are computed on the fly and discarded during each relaxation step. This eliminates the need to store four additional Jacobian matrices (six for 3-D flows) at each grid point, at the expense of additional computation. For the 2-D perfect gas implementation of the algorithms, the full matrix method requires 3152 floating point operations for the implicit part of the problem, as compared to only 726 for the diagonal method. However, the additional

computation is entirely local, and thus the full matrix DP-LUR method requires no more communication per time step than the diagonal version. In addition, with the simplifications discussed above, the algorithm remains inherently data-parallel and free of all data dependencies. Since the implicit portion of the full matrix method requires about four times as many floating point operations as the diagonal method during each time step, but has the same amount of inter-processor communication, we expect that the full matrix method will have a higher parallel efficiency. This is in fact the case, as we will see in Chapter 7, where the parallel performance of the new algorithms are discussed in detail.

One potential problem with this method is that by removing the Yoon and Jameson approximation we are no longer assured of diagonal dominance for all situations, and therefore would no longer expect the algorithm to be unconditionally stable in the linear limit. This is possibly why no equivalent serial Gauss-Seidel version of the algorithm has been presented in the literature. However, although we have not performed a formal stability analysis of the method, experience has shown that it is numerically robust for both high and low CAR grids, as will be seen in the next section.

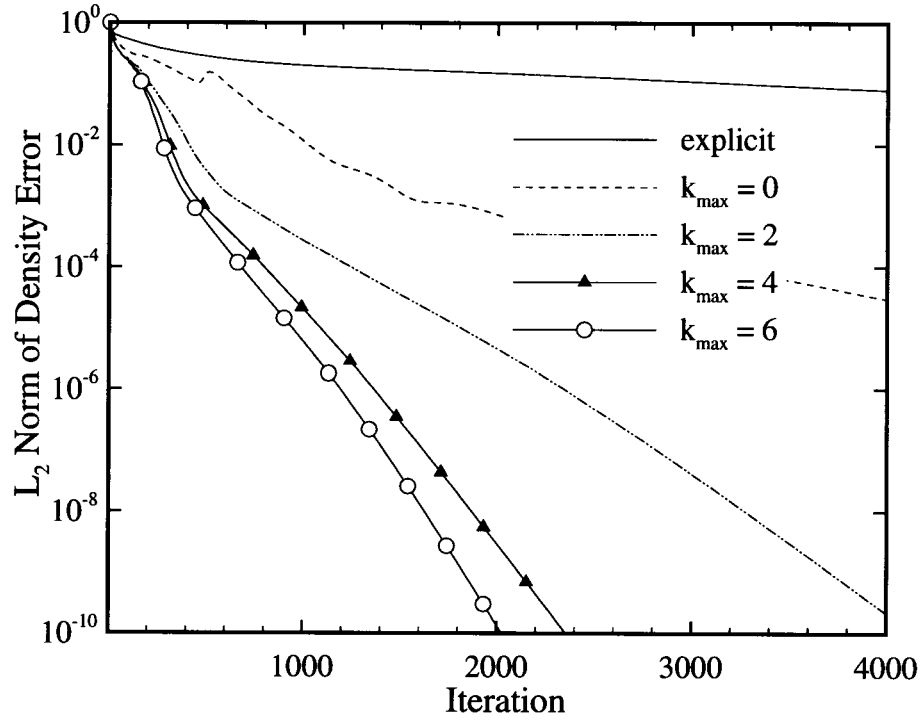
5.3 Performance of the Full Matrix Method

The full matrix DP-LUR method has been tested on two- and three-dimensional geometries, with an emphasis on evaluating the convergence properties and parallel performance of the new method. The primary test case is the same as that used in Chapter 4: the Mach 15 flow over a cylinder-wedge blunt body at 69 km altitude, with Reynolds numbers based on freestream conditions and the body length varying from 3×10^3 to 3×10^7. Unless noted otherwise, the results presented here are based on modified first-order Steger-Warming flux vector splitting for the numerical fluxes [MacCormack and Candler (1989)]. However, the derivation of the full matrix DP-LUR method remains general, and it can therefore be used with many flux evaluation methods. Although the discussion in this section concentrates

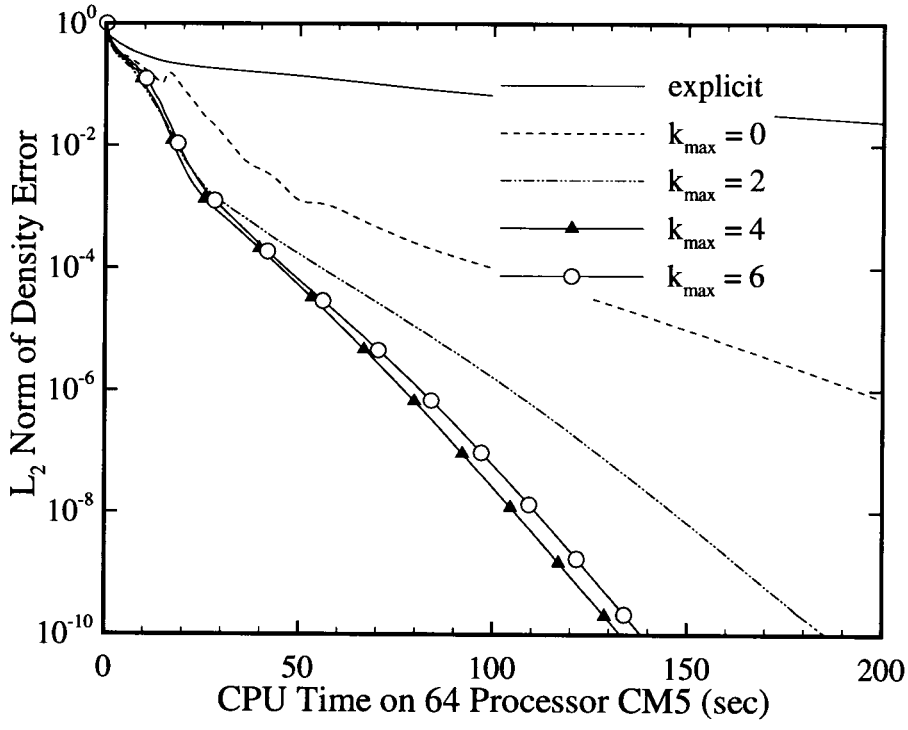
on the two-dimensional implementation, the extension to three-dimensional flows is straightforward, and the full matrix method has been used with success on several three-dimensional flows [Olejniczak *et al.* (1996b) & Johnson *et al.* (1997)].

As expected, the full matrix DP-LUR method is not unconditionally stable, and thus it is not possible to use essentially infinite time steps such as those used for the diagonal method. However, because Eq. (5.5) is a more physical representation of the problem than Eq. (4.14), we see that the numerical stability and convergence rate of the method are sensitive to the magnitude of the implicit time step, rather than the CFL number. In fact, for the cases discussed here, the maximum stable time step is governed solely by the physics of the flowfield, independent of the mesh spacing (and therefore independent of the CFL number.)

The effect of the number of relaxation steps (k_{max}) on the convergence rate of the full matrix method is shown in Fig. (5.1a) for the 2-D cylinder-wedge at $Re = 3 \times 10^4$. The computational grid is identical to that used in Fig. (4.5). In the figure $k_{max} = 0$ corresponds to performing just the initial pointwise matrix inversion and back substitution at each $i-j$ point in the domain, with none of the implicit off-diagonal terms included. The full matrix method with $k_{max} = 0$ is similar to the method developed by Gnoffo (1990) in the Langley Aerothermodynamic Upwind Relaxation Algorithm (LAURA). Although this approach does offer a significant improvement over the first-order Euler explicit method, we see from Fig. (5.1a) that the convergence rate of the method improves when the effect of the off-diagonal coupling terms is included ($k_{max} > 0$). The full matrix DP-LUR method exhibits convergence characteristics similar to the diagonal method, with the convergence rate steadily improving as k_{max} increases. Figure (5.1b) plots the convergence histories for the same case versus computer time on a 64 processor CM-5. We see that the computation of each relaxation step is an efficient process, and thus increasing k_{max} also increases the cost effectiveness of the method, up to $k_{max} = 4$. Therefore, all full matrix results shown here are run at $k_{max} = 4$, which is also the



a)



b)

Fig. 5.1 a) Convergence histories and b) CPU times on a 64 processor CM-5, for the full matrix DP-LUR method showing influence of k_{max}. 2-D cylinder-wedge blunt body in perfect air at $M_\infty = 15$ and $Re = 3 \times 10^4$. 128×128 grid with $y_+ = 1$.

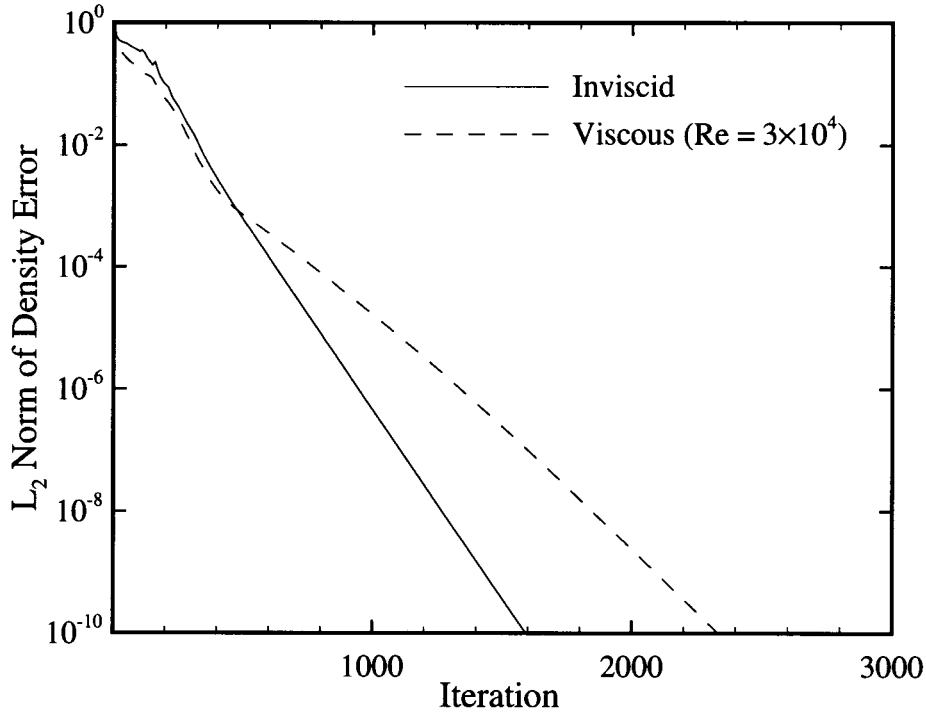


Fig. 5.2 Convergence histories for the viscous and inviscid implementations of the full matrix DP-LUR method. 2-D cylinder-wedge blunt body in perfect air at $M_\infty = 15$. Both cases run on the same 128 × 128 grid.

most stable implementation of the method.

Figure (5.2) compares the convergence histories of the viscous and inviscid implementations of the full matrix DP-LUR method. Both cases were run on the same 128 × 128 computational grid, which was chosen to give a $y_+ = 1$ resolved boundary layer for the viscous flow. We see that there is a significant degradation of the convergence rate for the viscous problem, which requires about 45% more iterations to reach ten orders of magnitude reduction in the density error norm. Both implementations converge at nearly the same rate during the first 400 iterations, which represents the non-linear motion of the bow shock through the grid. However, after the bow shock has reached its final location the convergence rate for the viscous solution decreases, while the inviscid solution converges at the same rate until ten orders of density error norm reduction are reached. This effect becomes more pronounced as the Reynolds number decreases. This trend, which was also

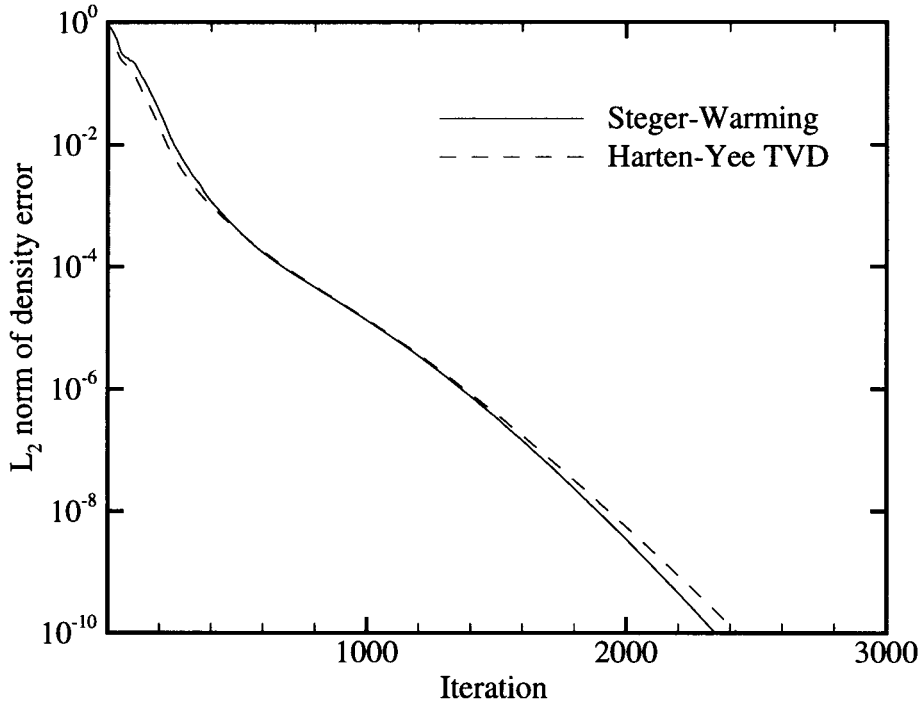


Fig. 5.3 Convergence histories for the full matrix DP-LUR method, showing the effect of the choice of the flux evaluation method. 2-D cylinder-wedge blunt body in perfect air at $M_\infty = 15$, and $Re = 3 \times 10^4$. 128×128 grid with $y_+ = 1$.

observed for the diagonal DP-LUR method, is primarily due to the fact that the implicit off-diagonal terms are on the right-hand side of Eq. (5.5), which means that their effect is only weakly coupled to the diagonal, and in fact is lagged by one relaxation step. This will slow the convergence of the viscous solution, due to the strong body-normal gradients within the boundary layer. This will be explored further in the next chapter.

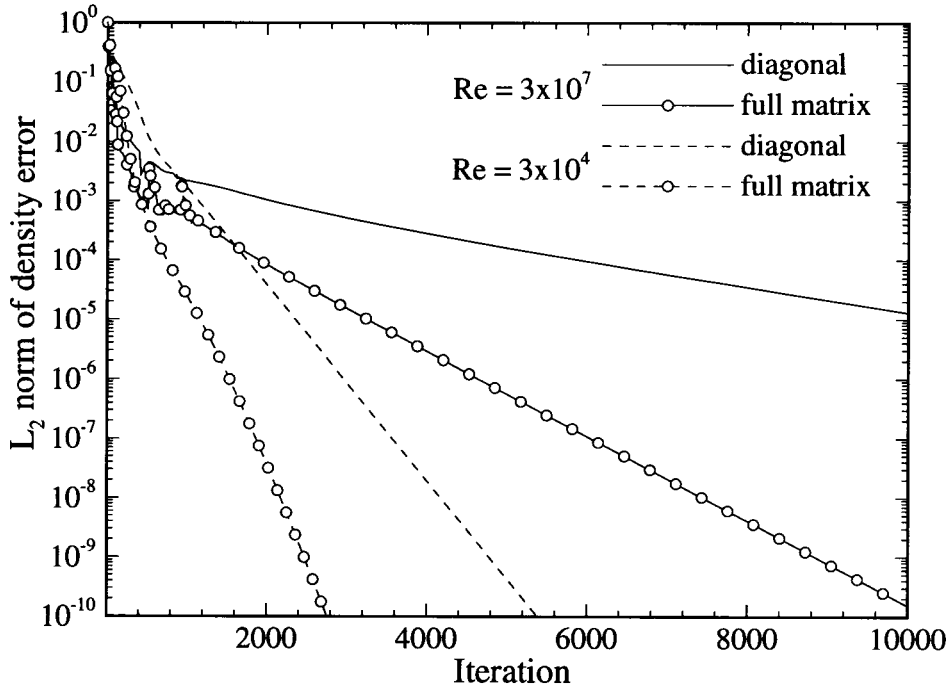
Figure (5.3) shows the effect of the method used to evaluate the numerical fluxes on the convergence rate of the full matrix DP-LUR method. Results are presented for the modified Steger-Warming approach and a Harten-Yee upwind TVD scheme. We see that both methods have good convergence properties for this test case, although the TVD method does show a slight degradation of the convergence rate.

The performance of the full matrix DP-LUR method is compared to the diag-

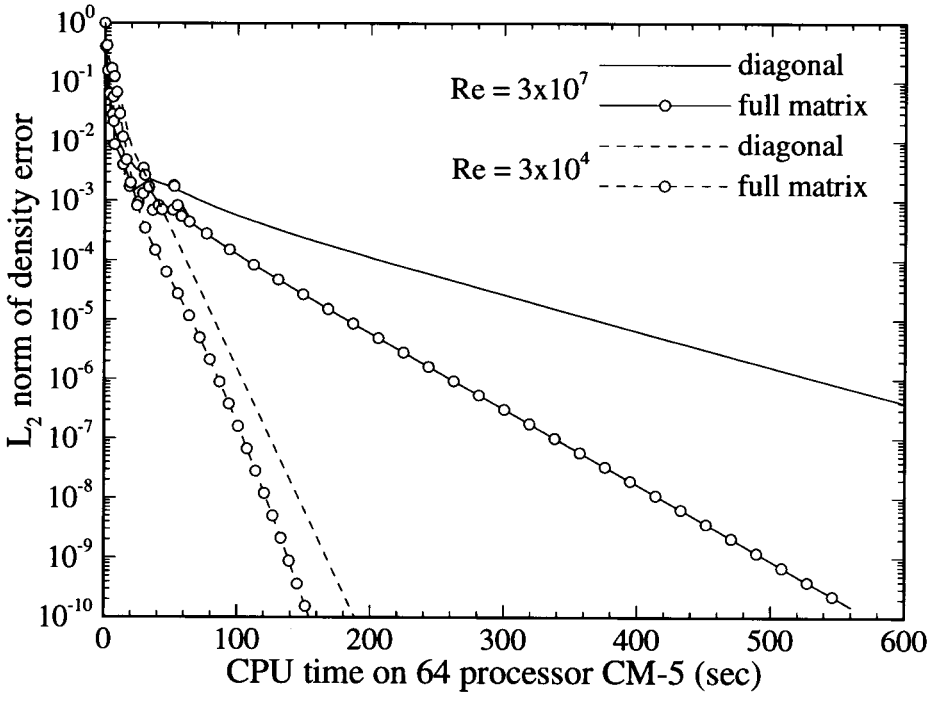
onal method in Fig. (5.4). Results are shown here only for the 2-D implementation; however both 3-D algorithms exhibit nearly identical convergence properties to their 2-D counterparts. Figure (5.4a) plots the convergence rate versus number of iterations for two Reynolds numbers. We see that the full matrix method requires significantly fewer iterations to reach a steady-state solution. The same results are plotted against computer time on a 64 processor CM-5 in Fig. (5.4b). Here we see that although each iteration of the more complex full matrix method takes about 1.7 times as long as the diagonal method for both the two- and three-dimensional implementations, it is clearly the most cost effective approach, especially for high Re flows. For the cases shown in Fig. (5.4), the full matrix DP-LUR method converges about 1.3 times faster for the $Re = 3 \times 10^4$ flow and 2 times faster for the $Re = 3 \times 10^7$ flow.

Most structured grids over complex two- and three-dimensional geometries have regions of high aspect ratio cells, even for low Reynolds number or inviscid flow simulations. Therefore, it is useful to directly study the effects of the cell aspect ratio on the convergence rate of the two DP-LUR methods. Figure (5.5a) presents the computer time on a 64 processor CM-5 required for the density error norm to fall ten orders of magnitude for the two 2-D inviscid algorithms as a function of the maximum cell aspect ratio. We can see that the full matrix method converges slightly faster than the diagonal method even for the lowest CAR grid tested (CAR = 10). This is an interesting result, since the full matrix method was developed specifically to improve high cell aspect ratio performance. As the CAR increases, both methods require more iterations to reach a steady-state solution, but the full matrix method is affected by the grid to a lesser extent, and converges more than two times faster on the highest cell aspect ratio grid tested (CAR = 10,000).

In Fig. (5.5b) we also plot computer time for ten orders of density error norm reduction versus CAR, but this time for a low Reynolds number ($Re = 3000$) flow. For this case the full matrix method is superior on all grids, and in fact is about

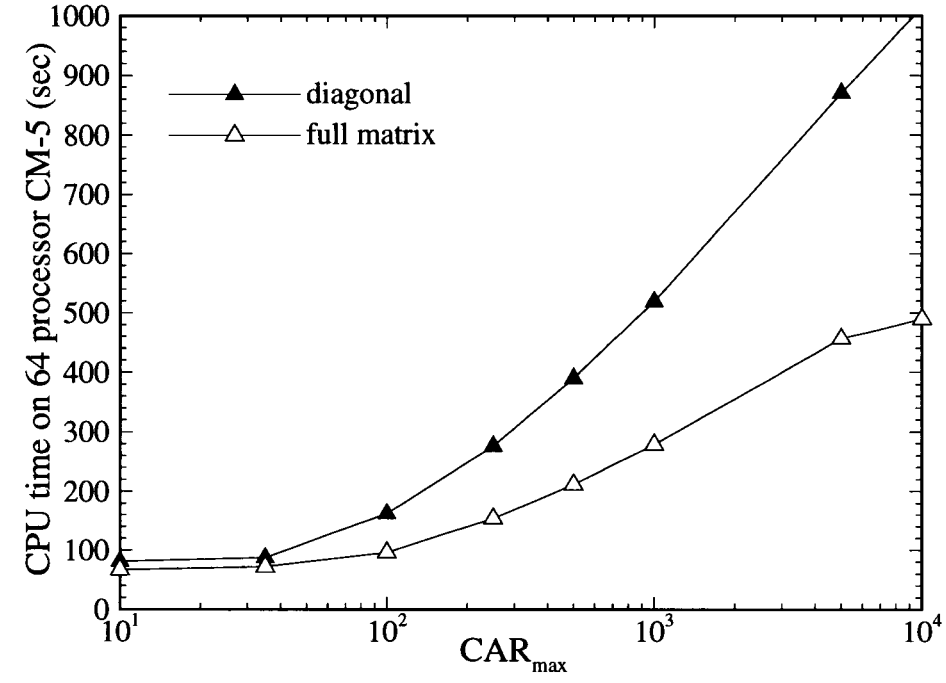


a)

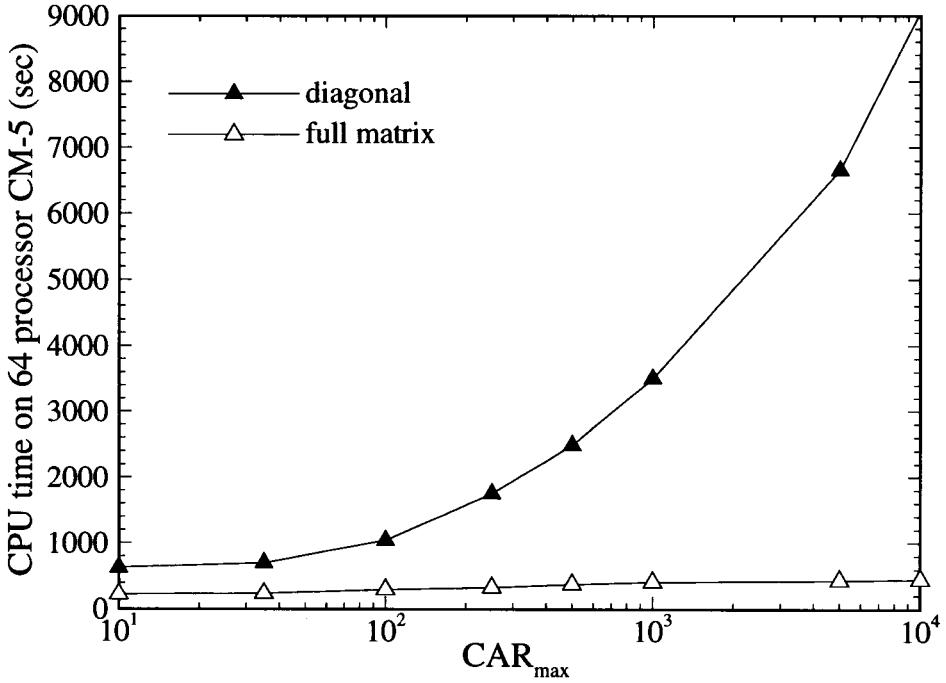


b)

Fig. 5.4 a) Convergence histories and b) CPU times on a 64 processor CM-5, for the full matrix and diagonal DP-LUR methods. 2-D cylinder-wedge blunt body in perfect air at $M_\infty = 15$. 128×128 grid with $y_+ = 1$ for each case.



a)



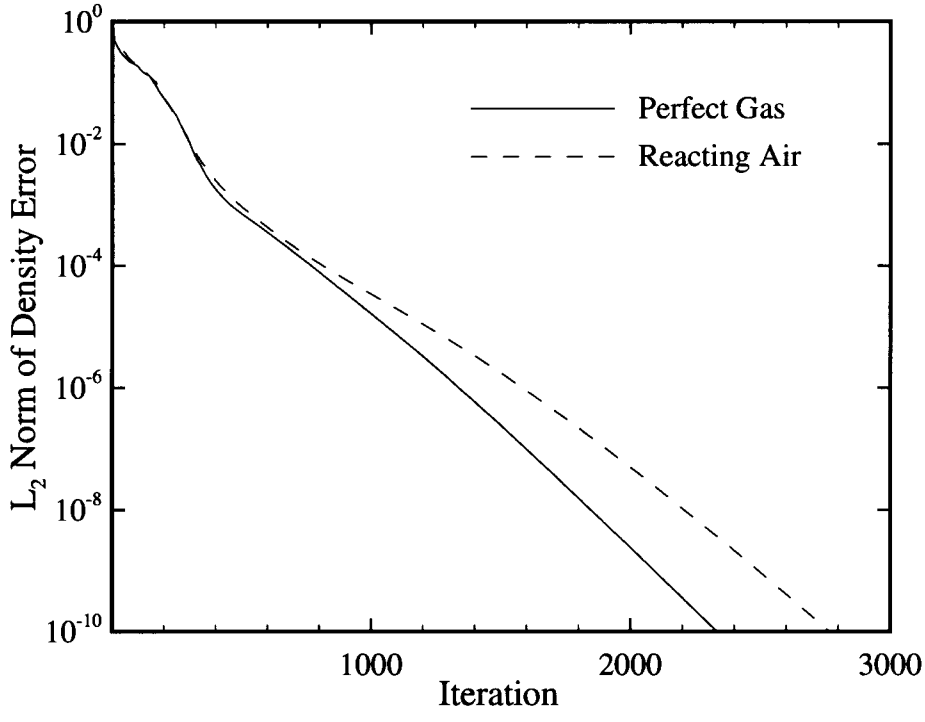
b)

Fig. 5.5 CPU times on a 64 processor CM-5 to achieve ten orders of density error norm reduction for the full matrix and diagonal DP-LUR methods as a function of maximum cell aspect ratio. a) Inviscid, and b) viscous with $Re = 3000$. 2-D cylinder-wedge blunt body in perfect air at $M_\infty = 15$; 128×128 grid.

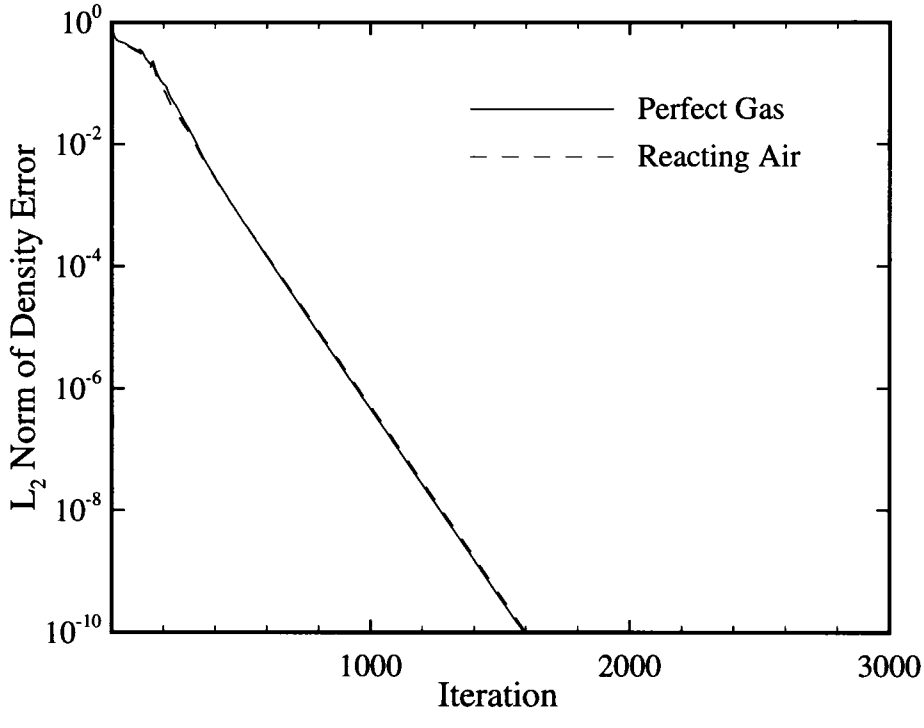
20 times faster on the highest cell aspect ratio grid tested. It is also interesting to directly measure the effect that increasing the CAR has on each of these methods. Dividing the computer time required for the highest CAR grid (10,000) by the time for the lowest aspect ratio grid (10), we see that while the diagonal method takes 14 times longer for the high CAR case, the time required for the full matrix method only doubles. This shows that the more physical approach of the full matrix DP-LUR method is better suited to the solution of highly viscous (low Re) flows.

The results presented in Fig. (5.5) show that the use of the full matrix method reduces, but does not eliminate, the effect of cell aspect ratio on the convergence rate. The remaining dependence is primarily due to the fact that the implicit off-diagonal terms are only indirectly coupled to the diagonal of Eq. (5.5). This will be further examined in the next chapter. Another possibility is suggested by Buelow *et al.* (1994), who show that the high CAR convergence of an ADI based scheme can be improved significantly with the use of viscous preconditioning and method of characteristics boundary conditions. These methods have not been implemented in this work, but it is possible that they may further improve the convergence of both DP-LUR methods.

The full matrix DP-LUR method was next tested with the five species thermo-chemical nonequilibrium air model discussed in Chapter 3. Figure (5.6a) presents a comparison of the convergence histories for a reacting air and perfect gas simulation of a viscous flow with a Reynolds number of 3×10^4. Both calculations are performed on the same computational grid. We see that there is a slight degradation of the convergence rate as compared to the perfect gas simulation. For the case shown here, the reacting air simulation requires about 20% more iterations to reach a ten order of magnitude reduction in the density error norm. Figure (5.6b) shows the same comparison, this time for an inviscid flow. The freestream conditions and computational grid are the same as those used in Fig. (5.6a). For this case we see that there is no decrease in the convergence rate for the reacting air solution. Both



a)



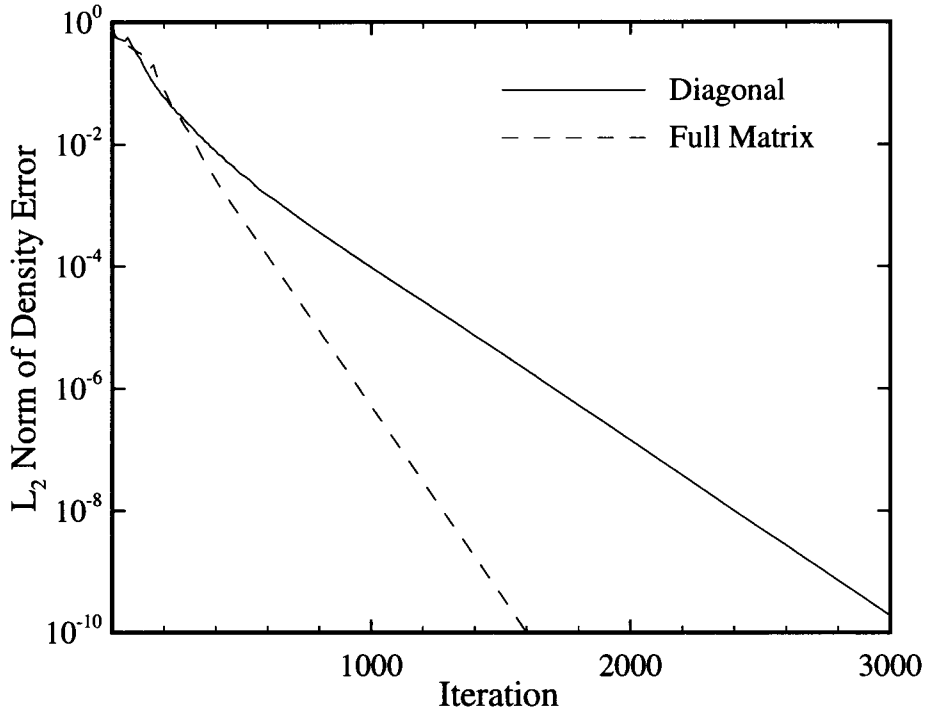
b)

Fig. 5.6 Convergence histories for the full matrix DP-LUR method showing the effect of extension to reacting air problems. 2-D cylinder-wedge blunt body at $M_\infty = 15$ and 69 km altitude. a) Viscous with $Re = 3 \times 10^4$, and b) inviscid. All cases run on the same 128×128 grid.

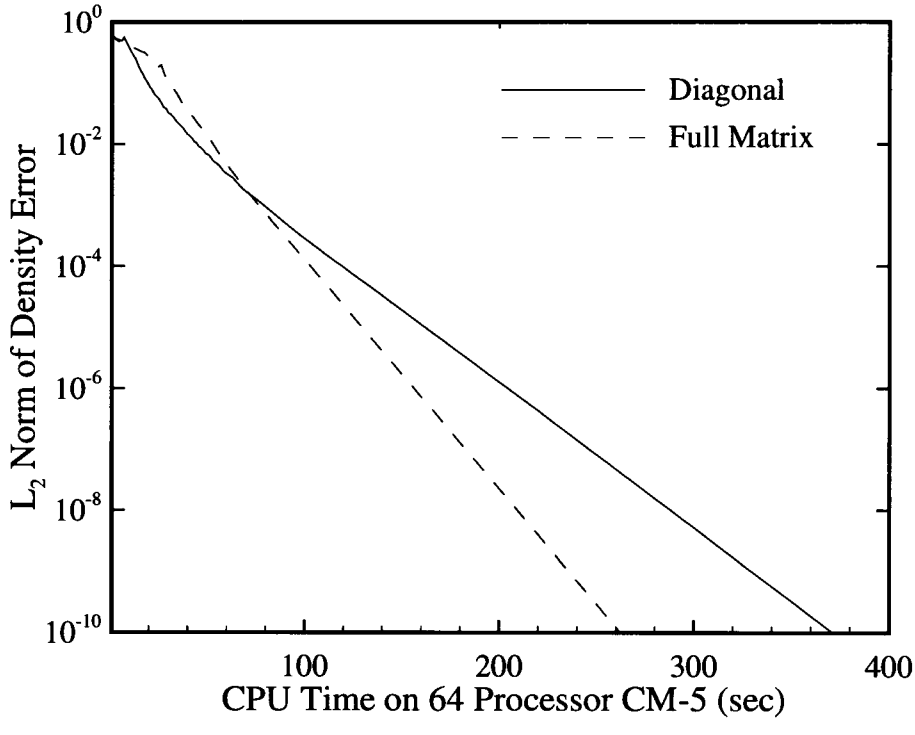
the perfect gas and reacting air codes converge in about 1600 iterations. This implies that there is something about the formulation of the full matrix method that slows the convergence rate when reacting viscous flows are simulated. There are two possibilities that must be explored. First, the viscous Jacobian for the reacting flow case may require modification. Second, the placement of the off-diagonal terms on the right-hand side of the equation may have a greater effect on reacting viscous flows. This problem is still being investigated.

The simulation of reacting flows is an area in which there will be a definite trade-off between the diagonal and full matrix DP-LUR methods. The large equation sets encountered in reacting flow simulations make the diagonal method attractive, since the full matrix method requires the formation and inversion of an exact Jacobian, and therefore the memory required will scale, at least to some extent, with the square of the number of conservation equations. For the 2-D perfect gas implementation, the full matrix DP-LUR method requires about 1.4 times as much memory as the diagonal method, while the 2-D reacting air full matrix method requires about 2.2 times as much memory as the diagonal method. However, the amount of computational work required will still scale nearly linearly with the number of equations, since most of the computational work is due to matrix-vector multiplications. Both the diagonal and full matrix reacting air implementations require about 3 times as many floating point operations per time step as the corresponding perfect gas codes.

The convergence histories for the diagonal and full matrix DP-LUR methods are compared in Fig. (5.7) for a reacting air flow at 69 km altitude. Results are shown for an inviscid flow, so that Reynolds number effects can be avoided. We see in Fig. (5.7a) that the full matrix method reaches a ten order of magnitude reduction in the density error norm in about 1600 iterations, as compared to 3200 iterations for the diagonal method. Figure (5.7b) shows the same information, plotted versus CPU time on a 64 processor CM-5. We see that the full matrix method is also the most



a)



b)

Fig. 5.7 Convergence histories for the full matrix and diagonal DP-LUR methods. 2-D cylinder-wedge blunt body in inviscid reacting air at $M_\infty = 15$ and 69 km altitude; 128×128 grid.

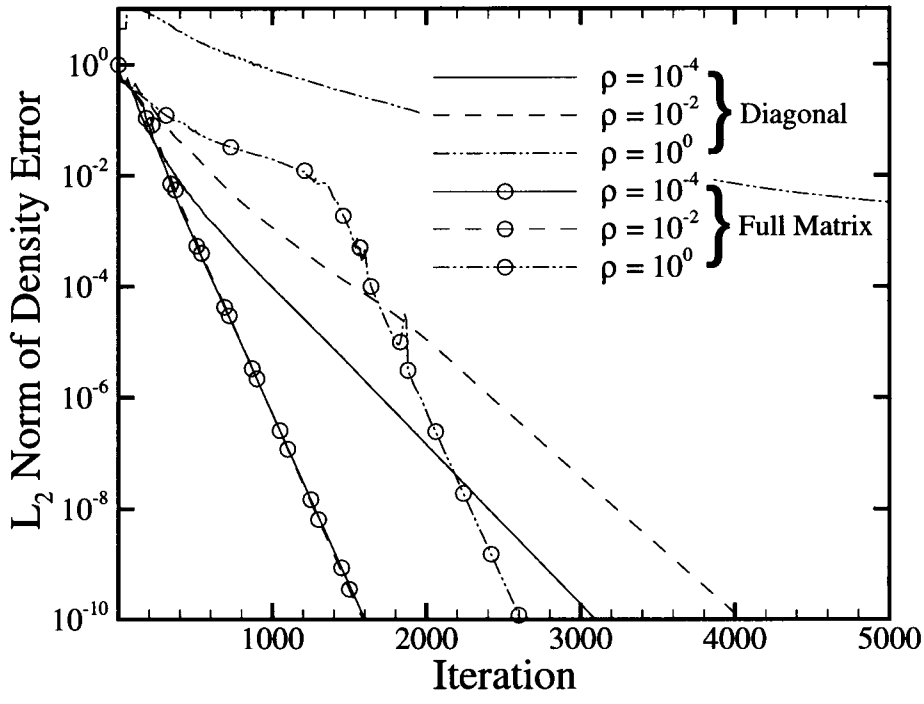


Fig. 5.8 Convergence histories for the reacting air implementations of the diagonal and full matrix DP-LUR methods, showing the effect of source term stiffness, which increases with freestream density. 2-D cylinder-wedge blunt body in inviscid air at $M_\infty = 15$. Same 128×128 grid used for all cases.

cost effective approach, requiring only 260 seconds, as opposed to 370 for the diagonal method.

The diagonal DP-LUR method cannot be easily extended to general chemistry mechanisms, due to the modifications that must be made to enhance the stability and convergence rate of the method, as discussed in Chapter 4. The approximations made to the source term Jacobian will degrade the convergence rate for numerically stiff reacting flows, as the source terms become large compared to the convective and viscous terms. For stiff flows, the exact treatment of the source term Jacobian that is possible with the full matrix approach should greatly improve the convergence rate of the method. This effect can be seen in Fig. (5.8), which plots the convergence histories of the diagonal and full matrix DP-LUR methods for a series of reacting air flows, with freestream densities ranging from $\rho = 10^{-4}$ to $\rho = 10^0$. Since the

reaction rates are strong functions of density, the chemical time scale will decrease rapidly with increasing freestream density, bringing the flow closer to equilibrium and thus increasing the stiffness of the problem. The simulations were run in inviscid air, so that the effects of the finite-rate chemistry could be easily isolated from Reynolds number effects. From Fig. (5.8) we see that while both methods have good convergence characteristics for the least stiff case ($\rho = 10^{-4}$), the convergence rate of the diagonal method degrades as ρ increases. The full matrix method does show some degradation of convergence for the stiffest flow ($\rho = 10^{0}$), but this is primarily due to the non-linear motion of the bow shock through the grid. This effect, combined with the stiff source terms, reduces the maximum stable time step significantly until the shock has reached its final location. However, after the shock motion has stopped the time step can be quickly increased, and we see that the final convergence rate is equal to that of the lower density (less stiff) flows.

An important result from this chapter is that, for all cases shown here, the full matrix DP-LUR method converges faster than the diagonal method. This is perhaps not a surprising result, since the full matrix method is a better physical representation of the problem than the diagonal, and thus we would expect convergence in fewer iterations. However, this does not mean that the diagonal DP-LUR method should be discarded. The diagonal method will remain useful for very large (memory limited) applications, since it does not require the storage of any Jacobian matrices, and therefore uses significantly less memory than the full matrix method. This will be especially important for full three-dimensional vehicle simulations, which can require a large number of grid points.

We have seen that by eliminating the Yoon and Jameson approximation to the Jacobian matrices, we have reduced the effects of cell aspect ratio and source term stiffness on the convergence rate of the DP-LUR method. However, Fig. (5.5) demonstrates that the effect of CAR on the convergence rate is reduced, but not eliminated. In addition, we have seen in Fig. (5.2) that there is still a degradation

in the convergence rate of the full matrix method when viscous flows are simulated. The next chapter proposes a new data-parallel algorithm which will eliminate both of these problems.

Chapter 6

Data-Parallel Line Relaxation Method

6.1 Introduction

While both variations of the DP-LUR method are efficient for the simulation of many problems, they are both affected to some extent by the high cell aspect ratio grids necessary to resolve the boundary layers of high Reynolds number flows. The dependence of the convergence rate on the cell aspect ratio was reduced by the introduction of the full matrix DP-LUR method, but some performance degradation is still apparent. The remaining dependence is primarily due to the fact that these methods place all of the implicit off-diagonal terms on the right-hand side, and thus their effect is only indirectly coupled to the diagonal.

6.2 Gauss-Seidel Line Relaxation Method

A better approach is possible for viscous external flows if we recognize that the viscous flow gradients will be much stronger in the body-normal (η) direction. Thus, the physical problem is much more strongly coupled in the body-normal direction, and it is possible to move just these body-normal implicit terms back to the left-hand side, resulting in the following equation

$$\begin{aligned} \hat{B}_{i,j}\delta U^n_{i,j+1} + \hat{A}_{i,j}\delta U^n_{i,j} - \hat{C}_{i,j}\delta U^n_{i,j-1} = \\ - \hat{D}_{i,j}\delta U^n_{i+1,j} + \hat{E}_{i,j}\delta U^n_{i-1,j} + \Delta U^n_{i,j}, \end{aligned} \tag{6.1}$$

where the 'hat' matrices are defined from the Jacobians as

$$\begin{aligned}
\hat{A}_{i,j} &= I + \frac{\Delta t}{V_{i,j}}\big(\tilde{A}^n_{+i+\frac{1}{2},j} S_{i+\frac{1}{2},j} - \tilde{A}^n_{-i-\frac{1}{2},j} S_{i-\frac{1}{2},j} + \\
&\qquad \tilde{B}^n_{+i,j+\frac{1}{2}} S_{i,j+\frac{1}{2}} - \tilde{B}^n_{-i,j-\frac{1}{2}} S_{i,j-\frac{1}{2}}\big) \\
\hat{B}_{i,j} &= \frac{\Delta t}{V_{i,j}} \tilde{B}^n_{-i,j+\frac{1}{2}} S_{i,j+\frac{1}{2}} \\
\hat{C}_{i,j} &= \frac{\Delta t}{V_{i,j}} \tilde{B}^n_{+i,j-\frac{1}{2}} S_{i,j-\frac{1}{2}} \\
\hat{D}_{i,j} &= \frac{\Delta t}{V_{i,j}} \tilde{A}^n_{-i+\frac{1}{2},j} S_{i+\frac{1}{2},j} \\
\hat{E}_{i,j} &= \frac{\Delta t}{V_{i,j}} \tilde{A}^n_{+i-\frac{1}{2},j} S_{i-\frac{1}{2},j}.
\end{aligned} \tag{6.2}$$

In this way it is possible to solve Eq. (6.1) for all j-points at each i-location as a series of fully coupled block tri-diagonal systems aligned in the body-normal direction. This was the approach used by MacCormack (1985) in the Gauss-Seidel Line Relaxation (GSLR) method. In this method the problem is solved via a series of alternating forward and backward sweeps through the flowfield in the i-direction, using the latest available data for the terms on the right-hand side. The forward sweeps consist of the solution of

$$\begin{aligned}
\hat{B}_{i,j}\delta U^{(k)}_{i,j+1} + \hat{A}_{i,j}\delta U^{(k)}_{i,j} - \hat{C}_{i,j}\delta U^{(k)}_{i,j-1} = \\
- \hat{D}_{i,j}\delta U^{(k-1)}_{i+1,j} + \hat{E}_{i,j}\delta U^{(k)}_{i-1,j} + \Delta U^n_{i,j}, \qquad \text{for } k = 1, 3, \ldots
\end{aligned} \tag{6.3a}$$

and the backward sweeps consist of

$$\begin{aligned}
\hat{B}_{i,j}\delta U^{(k)}_{i,j+1} + \hat{A}_{i,j}\delta U^{(k)}_{i,j} - \hat{C}_{i,j}\delta U^{(k)}_{i,j-1} = \\
- \hat{D}_{i,j}\delta U^{(k)}_{i+1,j} + \hat{E}_{i,j}\delta U^{(k-1)}_{i-1,j} + \Delta U^n_{i,j}, \qquad \text{for } k = 2, 4, \ldots
\end{aligned} \tag{6.3b}$$

The index k is the sweep counter. Therefore, a superscript k means that information from the current sweep should be used for δU, while a superscript $k-1$ indicates that the value should be taken from the previous sweep. During the first sweep ($k = 1$) we set $\delta U^{(0)} = 0$. In principle, any number of sweeps can be used, however for steady-state problems two or four sweeps each iteration are usually sufficient to maintain stability.

Since Eq. (6.3) is a much more exact representation of the problem than the equations developed in Chapters 4 and 5, it is no longer acceptable to implement the implicit boundary conditions simply by setting the boundary δU's to zero before beginning the implicit routine. However, it is straightforward to implement the boundary conditions by folding them into the appropriate 'hat' matrices [MacCormack (1985)]. This procedure is outlined briefly here. Consider the case where a solid wall boundary exists at the $j = 3/2$ cell surface. If we write the implicit equation for this case for the $j=2$ line, we get

$$\begin{aligned} \hat{B}_{i,2}\delta U^n_{i,3} + \hat{A}_{i,2}\delta U^n_{i,2} - \hat{C}_{i,2}\delta U^n_{i,1} = \\ - \hat{D}_{i,2}\delta U^n_{i+1,2} + \hat{E}_{i,2}\delta U^n_{i-1,2} + \Delta U^n_{i,2}. \end{aligned} \tag{6.4}$$

From this equation, we see that we need to solve for the boundary value $\delta U_{i,1}$, which is outside the solution domain. We can do this simply by relating the value of δU at $j = 1$ to that at $j = 2$, which is the first interior cell center. Since a solid wall exists between $j = 1$ and $j = 2$, we know that there can be no flow through the boundary. In addition, if the wall is viscous, we can enforce the no-slip condition. The remainder of the flow variables can then be determined by zeroth-order extrapolation, conservation of total temperature, or other means as appropriate. Using this analysis we can write $\delta U_{i,1}$ in terms of $\delta U_{i,2}$, using an appropriate rotation matrix, R. For example,

$$\delta U^n_{i,1} = R_{i,1}\delta U^n_{i,2}, \tag{6.5}$$

Then Eq. (6.4) for the $j=2$ line of cells becomes

$$\begin{aligned} \hat{B}_{i,2}\delta U^n_{i,3} + \hat{A}_{i,2}\delta U^n_{i,2} - \hat{C}_{i,2}R_{i,1}\delta U^n_{i,2} = \\ - \hat{D}_{i,2}\delta U^n_{i+1,2} + \hat{E}_{i,2}\delta U^n_{i-1,2} + \Delta U^n_{i,2}. \end{aligned} \tag{6.6}$$

There are now two terms in Eq. (6.6) that operate on $\delta U_{i,2}$, so they can be folded together, as in

$$\begin{aligned} \hat{B}_{i,2}\delta U^n_{i,3} + \mathring{A}_{i,2}\delta U^n_{i,2} = \\ - \hat{D}_{i,2}\delta U^n_{i+1,2} + \hat{E}_{i,2}\delta U^n_{i-1,2} + \Delta U^n_{i,2}, \end{aligned} \tag{6.7}$$

where $\mathring{A}_{i,2} = \hat{A}_{i,2} - \hat{C}_{i,2}R_{i,1}$. A similar process is used for other types of solid wall and flow-through boundary conditions. For more details, see MacCormack (1985) or Candler (1988).

The Gauss-Seidel Line Relaxation method works well for two-dimensional flows on a serial or vector machine, but it is not straightforward to extend the method to three-dimensional flows. The obvious approach to the three-dimensional case is to continue to set up the problem as a series of block tri-diagonal systems normal to the body, and sweep in both the axial and circumferential directions. However, this approach can lead to a non-physical bias in the converged solution due to the biasing that is inherent in the Gauss-Seidel sweeping process [MacCormack (1990)]. In addition, the data-dependencies in the Gauss-Seidel sweeps would make the algorithm inefficient on a parallel machine. However, it is possible to modify the GSLR method to make it amenable to a data-parallel implementation, as we shall see in the next section.

6.3 Data-Parallel Line Relaxation Method

The data-dependencies inherent in the GSLR method can be eliminated by replacing the Gauss-Seidel sweeps with a series of line relaxation steps, using a procedure similar to that outlined in the previous chapters for the DP-LUR method [Wright *et al.* (1997a)]. The resulting Data-Parallel Line Relaxation (DPLR) method is then described by following scheme. First, the implicit terms on the right hand side of Eq. (6.1) are neglected and the resulting block tri-diagonal system is factored and solved for $\delta U^{(0)}$

$$\hat{B}_{i,j}\delta U^{(0)}_{i,j+1} + \hat{A}_{i,j}\delta U^{(0)}_{i,j} - \hat{C}_{i,j}\delta U^{(0)}_{i,j-1} = \Delta U^{n}_{i,j}.$$

Then a series of k_{max} relaxation steps are made using

$$\text{for } k = 1,\ k_{max}$$

$$\hat{B}_{i,j}\delta U^{(k)}_{i,j+1} + \hat{A}_{i,j}\delta U^{(k)}_{i,j} - \hat{C}_{i,j}\delta U^{(k)}_{i,j-1} = - \hat{D}_{i,j}\delta U^{(k-1)}_{i+1,j} + \hat{E}_{i,j}\delta U^{(k-1)}_{i-1,j} + \Delta U^{n}_{i,j}, \tag{6.8}$$

then

$$\delta U^{n}_{i,j} = \delta U^{(k_{max})}_{i,j}.$$

The DPLR method requires a single lower-upper (LU) factorization and $k_{max} + 1$ back substitutions per iteration. By using this approach, all of the data required for each relaxation step have already been computed during the previous step. Therefore, as long as the data are distributed on the processors in such a way that the body-normal direction is entirely local, the entire relaxation step can be performed simultaneously in parallel with no data-dependencies. This approach works well for a coarse or medium-grained parallel computer. However, when the number of processors becomes larger than the number of points in the i-direction, it is no longer possible to lay out the data so that the body-normal direction is entirely local, and thus the required block tri-diagonal factorization and back-substitutions must be distributed across the processors. There have been several algorithms proposed for the efficient parallel solution of a distributed block tri-diagonal system [Evans (1984) & Amodio and Mazzia (1995)], however these techniques are not used in this research, since we are dealing with medium-grained parallel computers.

Another advantage of the DPLR method is that since the implicit off-diagonal terms are all equally lagged by one relaxation step, the biasing problem no longer exists, and implementation of a three-dimensional version of the algorithm becomes straightforward. The DPLR approach will use significantly more memory than the full matrix and diagonal DP-LUR method, since five Jacobian matrices (seven for three-dimensional flows) must now be computed and stored at each grid point, as compared to one for the full matrix DP-LUR method and none for the diagonal method. For perfect gas flows the DPLR method uses about twice as much mem-

ory as the full matrix DP-LUR method for the two-dimensional implementation, and four times as much for the three-dimensional version. As the number of equations increases, the storage required for the Jacobians will scale quadratically with the number of equations, and thus will grow much faster than the storage required for other arrays, which will scale approximately linearly with the number of equations. Therefore, for very large equation sets we expect the memory use of the DPLR method to be about five times as much as the full matrix method for 2-D flows, and seven times as much for 3-D flows. However, the DPLR method uses no more memory than the original GSLR method, and should converge much faster than either DP-LUR approach, due to the more exact formulation of the implicit operator.

6.4 Performance of the DPLR Method

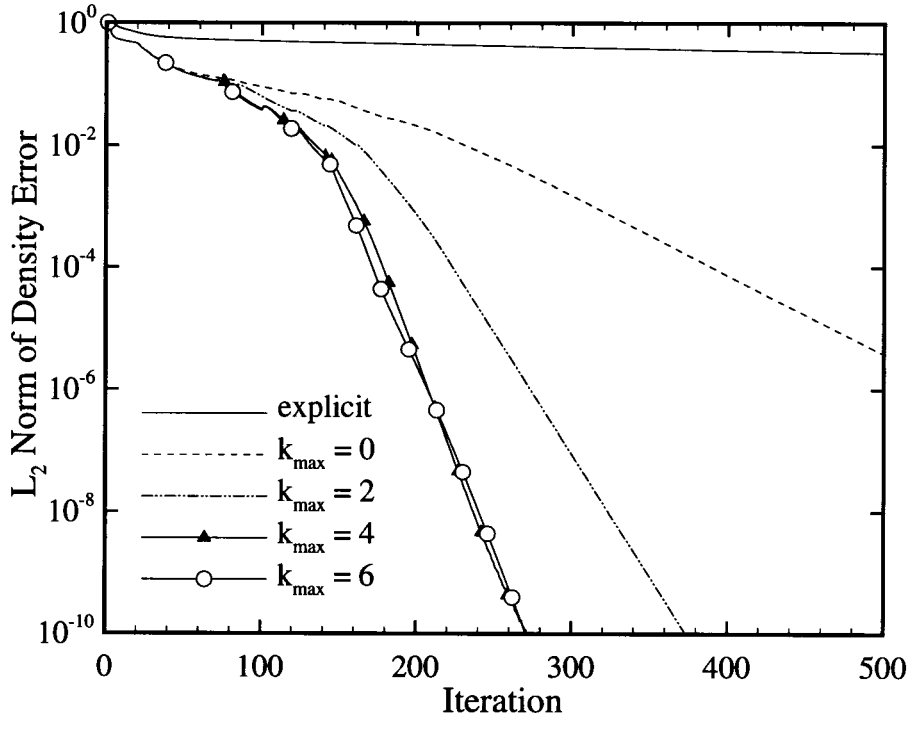
The perfect gas implementation of the DPLR method has been tested on two- and three-dimensional geometries with an emphasis on evaluating the convergence properties and parallel performance of the new method. The primary test remains the same as that used in Chapters 4 and 5: the Mach 15 perfect gas flow over a cylinder-wedge blunt body, with Reynolds numbers based on the freestream conditions and body length varying from 3×10^4 to 3×10^8. The three-dimensional computations were performed on multiple planes of the same 2-D grids, which makes it easy to directly compare the convergence properties of the two- and three-dimensional implementations of the method. At this time only a perfect gas version of the DPLR method has been implemented, therefore no results are shown for reacting air simulations.

Although the results presented here are based on modified first-order Steger-Warming flux vector splitting for the numerical flux evaluation, it is important to note that the derivation of the DPLR method is general and can be used with many flux evaluation methods. In fact Liou and Van Leer (1988) have shown that the use of Steger-Warming splitting can be effective and robust for the implicit part of the

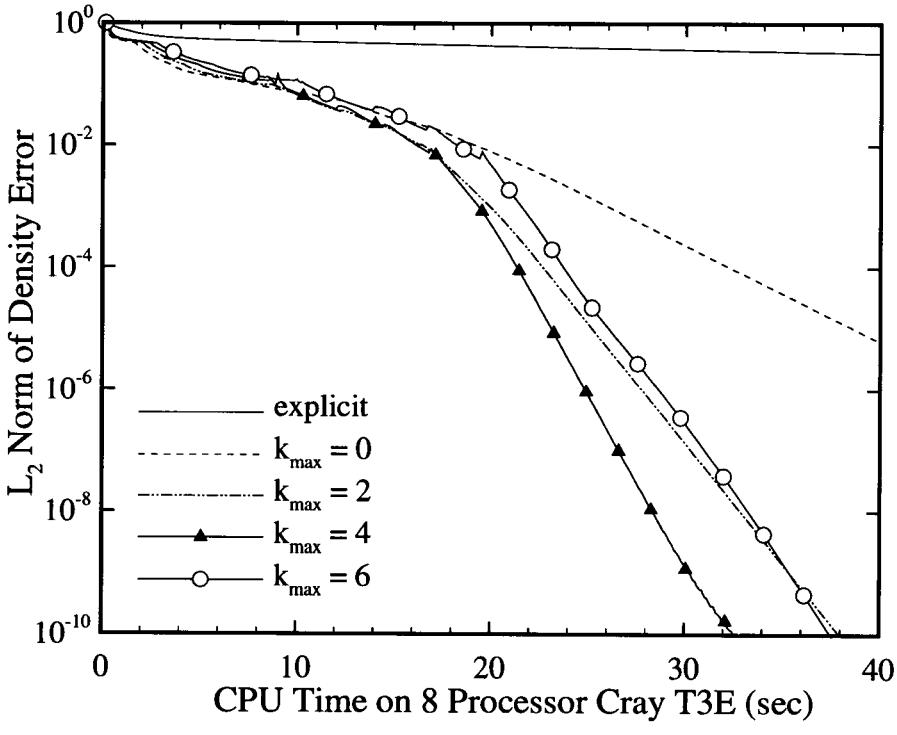
problem, even when the fluxes are evaluated by a different method.

The DPLR method is in all instances more sensitive to the size of the implicit time step than the DP-LUR method, as expected, since the DPLR method involves a more exact representation of the problem, and therefore the size of the time step will have more physical meaning. In all cases presented here the implicit time step was chosen to correspond to a CFL number of one in the first iteration, and was rapidly increased to its maximum stable value. The size of the maximum stable time step for each case was governed by the freestream conditions, with no limitation due to the mesh spacing. Therefore the maximum CFL number varied considerably from case to case.

Figure (6.1a) shows the effect of the number of relaxation steps (k_{max}) on the convergence rate of the method for the 2-D cylinder-wedge geometry at a Reynolds number of 3×10^4 and $y_+ = 1$. The line marked $k_{max} = 0$ corresponds to performing just the initial block tri-diagonal solution along each i-line, with no implicit coupling in the i-direction. The $k_{max} = 0$ approach is similar to that proposed by Wang and Widhopf (1990), and later implemented in parallel by Taylor and Wang (1995). Although the $k_{max} = 0$ approach offers a significant improvement over the first-order Euler explicit method, we see from Fig. (6.1a) that the convergence rate of the method improves when the effect of the implicit i-direction terms is included ($k_{max} > 0$). The DPLR method shows a similar dependence on k_{max} to the DP-LUR methods, with the convergence rate steadily improving as k_{max} increases, up to $k_{max} = 6$. Figure (6.1b) shows the effect of the number of relaxation steps on the cost of the method, evaluated as the total CPU time on an eight processor Cray T3E. Since the cost of evaluating the Jacobian matrices and setting up the LU factored block tri-diagonal system is much greater than the cost of performing additional back substitutions, we see that increasing k_{max} also improves the cost effectiveness of the method, up to $k_{max} = 4$. As shown in the figure, values of k_{max} larger than 4 give little improvement in convergence, and are not cost effective.



a)



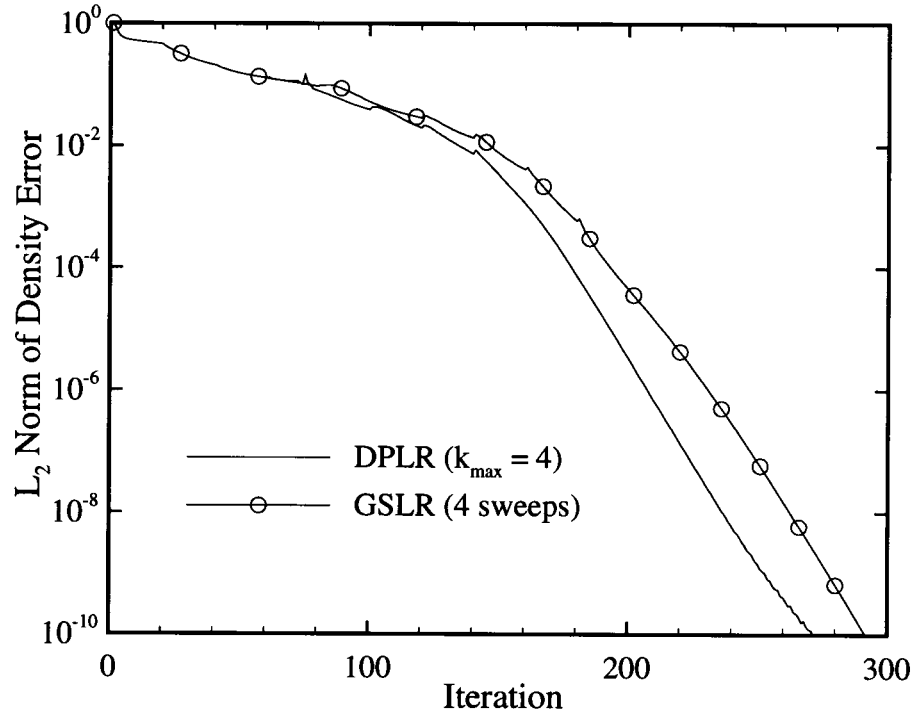
b)

Fig. 6.1 a) Convergence histories and b) CPU times on an 8 processor Cray T3E for the DPLR method showing influence of k_{max}. 2-D cylinder-wedge blunt body in perfect air at $M_\infty = 15$ and $Re = 3 \times 10^4$; 128×128 grid with $y_+ = 1$.

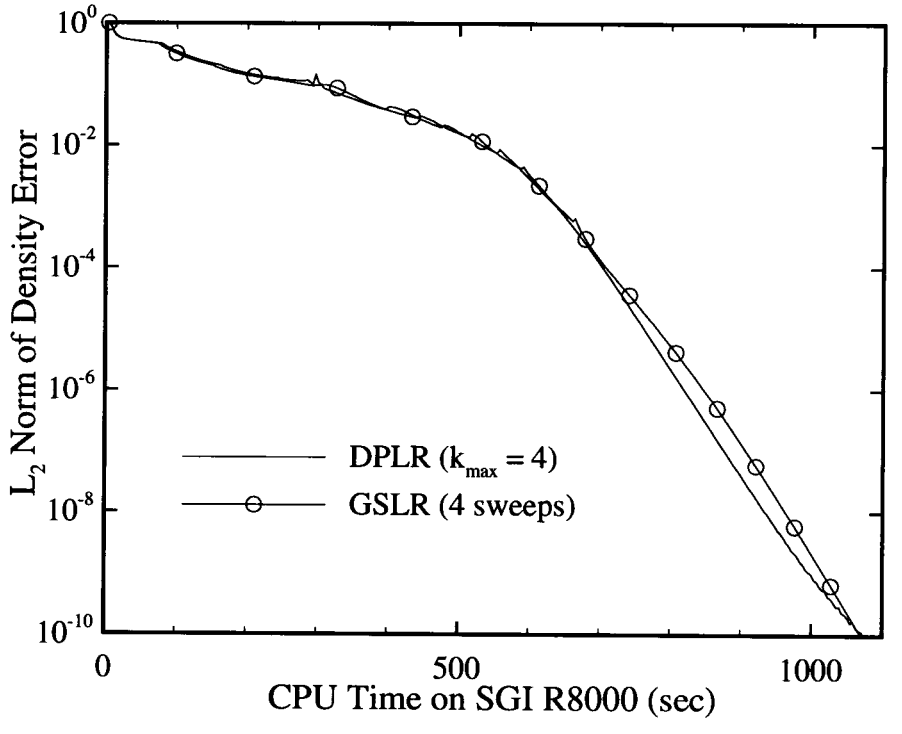
Therefore, all of the results presented in this paper were run at $k_{max} = 4$.

The convergence rate of the new DPLR method is compared to the original Gauss-Seidel Line Relaxation method in Fig. (6.2). Both methods are tested at the same conditions as in Fig. (6.1). Figure (6.2a) compares the number of iterations required for each method to reach a ten order of magnitude reduction in the density error norm. We see that both methods behave similarly, with the convergence rate increasing significantly after about 150 iterations. The slower convergence rate at the beginning of the solution is due to the motion of the bow shock through the computational grid. Since this is a highly non-linear process, the shock will move at most one computational cell per iteration. However, once the bow shock has reached its final location, the block tri-diagonal solutions rapidly drive the error norm toward machine zero. We see that although both methods achieve a ten order of magnitude reduction in the density error norm in less than 300 iterations, the DPLR method using $k_{max} = 4$ actually performs a little better than the GSLR method with four sweeps through the flowfield. This is because the DPLR method allows larger time steps to be used for this case. If both methods are run with the same time step their performance is nearly identical. This is surprising, since the GSLR method always uses latest available data, and thus should allow information to travel across the entire computational domain during each implicit iteration, while with the DPLR method information can travel only k_{max} cells per iteration in the axial (i) direction. However, both methods are identical in their treatment of the body-normal terms.

A direct comparison of the CPU time required for the GSLR and DPLR methods on a parallel machine would be meaningless, due to the poor parallel efficiency of the GSLR method. However, it is possible to get some idea of the relative amount of work required for the each method by comparing CPU times on a single processor machine. Figure (6.2b) shows this comparison on an SGI R8000 workstation. We see that the cost effectiveness of the two methods is almost equal for this problem.



a)



b)

Fig. 6.2 a) Convergence histories and b) CPU times on a single processor SGI R8000 machine for the DPLR and GSLR methods. 2-D cylinder-wedge blunt body in perfect air at $M_\infty = 15$ and $Re = 3 \times 10^4$. 128×128 grid with $y_+ = 1$.

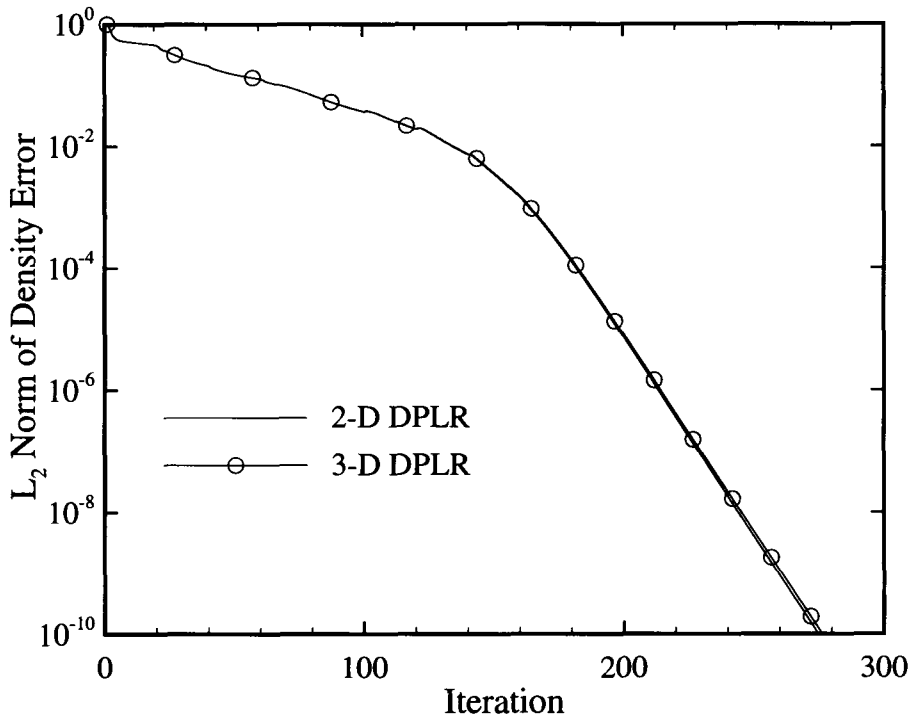


Fig. 6.3 Convergence histories for the 2-D and 3-D versions of the DPLR method. Cylinder-wedge blunt body in perfect air at $M_\infty = 15$ and $Re = 3 \times 10^4$; 128×128 grid with $y_+ = 1$.

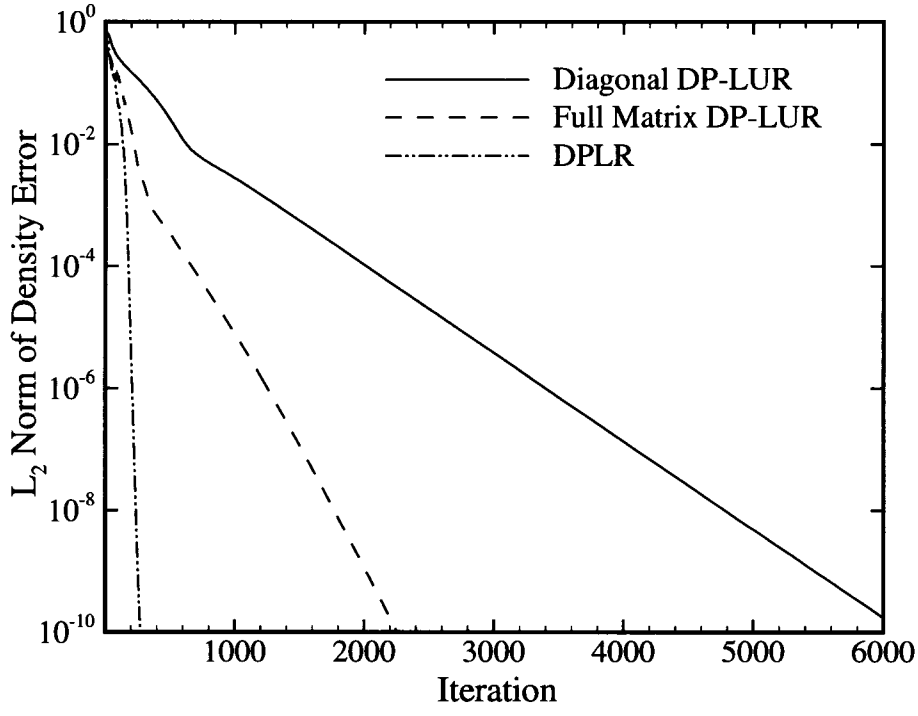
This is because the cost of the two implicit routines is nearly identical. The only difference in the number of floating point operations is the additional back substitution required by the DPLR method for the $k = 0$ step. For the 2-D perfect gas implementation of the algorithms this adds an additional 90 floating point operations to the 1028 required for the implicit portion of the GSLR method. As a result, each iteration of the GSLR method takes 3.67 seconds on the SGI workstation, while each iteration of the DPLR method requires 3.95 seconds.

Figure (6.3) plots the convergence rate of the two- and three-dimensional DPLR methods for the cylinder-wedge blunt body using the same test conditions as the previous figure. The 3-D calculation was performed on 16 identical planes of the baseline 128×128 grid. We see that there is essentially no difference in the convergence rate between the 2-D and 3-D implementations. The same behavior is seen for all cases tested to date. By performing the three-dimensional calculation on

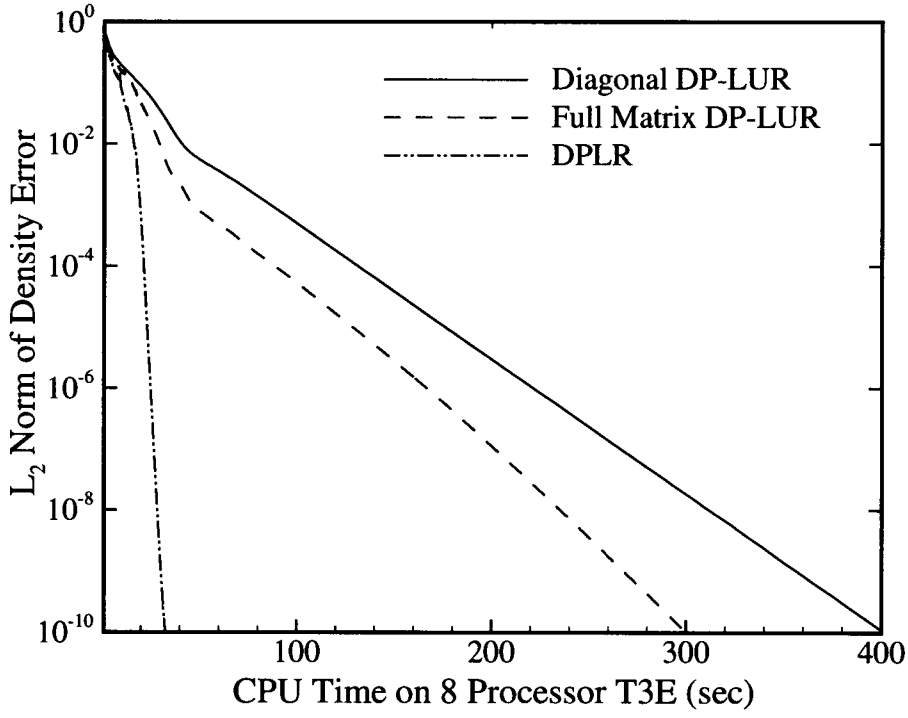
multiple planes of a 2-D grid we can also easily check for any evidence of the solution bias exhibited by the 3-D GSLR method. This effect is always more noticeable in such cases, since any solution bias will tend to create a non-physical cross-flow velocity in the direction in which the multiple planes are projected. This will be evident even before any other differences can be detected between the solutions on different planes. For the case shown in Fig. 5, the maximum cross-flow velocity in the flowfield is 12 orders of magnitude smaller than the freestream velocity. We feel that this value is sufficiently small to be attributed to machine roundoff errors, and thus the DPLR method has eliminated the solution bias problem.

The new DPLR method is compared to the diagonal and full matrix DP-LUR methods in Fig. (6.4) for the cylinder-wedge blunt body at a Reynolds number of 3×10^4 and $y_+ = 1$. All three methods are run at $k_{max} = 4$. We see in Fig. (6.4a) that the DPLR method is very efficient, achieving ten orders of magnitude reduction in the density error norm in less than 300 iterations, as compared to 2300 iterations for the full matrix method, and 6200 iterations for the diagonal DP-LUR method. However, convergence in fewer iterations does not necessarily imply that the method will be more cost effective on a parallel machine. It is also necessary to know how much each iteration of the method will cost. Therefore, in order to examine the effectiveness of the new algorithm, Fig. (6.4b) compares the cost of the methods, plotted as CPU time on an eight processor Cray T3E. From Fig. (6.4b) we see that the DPLR method also has a high parallel efficiency, and is by far the most cost effective of the three methods. For this problem the Data-Parallel Line Relaxation method reaches a ten order of magnitude reduction in the density error norm in just 33 seconds, compared to 300 for the full matrix DP-LUR method and 400 for the diagonal method. This shows that the DPLR method can potentially be a powerful tool for the simulation of viscous flows.

Figure (6.5) shows the convergence histories of the DPLR method for three viscous flows with Reynolds numbers ranging from 3×10^4 to 3×10^8. The 128×128



a)



b)

Fig. 6.4 a) Convergence histories and b) CPU times on an 8 processor Cray T3E for the DPLR method as compared to the two DP-LUR methods. 2-D cylinder-wedge blunt body in perfect air at $M_\infty = 15$ and $Re = 3 \times 10^4$; 128×128 grid with $y_+ = 1$.

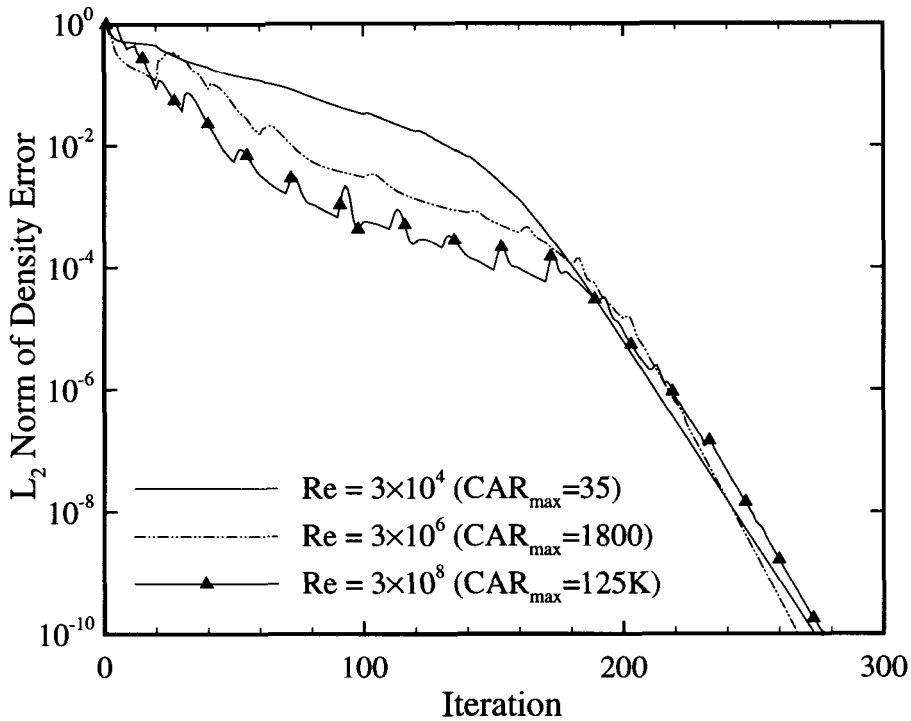


Fig. 6.5 Convergence histories for the DPLR method showing influence of Reynolds number. 2-D cylinder-wedge blunt body at $M_\infty = 15$; 128 × 128 grid with $y_+ = 1$ for each case.

grid for each case was chosen so that $y_+ = 1$ for the first cell above the body, ensuring that the boundary layers for all cases are equally well resolved. Since the boundary layer thickness decreases with increasing Re, the maximum cell aspect ratio (CAR) of the grid must increase as well to meet the $y_+ = 1$ requirement. For the test cases in Fig. (6.5), the maximum CAR ranges from 35 for $Re = 3 \times 10^4$ to about 125,000 for $Re = 3 \times 10^8$. From the figure we can see that, while each of the cases behaves differently during the early phases of the flow evolution, all converge at nearly the same rate after the bow shock reaches its final location. In addition, there is essentially no increase in the number of iterations required to reach a steady-state solution as the Reynolds number (and therefore the cell aspect ratio) is increased.

Figure (6.6) compares the convergence properties of the DPLR method with the DP-LUR methods on several viscous flows with Reynolds numbers ranging from

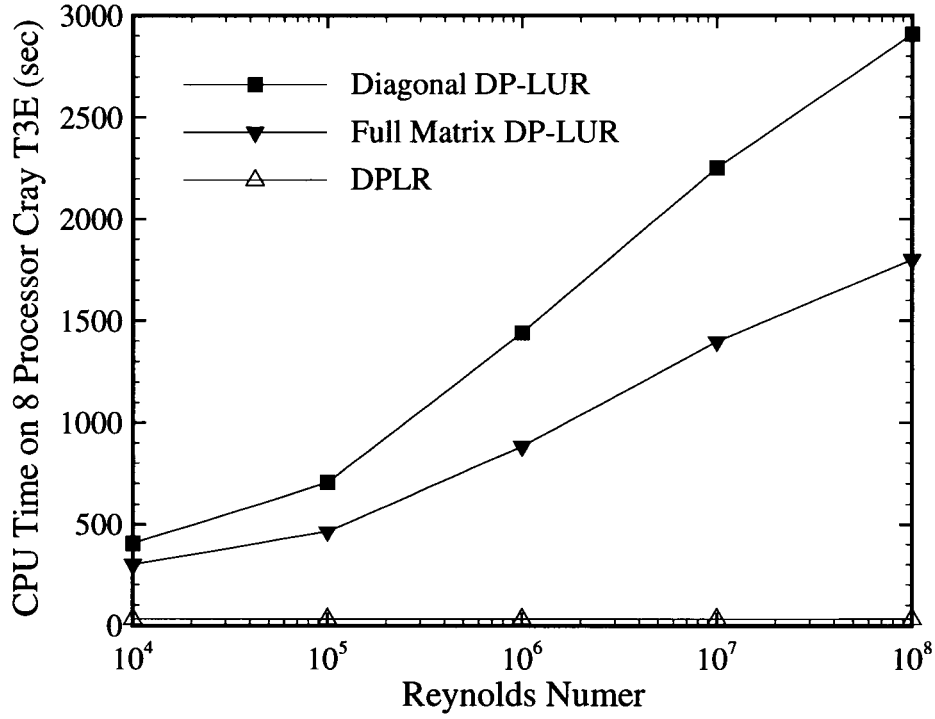


Fig. 6.6 CPU times on an 8 processor Cray T3E required to achieve 10 orders of error norm convergence for the DPLR and DP-LUR methods as a function of Reynolds number. 2-D cylinder-wedge blunt body in perfect air at $M_\infty = 15$; 128×128 grid with $y_+ = 1$ for each case.

3×10^4 to 3×10^8. Once again, the 128×128 grid for each case was chosen so that $y_+ = 1$ for the first cell above the body. As the Reynolds number is increased by four orders of magnitude, the amount of computer time required to achieve a ten order of magnitude reduction in the density error norm remains constant for the DPLR method, while the time increases by a factor of 6 for the full matrix method and 7 for the diagonal method. This shows that the more exact implicit operator used in the DPLR method eliminates the convergence degradation on high cell aspect ratio grids.

Figure (6.7) compares the viscous and inviscid implementations of the DPLR method for two of the cases in Fig. (6.6). The grid for each case was chosen to satisfy the $y_+ = 1$ condition for the viscous solution. The inviscid solutions were then obtained on the same computational grids. We see that the convergence rates

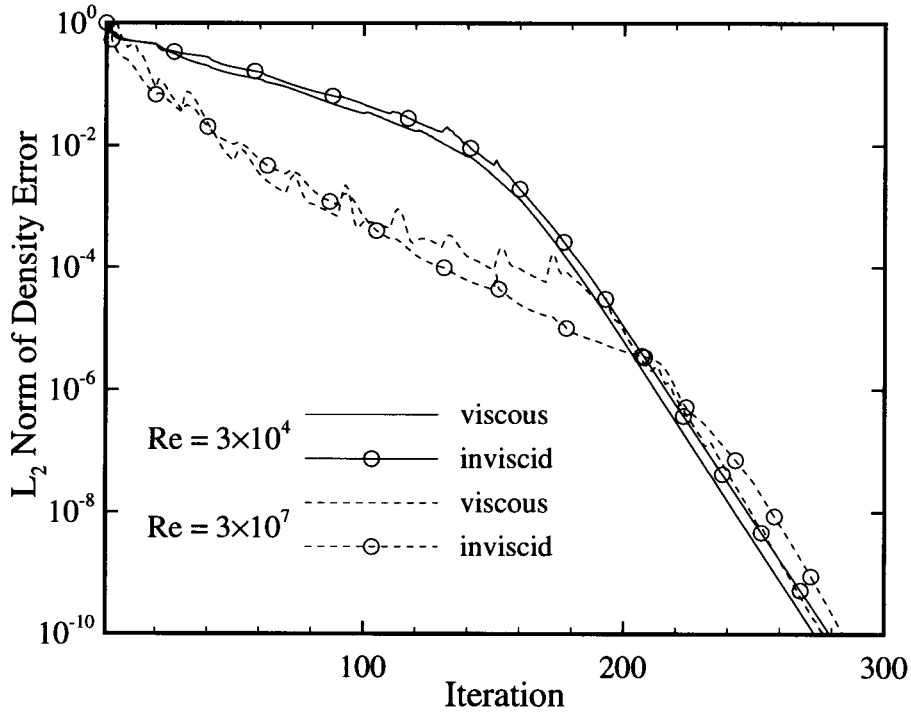
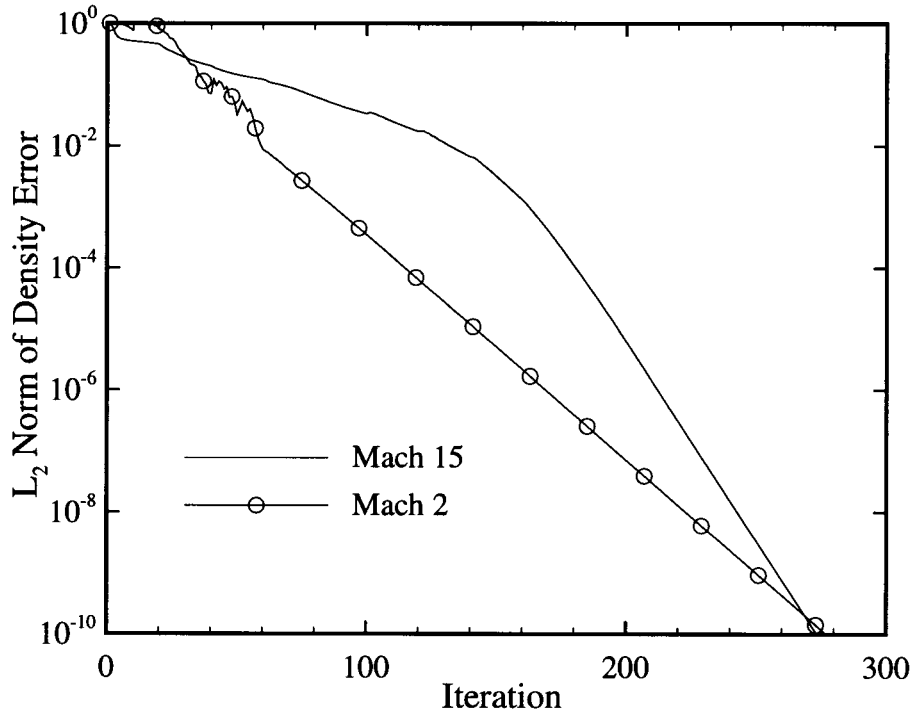


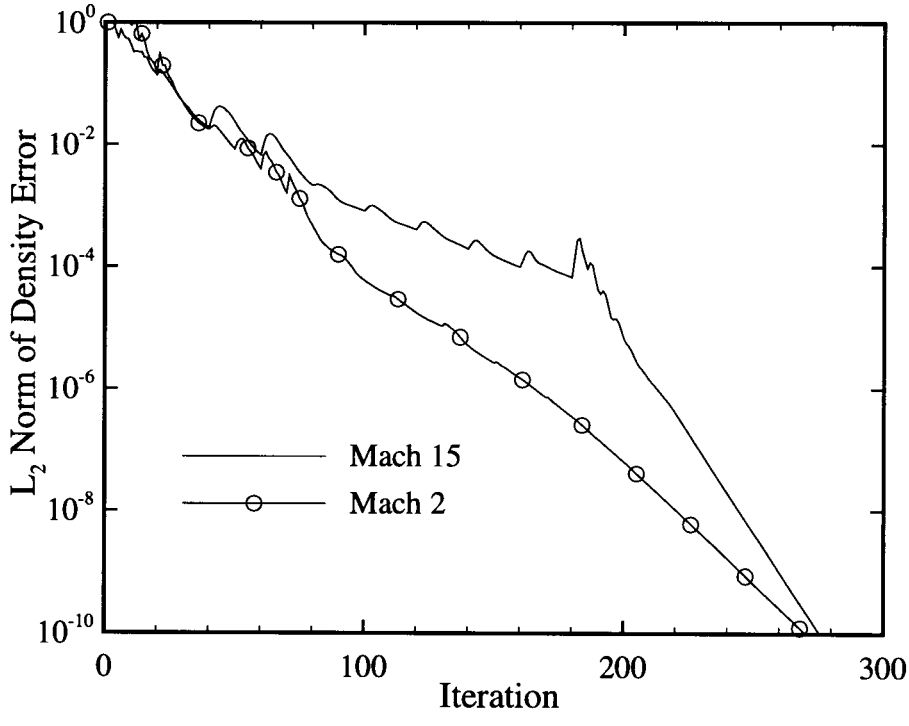
Fig. 6.7 Convergence histories for the viscous and inviscid implementations of the DPLR method. 2-D cylinder-wedge blunt body at $M_\infty = 15$; 128×128 grid with $y_+ = 1$ for each viscous case.

for the viscous and inviscid versions of the method are almost identical. This is in direct contrast to the diagonal and full matrix DP-LUR methods, which always require more iterations to converge the viscous solution. This again shows the benefit of moving the body-normal terms back to the left-hand side of the equation and coupling them directly to the diagonal.

Figure (6.8) compares the convergence rate of the DPLR method on two cylinder-wedge flows at Mach 15 and Mach 2 for two different Reynolds numbers. Both cases in Fig. (6.8a) are run at a Reynolds number of 3×10^4, and both cases in Fig. (6.8b) are run at a Reynolds number of 3×10^7. We see that all four cases reach a ten order of magnitude reduction in the density error norm in less than 300 iterations. However, there are differences in the convergence histories. The low Mach number cases require fewer iterations for the bow shock to reach its final location, since the low Mach number flow is less non-linear. In addition, once the



a)



b)

Fig. 6.8 Convergence histories for the DPLR method on a high Mach number and low supersonic Mach number flow. 2-D cylinder-wedge blunt body in perfect air. a) $Re = 3 \times 10^4$ and b) $Re = 3 \times 10^7$; 128×128 grid with $y_+ = 1$ for each case.

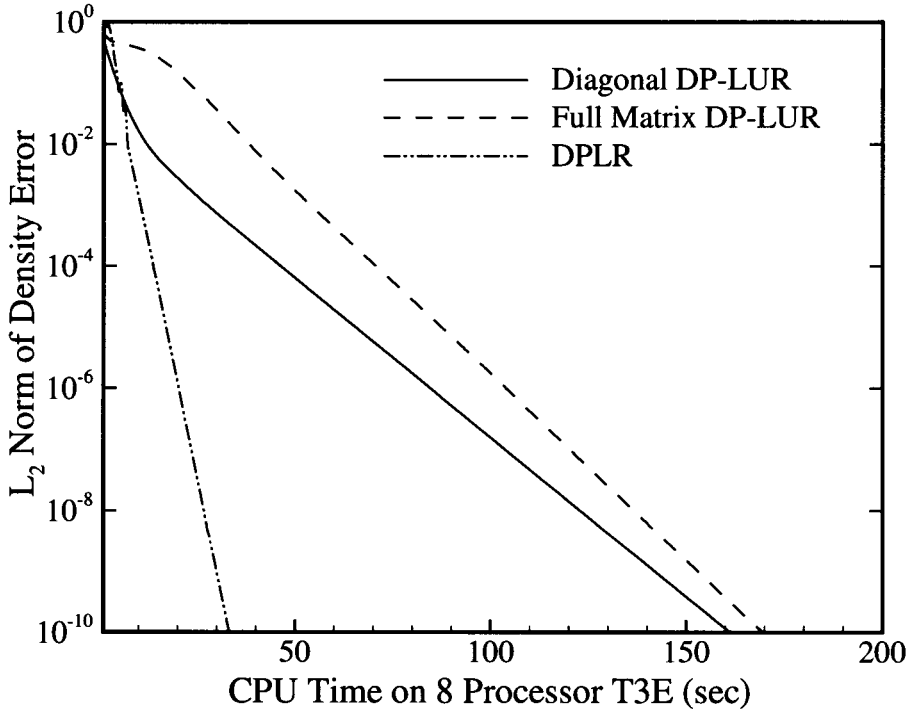


Fig. 6.9 CPU times on an 8 processor Cray T3E for the DPLR method as compared to the two DP-LUR methods. 2-D cylinder-wedge blunt body in perfect air at $M_\infty = 2$ and $Re = 3 \times 10^4$; 128×128 grid with $y_+ = 1$.

shock has reached its final location, the convergence rates of the Mach 2 flows are slower than those for the Mach 15 flows, due to the longer characteristic flow time. Similar results are obtained at other Mach numbers. This shows that the DPLR method can be an effective tool for the solution of both supersonic and hypersonic flows.

Figure (6.9) compares the cost effectiveness of the DPLR method with the full matrix and diagonal DP-LUR methods for a cylinder-wedge flow at Mach 2 and a Reynolds number of 3×10^4. We see that the DPLR method is still clearly the most cost effective approach, requiring only 33 seconds to reach a ten order of magnitude fall in the density error norm, as compared to 160 seconds for the diagonal DP-LUR method, and 170 seconds for the full matrix method. If we compare the times required for the Mach 2 flow shown here with the times required for the Mach 15 flow shown in Fig. (6.4), we see that the time required for the DPLR method is

unaffected by the Mach number. However, the time required by the full matrix and diagonal DP-LUR methods is a strong function of the Mach number, and decreases with decreasing Mach number. This is because the Mach 2 flow is much less non-linear than the Mach 15 flow, and therefore the approximations made in the DP-LUR methods have less effect. It is also interesting that for the Mach 2 flow the diagonal DP-LUR method is actually slightly faster than the full matrix method.

Chapter 7

Parallel Performance of the Methods

7.1 Introduction

This chapter discusses the parallel performance of the new Data-Parallel Lower-Upper Relaxation and Data-Parallel Line Relaxation methods. The performance and parallel efficiency of each method are examined for both a data-parallel version, implemented on a Thinking Machines CM-5 located at the University of Minnesota Army High Performance Computing Research Center (AHPCRC), and a message passing version, implemented on a Cray T3E located at the Minnesota Supercomputer Center (MSC). These machines have been introduced in Chapter 1.

7.2 Description of the Methods

In this section we present tables showing the number of floating point operations required by each portion of the DP-LUR and DPLR methods. These tables are intended to allow for easy comparison between the various implementations of the methods. Floating point operations are calculated for the data-parallel versions only, but the numbers are representative of those obtained for the message-passing versions as well. In these tables arithmetic is counted as a single floating point operation, while functions such as square root and exponentiation are counted as eight. Note that this weighting is not necessarily representative of the number of clock periods required for each operation. For example, on the Cray T3E a floating point multiply or add requires four clock cycles, while a divide requires 60, and a square root or exponentiation can require hundreds of clock cycles.

Table (7.1) presents the floating point operations required for the viscous per-

fect gas algorithms. In this table the explicit portion of the code includes the evaluation of the Euler and viscous fluxes ($\Delta U_{i,j}$), the implicit portion of the code includes those operations necessary to form the Jacobians and compute $\delta U_{i,j}$, and the update portion of the code includes those operations necessary to advance the solution to the next time level. For both the 2-D and 3-D implementations the full matrix DP-LUR method is the most computationally intensive of the three, requiring about 2 times as many floating point operations as the diagonal DP-LUR method and 1.3 times as many as the DPLR method.

	diag DP-LUR		FM DP-LUR		DPLR	
Function	2-D	3-D	2-D	3-D	2-D	3-D
Fluxes	1151	2450	1151	2450	1151	2450
Implicit	726	1780	3152	5427	1861	4106
Update	31	33	31	33	31	33
Totals	1877	4263	4303	7910	3012	6523

Table 7.1 Number of floating point operations required for each portion of the perfect gas implementation of the DP-LUR and DPLR methods.

	diag DP-LUR		FM DP-LUR	
Function	2-D	3-D	2-D	3-D
Fluxes	2622	4846	2622	4846
Implicit	1487	2655	7531	10908
Chemistry	1812	1859	1873	1949
Update	1011	1050	1011	1050
Totals	6932	9510	13037	17853

Table 7.2 Number of floating point operations required for each portion of the reacting air implementation of the diagonal and full matrix DP-LUR methods.

Table (7.2) presents the same information for the reacting air versions of the DP-LUR algorithms. The DPLR method has not been implemented for the reacting air model at this time. The explicit, implicit, and update portions of the code have the same definitions as before. The chemistry routine includes those operations required to evaluate the chemical and vibrational source terms and their derivatives. Note that the update procedure is much more computationally intensive than the perfect gas case, due to the required calculation of the vibrational temperature and

mixture values for the transport coefficients as detailed in Chapter 2.

7.3 Parallel Performance on the CM-5

The DP-LUR and DPLR methods were implemented on the CM-5 in the data-parallel programming style, using CM-Fortran as discussed in Chapter 1. A number of runs were performed to determine how the performance of the method is affected by the size of the problem and the size of the machine. The first thing that was noted was that the performance is strongly dependent on the degree of load balancing. In order to achieve the best performance it is necessary that the number of grid points in each parallel dimension is a multiple of the number of processors. If this criterion is not followed, the compiler will "pad" all of the arrays to ensure proper load balancing in each parallel dimension, resulting in memory usage and CPU time corresponding to the larger padded size. In addition, the CSHIFT intrinsic function is very inefficient in any dimension that is not a multiple of the number of processors, resulting in a larger amount of time spent on communication. The combination of these two effects can result in a 25% or greater degradation of performance. Therefore, since all of the partitions of the CM-5 have a number of processors equal to a power of two, all of the results presented in this section are run using power-of-two sized computational grids.

Figure (7.1) shows the effect of the problem size on the two-dimensional perfect gas implementation of the diagonal DP-LUR method, for a range of power-of-two sized grids. In Fig. (7.1) we plot the performance in Gigaflops (GF) on a 512 processor CM-5 versus the total number of points in the computational grid. We see that the performance of the method is a strong function of the number of grid points, increasing rapidly as the problem size increases. This is primarily due to the vector nature of the machine; larger problem sizes imply a longer vector length. The last point in the figure (32 million grid points) represents the memory limit of the machine for this implementation of the method. From Fig. (7.1), we see that the performance seems to be reaching a maximum of about 19.5 GF just as the memory

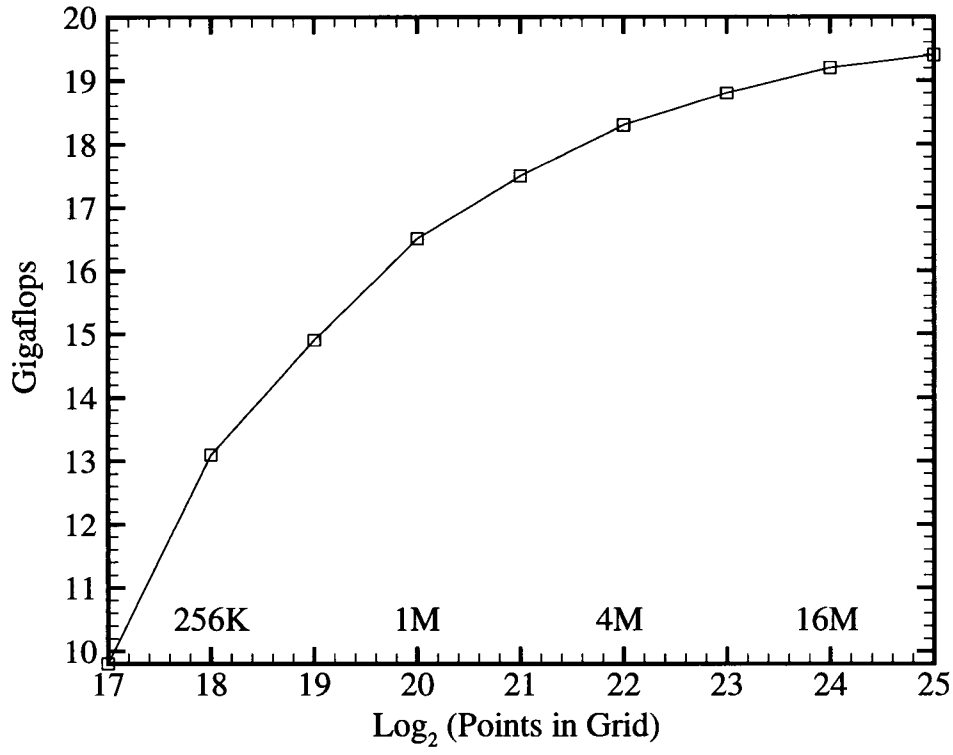


Fig. 7.1 Floating point performance for the inviscid two-dimensional perfect gas implementation of the diagonal DP-LUR method as a function of the number of grid points on a 512 processor CM-5.

limit is reached. This is about 30% of the 64 GF peak theoretical performance of the 512 processor CM-5. The total amount of time spent on communication can be measured for this code, using the performance analysis tools available on the CM-5. For the case shown here, when the maximum number of grid points are used only about 7.5% of the total time is spent on communication.

It is important to note that the peak theoretical performance of the machine is an idealized number, based purely on chip speed assuming infinite vector lengths and no inefficiencies such as page faults or inter-processor communication. Therefore this level of performance can never be obtained in practice. An idea of the operational performance of the CM-5 processors can be obtained by examining the NAS 1 Parallel Benchmark results [Saini and Bailey (1996)]. These benchmarks include a series of pseudo applications designed to simulate the types of activities performed

by actual codes. Table (7.3) summarizes the results of three of the benchmark cases on the CM-5. The first pseudo application tested is the embarrassingly parallel (EP) benchmark, which involves almost no inter-processor communication, and thus provides an upper estimate for the operational machine performance. The other two results shown are for the lower-upper (LU) and block-tridiagonal (BT) benchmarks, which are meant to simulate two typical CFD applications. From Table (7.3) we see that the performance of the EP benchmark is much higher than that of the CFD pseudo applications. The EP benchmark has a per-processor performance of 37.1 Megaflops (MF), which corresponds to 19.0 GF on a 512 processor CM-5. This number is slightly lower than the 19.5 GF obtained for the diagonal DP-LUR method. This shows that the DP-LUR algorithm has a high parallel efficiency on the CM-5, as long as large vector lengths are used.

NAS 1 Benchmark	MFlops per Processor
EP	37.1
LU	2.9
BT	11.9

Table 7.3 Floating point performance for three of the NAS 1 Parallel Benchmark pseudo applications on the CM-5. Data taken from Saini and Bailey (1996).

In Fig. (7.2) we examine the effect of the machine size on the performance of the algorithm. In the figure we plot the performance of the diagonal DP-LUR method in Megaflops per processor versus the number of grid points on each processor. We see that the curves for each of the machines lie essentially on top of each other, and therefore the number of points per processor is the most important performance indicator for the DP-LUR algorithm. The performance for any machine size is a function only of the number of points per processor, and not on the machine size itself. For example, if there are 4096 points on each node of the machine, each processor performs at 34 MF, regardless of the total number of processors used. Thus, the DP-LUR method is perfectly scalable from the smallest to the largest available machines.

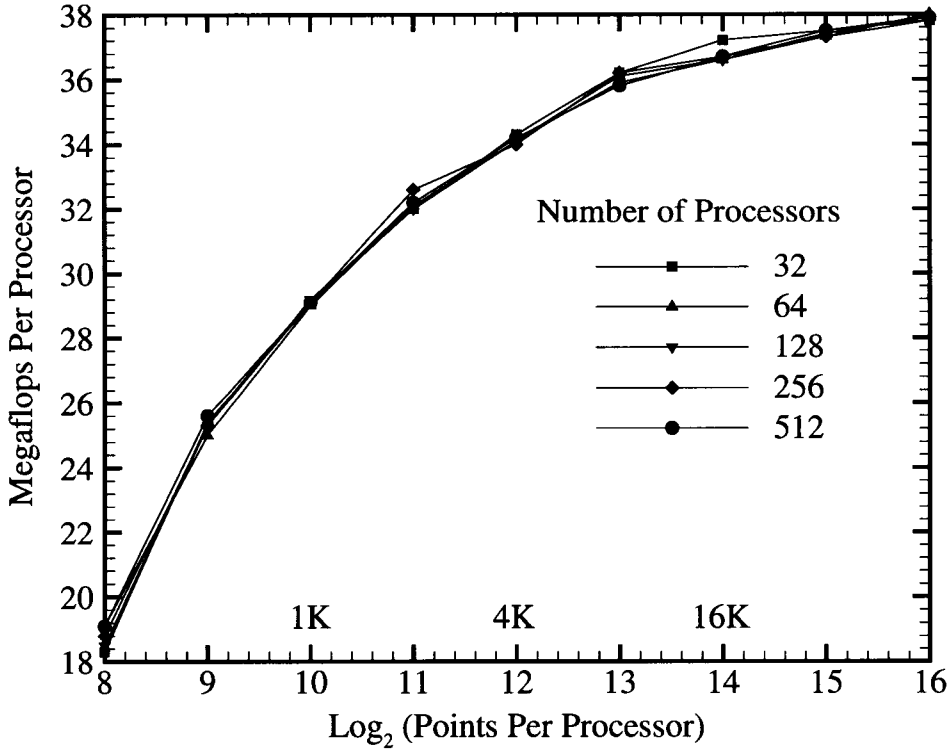


Fig. 7.2 Per-processor floating point performance for the inviscid two-dimensional perfect gas implementation of the diagonal DP-LUR method as a function of the number of grid points per processor for different partitions of the CM-5.

Based on the data in Fig. (7.2), we can calculate a vector half-performance length for this algorithm, which we define as the vector length required to achieve 50% of the peak operational performance of the vector portion of the code. However, on the CM-5 it is impossible to determine whether a given operation is performed as a vector operation. Therefore, since there are no portions of the code which are obviously not vectorizable, we assume that 100% of the parallel computations performed are vectorized. The amount of time spent on inter-processor communication scales linearly with problem size, and thus will not affect the calculation of the half-performance length. Using these assumptions the vector half-performance length for this algorithm is about 64, which is quite large. In addition, with four vector units per processing node, this indicates that each processor must hold 256 grid points to achieve 50% of the peak operational performance.

The floating point performance of each of the routines in the perfect gas and reacting air implementations of the diagonal DP-LUR method is summarized in Table (7.4). Because the performance of the method depends strongly on the number of points per processor, the results in Table (7.4) should only be used to compare the relative efficiencies of the code. For both the perfect gas and reacting air implementations the flux evaluation is the most efficient portion of the code. However, in both cases the implicit portion is also very fast. The perfect gas solution update runs at only 1.7 GF, but this number is misleading, since only 31 floating point operations are involved in this computation.

Function	2-D Perf Gflops	2-D Chem Gflops
Fluxes	20.0	17.6
Implicit	17.9	15.7
Chemistry	–	10.4
Update	1.7	8.5
Total	18.9	14.2

Table 7.4 Floating point performance of the viscous two-dimensional perfect gas and reacting air diagonal DP-LUR methods. Results obtained on a 64 processor CM-5 using a 1024×512 grid and scaled to a 512 processor machine.

In Table (7.4) we see that the performance for the chemically reacting flow is only 14.2 GF, as compared to 18.9 GF for the perfect gas case. Although both the flux and implicit routines are slightly slower for the reacting air code, we see that the primary cause of the slowdown is the evaluation of the chemical source terms and their derivatives, which runs at 10.4 GF, and the solution update procedure, which runs at only 8.5 GF. The poor performance of the chemical source term evaluation is surprising, since this calculation is entirely pointwise, and thus requires no inter-processor communication. However, further analysis shows that the reduced performance of this routine is a direct result of the implementation of the exponentiation function on the CM-5. For the five species air model it is necessary to compute 38 exponentials during the evaluation of the chemical source term. These operations consume almost 50% of the entire time spent on the source term evalu-

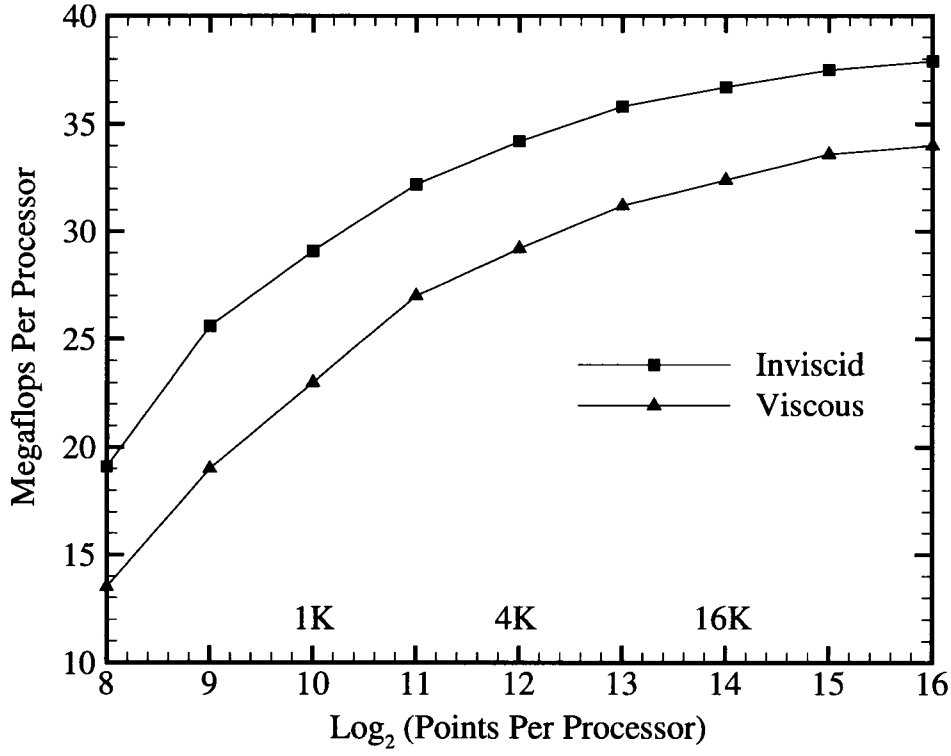


Fig. 7.3 Floating point performance for the inviscid and viscous versions of the two-dimensional perfect gas diagonal DP-LUR method as a function of the number of grid points on each processor.

ation, which requires a total of 1812 floating point operations. The solution update procedure is also slowed by the calculation of the 15 exponentials required to update the vibrational temperature with a Newton iteration, as discussed in Chapter 2.

Figure (7.3) compares the performance of the inviscid and viscous versions of the diagonal DP-LUR method. We see that the performance of the viscous implementation is between 4 and 5 MF per processor lower than the inviscid implementation for all problem sizes. This is due to the increased number of communications required to compute the viscous derivatives in the numerical flux evaluation. The peak performance of the viscous diagonal DP-LUR algorithm is about 34 MF per processor, which corresponds to 17.4 GF on the 512 processor CM-5.

Figure (7.4) presents the performance of the three-dimensional perfect gas implementation of the diagonal DP-LUR method as a function of the number of grid

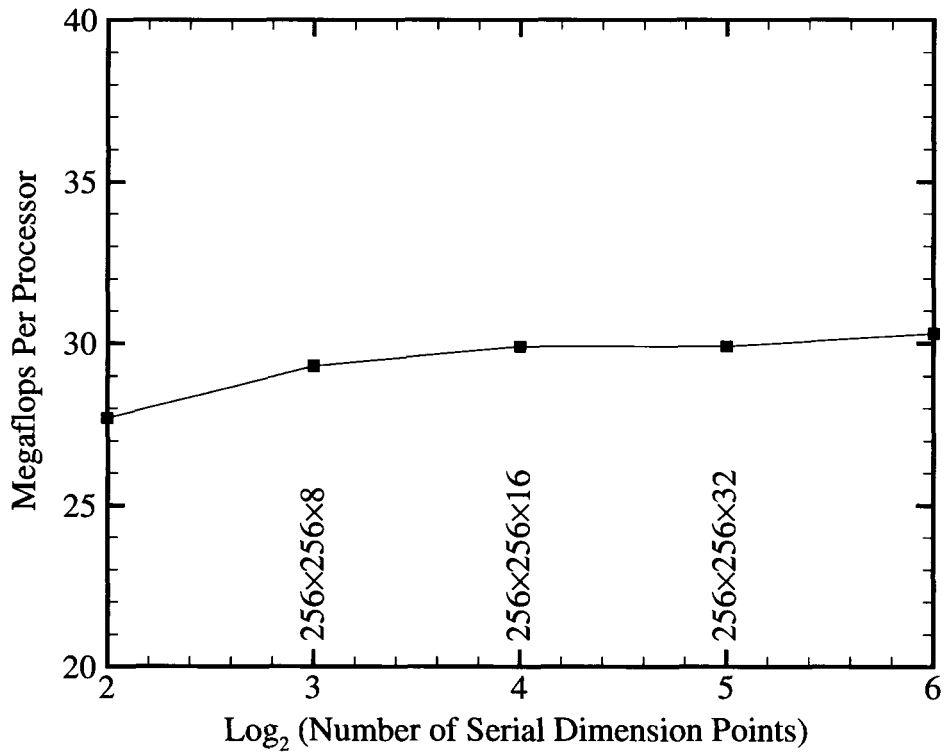


Fig. 7.4 Floating point performance of the three-dimensional perfect gas diagonal DP-LUR method, showing the effect of varying the number of points in the serial (k) dimension, while the number of points in the parallel (ij) dimensions is held constant.

points in the third dimension (k). For various performance reasons, this dimension is treated as a serial dimension in the CM-5 code. This means that for each (i, j) location, all associated k points lie on the same processor. Therefore, no inter-processor communications occur along this axis. In Fig. (7.4) the total number of points in the ij dimensions is held constant, while the number of points in the serial (k) dimension is increased. We see that, for values of k larger than 4, the performance of the 3-D diagonal DP-LUR method is almost constant. This implies that only the number of points in the parallel dimensions, ie. those points that are spread across all of the available processors, should be used when evaluating the vector length. The 3-D algorithm performs significantly better overall than the 2-D algorithm with the same number of points in the parallel dimensions, as shown in Fig. (7.5). This effect is also due to the fact that k is implemented as

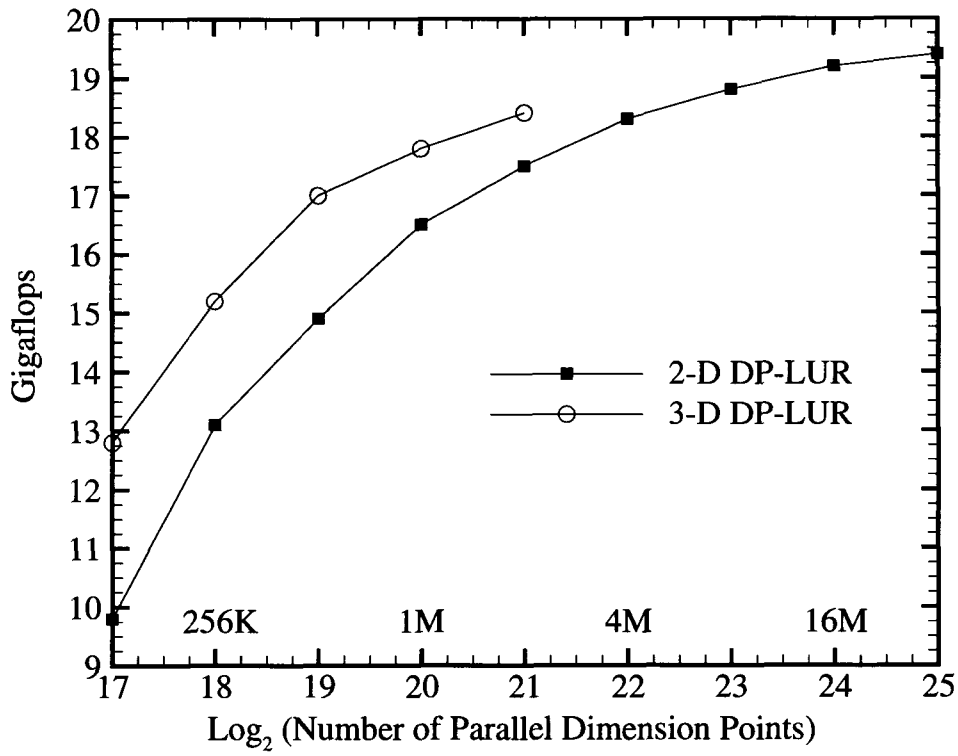


Fig. 7.5 Floating point performance of the two- and three-dimensional perfect gas diagonal DP-LUR methods, as a function of the number of points in the parallel (ij) dimensions. Results obtained on a 512 processor CM5.

a serial dimension, which increases the ratio of the number of computations per communication by about 50%. However, the 3-D version of the algorithm has a lower peak performance than the 2-D version, because of memory limitations. In other words, the available memory is used up before the vector lengths become large enough to achieve a higher peak performance.

The full matrix DP-LUR method retains the perfect scalability of the diagonal method, which is not surprising, since both the diagonal and full matrix methods use the same underlying algorithm to advance the solution in time. However, the full matrix method requires significantly more floating point operations per iteration than the diagonal, but no additional communication. Therefore we expect the full matrix method to have a higher parallel efficiency than the diagonal method. This is indeed the case, as shown in Fig. (7.6), which plots the performance of both

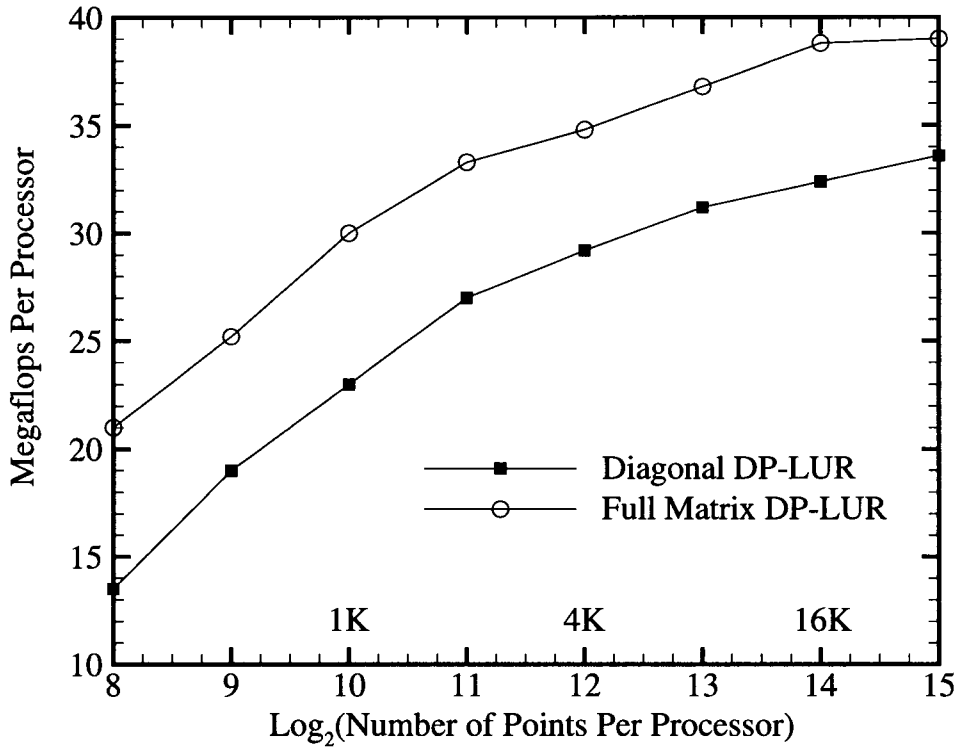


Fig. 7.6 Per-processor floating point performance of the two-dimensional viscous perfect gas full matrix and diagonal DP-LUR methods, as a function of the number of grid points on each processor.

two-dimensional viscous DP-LUR methods as a function of problem size. From the figure we see that the peak performance of the diagonal method is about 34 MF per processor, while the full matrix method achieves a peak performance of 39 MF per processor. On a 512 processor CM-5 these numbers correspond to a peak performance of 17.4 GF for the diagonal method and 20.0 GF for the full matrix method. All other comparisons made for the diagonal DP-LUR method hold for the full matrix method as well.

The data-parallel implementation of the DPLR method should retain the perfect scalability and high parallel efficiency that characterize the DP-LUR methods, since the algorithm design is very similar. However, it is difficult to show this on the CM-5. This is because the CM-5 is a vector parallel machine, and thus large vector lengths are required to ensure good performance. In addition, we have

shown that only the number of points in the dimensions that are spread across all of the processors should be used when evaluating the vector length. In order to solve the required block tri-diagonal system required by the DPLR method without inter-processor communication, it is necessary that all j-points corresponding to a particular i-location be entirely on-processor for the 2-D implementation, while with the DP-LUR methods both the i and j directions can be spread across the available processors. Therefore, when the 128×128 test case presented in the previous chapters is run on a 64 processor CM-5 with four vector units per processor, the vector length will be 64 for the DP-LUR method, but will actually be less than one for the DPLR method. This means that the DPLR method is not properly using the vector hardware on the processors. In fact, with a vector length less than one, some of the vector units on each processor will remain idle during the computation, while others will be processing vectors of length one. Therefore we would expect the two-dimensional DPLR method to be very slow on the CM-5, even though it may have a high parallel efficiency.

Method	Mflops per Processor
Diagonal DP-LUR	13.5
Full Matrix DP-LUR	21.0
DPLR	1.4

Table 7.5 Per-processor floating point performance of the two-dimensional perfect gas viscous DP-LUR and DPLR methods. Results obtained on a 64 processor CM-5 using a 128×128 grid.

This is indeed the case for the implementation tested. Table (7.5) summarizes the per-processor performance of each of the data-parallel relaxation methods for a two-dimensional perfect gas viscous simulation on a 128×128 grid. The performance of the DPLR method is only 1.4 MF per processor, as compared to 13.5 MF for the diagonal DP-LUR method and 21.0 MF for the full matrix method. This would not be a problem on a machine without vector hardware. The 3-D version of the DPLR method can be implemented with both the body-axial and circumferential directions as parallel dimensions, and the body-normal direction as the only serial

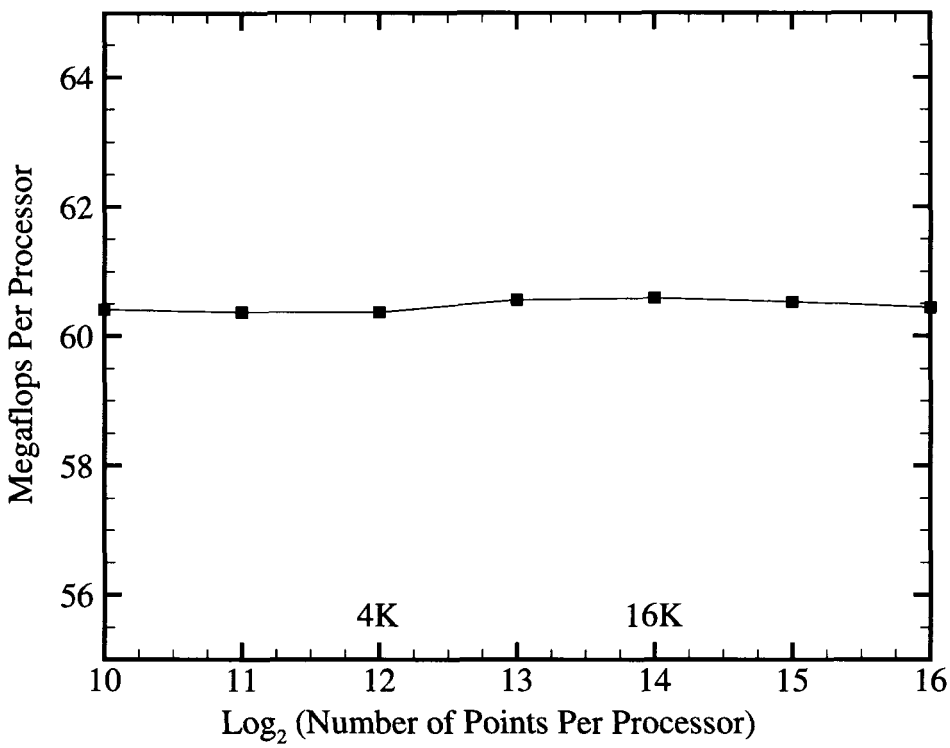


Fig. 7.7 Per-processor floating point performance of the two-dimensional viscous perfect gas full matrix method as a function of the number of grid points on each processor. Calculations performed on an eight processor T3E.

dimension. This would alleviate the problem. However, the 3-D DPLR method uses about four times as much memory as the 3-D DP-LUR method, due to the storage of the seven required Jacobians. Therefore, the problem size that fits in the 32 MB per-processor available memory is severely limited. Thus, while the DPLR method should be efficient on many data-parallel architectures, it is very inefficient on the CM-5.

7.4 Parallel Performance on the T3E

The DP-LUR and DPLR methods were implemented on the T3E in the message-passing programming style, using MPI. Since the processors of this machine do not have vector hardware, we expect the performance of the methods to be much less sensitive to the problem size. This is shown in Fig. (7.7), which plots the per-processor performance of the 2-D perfect gas full matrix DP-LUR method

versus problem size on an 8 processor T3E. We see that in all cases the method runs at about 60.5 MF per processor. The other methods are also unaffected by problem size. In addition, it is possible to run on any number of processors with this machine. Therefore parallel speedup curves can be constructed for each of the data-parallel relaxation methods. The parallel speedup is defined simply by

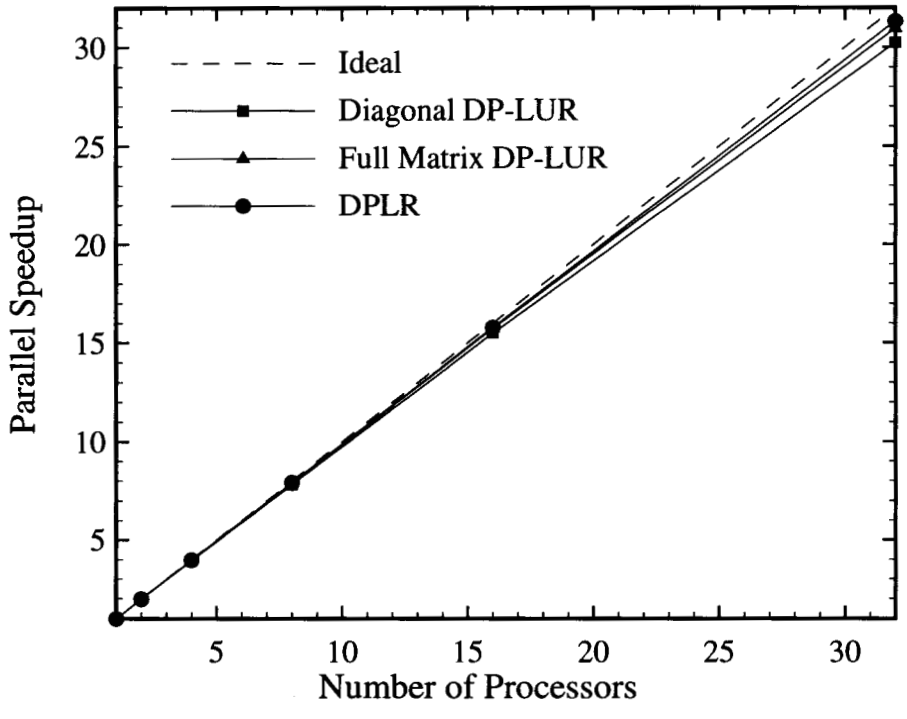
$$S = \frac{t_1}{t_n},$$

where S is the speedup, t_n is the time required on n processors, and t_1 is the time required for the same problem on one processor. We can also define a parallel efficiency as

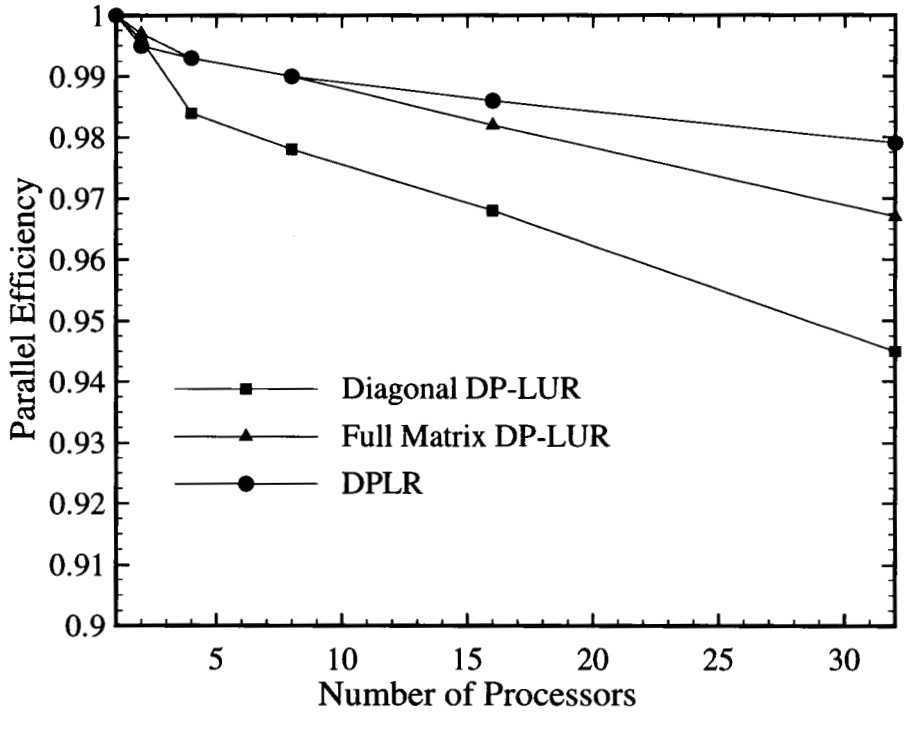
$$E = \frac{S}{n}.$$

The parallel efficiency will always be less than one for a real application, due to the time spent on inter-processor communication.

The speedup curves and parallel efficiencies for the two-dimensional perfect gas viscous implementation of each of the data-parallel relaxation methods are presented in Fig. (7.8). Each of the methods was run using a 512×512 computational grid. From the figure we see that all three of the methods have a very good speedup, up to the maximum number of processors tested. In fact, on 32 processors the speedup for the DPLR method is 31.3, which corresponds to a parallel efficiency of 0.98. This implies that only about 2% of the total time is spent on inter-processor communications. The parallel efficiency of the diagonal DP-LUR method is the lowest of the three, falling to 0.95 on 32 processors. The reason for the lower efficiency of the diagonal DP-LUR method becomes apparent if we examine the implicit portion of each algorithm. Each method requires a communication of the $\delta U_{i,j}$ vector during each relaxation step. The time spent on this communication is partially masked by the computation of each relaxation step with the use of non-blocking sends and receives. However, while the amount of data that must be communicated is constant for each of the methods, the amount of computational



a)



b)

Fig. 7.8 a) Parallel speedups and b) parallel efficiencies, for the data-parallel relaxation methods on the T3E. Two-dimensional perfect gas viscous implementation on a 512×512 grid.

work within each relaxation step is not. Therefore, we expect the diagonal DP-LUR method, which is much less computationally intensive than the full matrix or DPLR methods, to do a worse job of masking the communication time.

The sustained performance on the T3E for both the perfect gas and reacting air 2-D implementations of the DP-LUR and DPLR methods is shown in Table (7.6). We see that for the perfect gas implementation the full matrix code has the highest per-processor performance, of 60.3 MF per node. The diagonal DP-LUR and DPLR methods perform at 53.4 and 52.3 MF per node, respectively. The full matrix method is probably the most efficient due to the fact that it requires the largest number of floating point operations. The performance of the DPLR method is the slowest of the three, due to the increased number of memory loads and stores resulting from the use of the exact flux Jacobians. The three-dimensional algorithms have similar performance numbers. Results are presented here for the viscous implementations of the methods, but the inviscid implementations have almost identical performance numbers. This is because in message-passing the computation of the viscous fluxes requires no additional communication. The performance of the reacting air versions of the DP-LUR method are in each case worse than the perfect gas implementation, again due to the computation of the required exponentials, which can require hundreds of clock periods on the T3E.

Method	2-D PG	2-D Air
Diagonal DP-LUR	53.4	46.7
Full Matrix DP-LUR	60.3	48.1
DPLR	52.3	—

Table 7.6 Sustained performance for the 2-D perfect gas and reacting implementations of the DP-LUR and DPLR methods on the Cray T3E. Results obtained using a 128×128 grid on an 8 processor machine, and tabulated in terms of MF per processor.

The most efficient algorithm in terms of sustained performance on the T3E is the perfect gas version of the full matrix DP-LUR method. However, its peak performance is 60.3 MF per processor, which is only 10% of the peak theoretical

performance of the machine. This number seems quite low, but it is comparable to other published results. The NAS 2 parallel benchmark results offer the best comparison, since these codes have been written to simulate actual CFD applications, and the individual machine vendors have not been allowed to perform assembler level optimizations to the source code. Unfortunately, NAS 2 benchmark results have not yet been published for the T3E. However, results from the T3D for the block tri-diagonal (BT) pseudo application show a performance of about 10 MF per processor [Saphir *et al.* (1996)]. In addition, results for the NAS 1 benchmarks, which have been published for both the T3D and T3E, show that the sustained speed on the T3E is typically about 3.3 times that on the T3D [Saini and Bailey (1996)]. This would result in a performance of about 33 MF per processor on the T3E for the BT benchmark. Therefore, the 60 MF per processor obtained for the full matrix DP-LUR method seems reasonable. However, it is possible that further optimizations can be made to the source code that could increase the performance. These results show that single processor performance is the single most important characteristic of a well designed algorithm for the T3E.

Chapter 8

Inviscid Shock Interactions

8.1 Introduction

The pioneering work in the field of shock interactions was by Edney (1968), who classified six types of interactions based on an experimental study of oblique shock waves impinging on a variety of blunt bodies. These classifications have since become the basis for categorizing all shock interactions. Edney's work was motivated by the practical problem of understanding unexplained transient heating loads during the insertion of models into supersonic wind tunnels. This work led to the realization that shock interactions can cause anomalously high heating and pressure loads on supersonic and hypersonic vehicles. A dramatic example of this effect was the airframe damage sustained by the X-15 due to a shock interaction. The interaction occurred after a ram jet test article was attached to a ventral fin, and resulted in the loss of the ram jet during flight. There was also concern about the interaction between fuselage-generated and wing-generated shocks on preliminary Space Shuttle configurations [Hunt and Creel (1971) & Bertin and Graumann (1973)]. Later, it was discovered that extremely high heating loads were possible at the engine inlet cowl on the proposed National Aerospace Plane (NASP) due to shock interaction heating [Klopfer and Yee (1988)]. Because of these engineering issues, a large number of papers have been published in an attempt to classify and predict shock interaction phenomena.

Recently, Olejniczak and Candler (1995) and Olejniczak *et al.* (1996b) used shock interactions on double-wedge and double-cone geometries to study chemical kinetics models used in reacting hypersonic flows. The experimental results and nu-

merical simulations of these flows raised many questions, and made it clear that the underlying fluid dynamics of these interactions is not well understood. In particular, some of the cases showed unexplained large amplitude steady-state variations in the surface pressure and heat transfer rate. Also, surprisingly small changes in the geometry led to large changes in the overall flow structure. These flows are complicated by the fact that finite-rate chemical reactions are coupled to the fluid motion and that viscous effects play a large role. Therefore, we decided to eliminate some of these complications and investigate inviscid perfect gas flows on double-wedge geometries, with the goal of increasing our understanding of the underlying gas dynamics of the resulting shock interactions. By applying the CFD methods developed in this research to these flows, we are able to systematically explore the effects of the governing parameters on the flowfield at a level of resolution not previously possible.

Edney's classification system is based on an externally generated oblique shock impinging on the bow shock generated by a blunt body. The location of the impingement point determines the type of shock interaction that occurs. With a double-wedge geometry, shown schematically in Fig. (8.1), the first wedge generates the impinging oblique shock, and the second wedge generates either a second oblique shock or a curved bow shock. The resulting interactions between these shock waves can be classified according to Edney's scheme. Four of the six interactions that Edney classified appear. However, because of the geometrical constraints imposed by the double-wedge, there are differences between these interactions and those seen by Edney. In addition, a previously unidentified shock interaction is seen. The type of interaction that occurs depends on the relevant non-dimensional parameters, which for inviscid flow are the Mach number M, the ratio of specific heats γ, the ratio of wedge face lengths L_1/L_2, and the two wedge angles θ_1 and θ_2.

The double-wedge shock interactions are described in order of occurrence as the second wedge angle is increased. An emphasis is placed on the detailed structure of

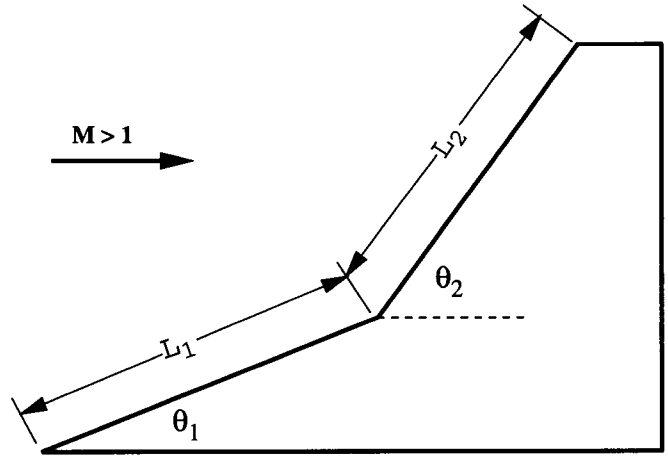


Fig. 8.1 Schematic diagram of a typical double-wedge geometry.

the interactions and the transition criteria between the various types. The discussion is split into two sections, considering high and low Mach number flows separately. This is done because for a given first wedge angle and ratio of specific heats, there is a critical freestream Mach number which determines the fundamental behavior of the flow downstream of the interaction point and, therefore, the basic nature of the interaction. This concept is described in detail below. In this work we have chosen to hold the first wedge angle, the ratio of wedge face lengths, and the ratio of specific heats constant so that we can fully explore the remaining two-dimensional M-θ_2 parameter space.

8.2 Numerical Method

The two-dimensional compressible Euler equations are solved using modified second-order Steger-Warming flux vector splitting [MacCormack and Candler (1989)]. Solutions were also obtained for some of the cases using the Harten-Yee upwind non-MUSCL TVD scheme discussed earlier [Yee 1989)]. Although the TVD method required fewer total grid points to obtain a grid-converged solution, the method converged much slower, resulting in an increased computational cost. The solution was advanced in time toward steady-state using either the diagonal or full matrix DP-LUR method, implemented on the Thinking Machines CM-5.

It is critical to resolve the shock interaction regions in these double-wedge flows. Although these flows are inviscid, the grid spacing introduces an artificial length scale that determines the thickness and minimum size of the shock waves. Therefore, it is important that the grid is sufficiently fine so that the implied length scale is smaller than any relevant physical length scale, such as the length of a transmitted shock. The structure of the grid, particularly around the triple points, can alter the angles of the shock waves and contact surfaces. We have taken care to design grids with enough points in the proper locations to give the correct results. Based on the results of a grid resolution study [Olejniczak *et al.* (1996a)] grid sizes of 512×512 points were determined to be sufficiently large to resolve all of the relevant features in these flowfields. For some of the interactions, smaller grids can predict the wrong type of interaction because of incorrect shock locations or shock angles. We decided to use grid sizes of 1024×1024 for all of the computations shown here, ensuring that even the smallest structures are resolved.

Each of the computations presented in this chapter required between 5000 and 15000 iterations to reach a steady-state solution, which required between one and four hours of CPU time on a 64 processor CM-5. In contrast, on a single processor SGI R8000 workstation, each of these calculations would have required on the order of 70 to 280 hours. This graphically demonstrates the usefulness of these new methods for the solution of large-scale problems.

8.3 Shock Polar Diagrams

It is useful to present shock interaction flows using shock polar diagrams, in which the pressure jump across a shock wave is plotted versus the flow deflection angle. Following the convention used by Edney, we present shock polars in which positive angles correspond to clockwise deflections from the horizontal. The path of the interaction is shown in bold.

Briefly, the shock polar represents the locus of all flow states that can be

obtained by passing through a shock of the given Mach number. The entire region behind a planar shock wave is then represented by a single point on a shock polar diagram. The maximum angle that a flow can be turned by an oblique shock is easily seen; this is the point of maximum deflection which separates the polar into the weak and strong regions. Just below this point on the shock polar is the sonic point. Above the sonic point lie solutions which produce subsonic flow behind the shock, and below this point lie solutions which produce supersonic flow. While the shock polar representation can be used to predict certain types of shock interactions, it is only an approximate representation of the flow when there are curved shock waves. In this situation, the shock polar is correct only in the vanishingly small region about the shock intersection point.

8.4 High Mach Number Interactions

In this research we have restricted ourselves to $\gamma = 1.4$, $L_1/L_2 = 1$, and $\theta_1 = 15°$, which allows us to fully explore the two-dimensional M-θ_2 parameter space. The shock interactions are presented in order of occurrence as the second wedge angle increases. The discussion is split into two sections considering interactions above and below a critical Mach number which determines the behavior of the flow downstream of the interaction region. The definition of this critical Mach number and the distinction between these two regimes will be clarified in the following discussion.

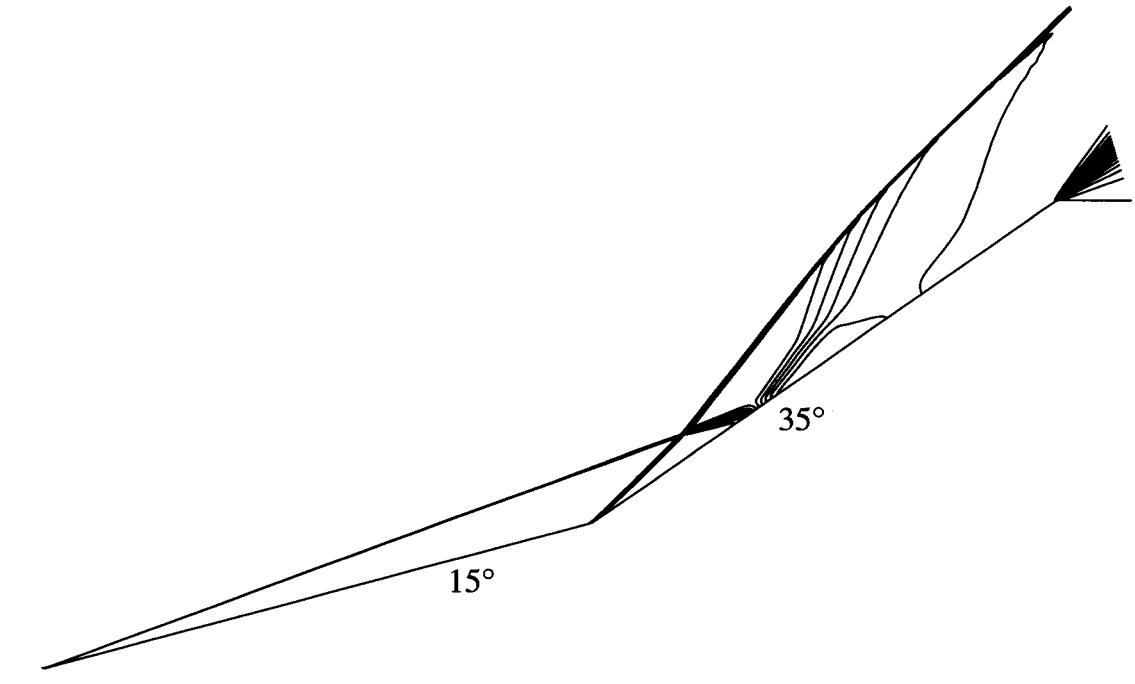
In this section we present results for inviscid double-wedge flows at Mach 9. These results are characteristic of shock interactions that occur above the critical Mach number. After presenting the Mach 9 results, we discuss the effect of varying the Mach number on the type of interaction that occurs, and summarize the transition criteria between the various interactions.

8.4.1 Type VI interactions

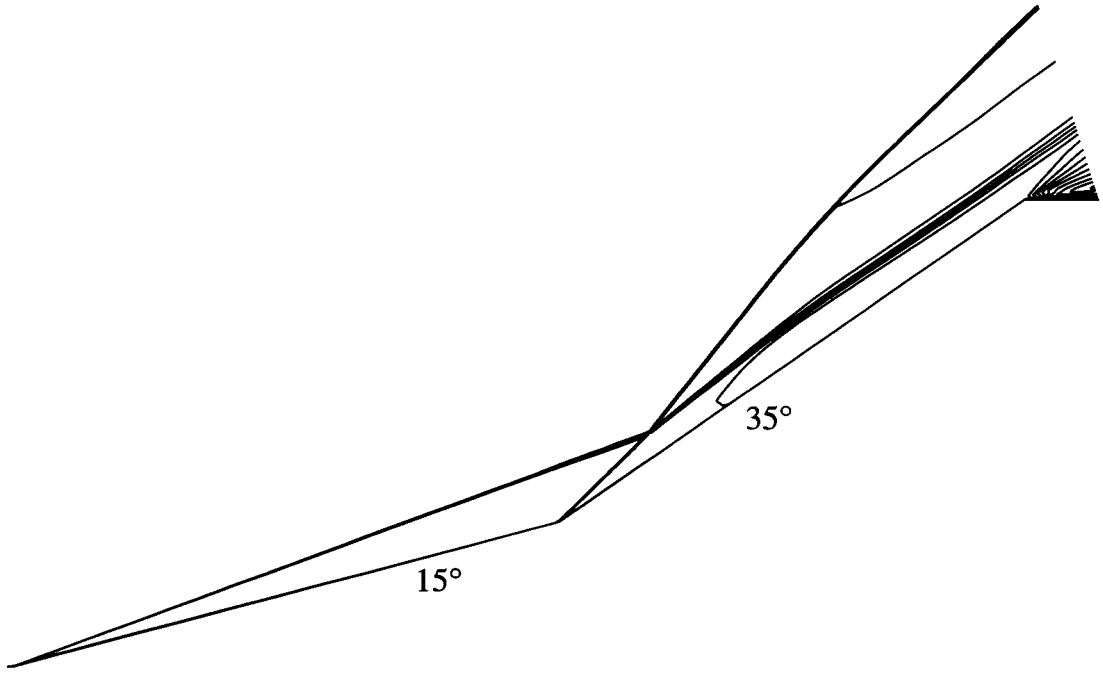
Figure (8.2) shows the computed pressure and Mach contours for a flow at

$M = 9$ with wedge angles of $\theta_1 = 15°$ and $\theta_2 = 35°$. This purely supersonic flow corresponds to an Edney Type VI shock interaction. The attached oblique shock from the nose intersects a second oblique shock from the corner between the wedges. In this discussion, we refer to the fluid which has passed though the nose shock as the inboard flow, while the fluid which passes only through the shock emitted by the second wedge is called the outboard flow. A contact discontinuity is emitted from the interaction point, which separates the inboard flow that has passed through both oblique shocks from the outboard flow. This contact discontinuity is clearly visible in Fig. (8.2b). However, since the contact discontinuity cannot support a pressure gradient, it is not directly visible in the pressure contour plot (Fig. 8.2a). An expansion fan is also emitted from the intersection point, which strikes and reflects off the surface of the second wedge. The reflected expansion waves are deflected as they pass through the contact discontinuity, and eventually intersect with and weaken the oblique shock, as seen in Fig. (8.2a). The waves are also partially reflected from the contact discontinuity as compression waves, and slightly re-compress the fluid along the surface. This can be seen from the closed pressure contour on the second wedge in Fig. (8.2a). This is a standard interaction that occurs when two oblique shocks of the same family intersect. However, the geometrical constraints imposed by the second wedge surface create the reflected expansion and compression waves which complicate the structure of the flow.

A schematic drawing of a typical Type VI interaction is shown in Fig. (8.3). The numbered regions correspond to the numbered points in the shock polar diagram for this interaction, shown in Fig. (8.4). Since the flow angle in Region (1) must be equal to the first wedge angle, Point (1) is on the $M = 9$ freestream shock polar at $\theta = -15°$. This fixes the pressure ratio and the Mach number in Region (1) ($M_1 = 5.04$). The shock polar for Region (1) can now be drawn originating from Point (1). Likewise, the flow in Region (2) must be parallel to the second wedge, and therefore Point (2) is located on the $M = 5.04$ polar at $\theta = -35°$. The location



a)



b)

Fig. 8.2 a) Pressure and b) Mach number contours for a Type VI interaction with $\theta_1 = 15°$, $\theta_2 = 35°$, $M = 9$, and $\gamma = 1.4$.

of Points (3) and (4) can be determined from compressible gas-dynamics: Point (4) must lie on the $M = 9$ polar; Point (3) must be at the same location as Point (4) because Regions (3) and (4) are separated by a contact discontinuity; and Points (2) and (3) must be connected by a Prandtl-Meyer expansion since the flow in Region (3) is underexpanded with respect to the flow in Region (4). Therefore, Points (3) and (4) lie on the intersection of the Mach 9 polar with the isentropic expansion path originating at Point (2). In this case the numerical results can be checked analytically, and the computed shock angles and pressures agree with this analysis.

For the Type VI shock interaction discussed here, Points (3) and (4) lie below the sonic point of the freestream polar. As the second wedge angle increases, Point (2) moves up and to the left on the shock polar diagram, until eventually the isentropic expansion path originating from Point (2) intersects the freestream polar above the sonic point. At this point a purely supersonic Type VI interaction can no longer exist. This is the transition criterion between a Type VI and a Type V interaction. For the case discussed here this criterion predicts transition at a second wedge angle of 39.76°. This value is 4.4° less than the maximum deflection angle for the freestream flow, which is 44.16°. Previously, Bertin and Hinkle (1975) conducted a set of double-wedge experiments with a gap between the wedges in order to bleed the boundary layer and prevent separation. Their experimental results showed Type VI-Type V transition within 2° of the maximum deflection angle. However, it is possible that the gap between the wedges or the viscous effects altered the transition point from that predicted by characteristic theory.

Our results show that the transition process actually occurs continuously over a small range of second wedge angles, and not instantaneously at a discrete value of θ_2 as experiments seem to indicate. This transition mechanism is necessary as it is impossible for the three shock Type VI configuration to jump directly to the seven shock Type V configuration. The transition from a Type VI to a Type V interaction is detailed in Fig. (8.5), which shows the pressure and Mach contours

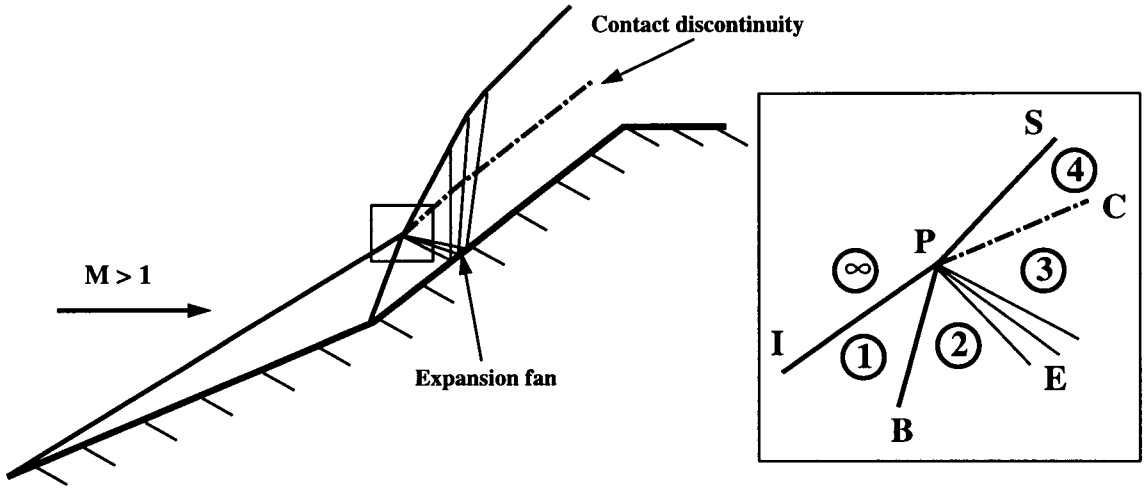


Fig. 8.3 Schematic diagram of a Type VI shock interaction with a zoom of the interaction region.

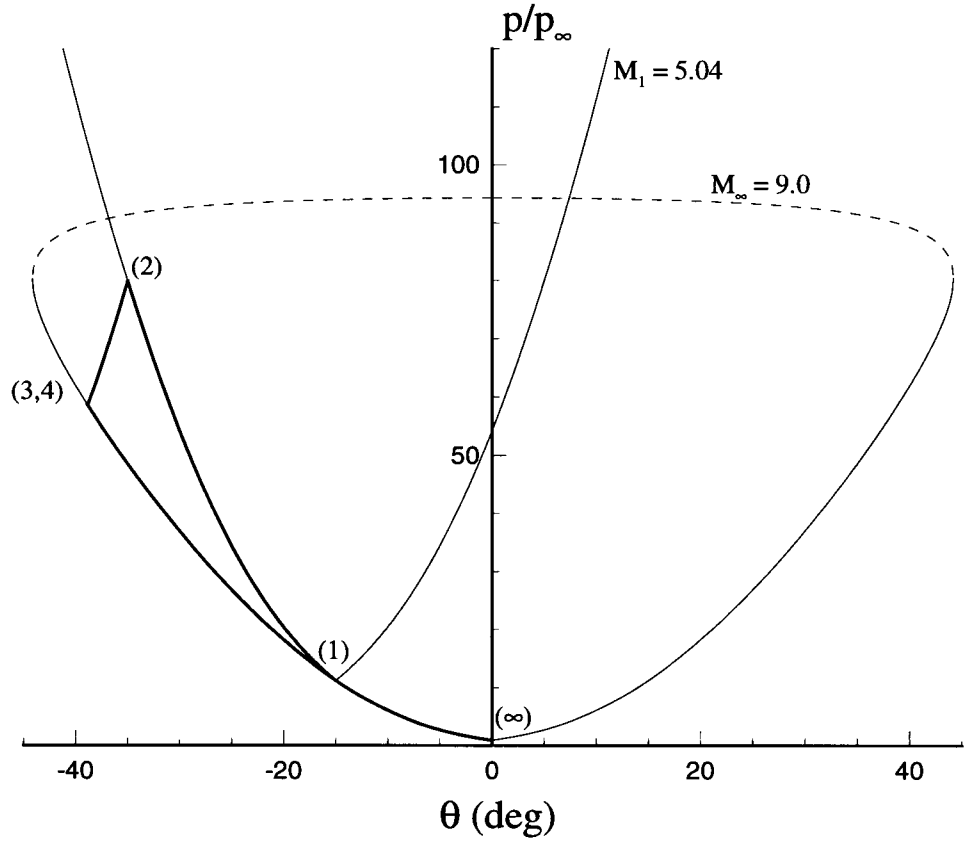
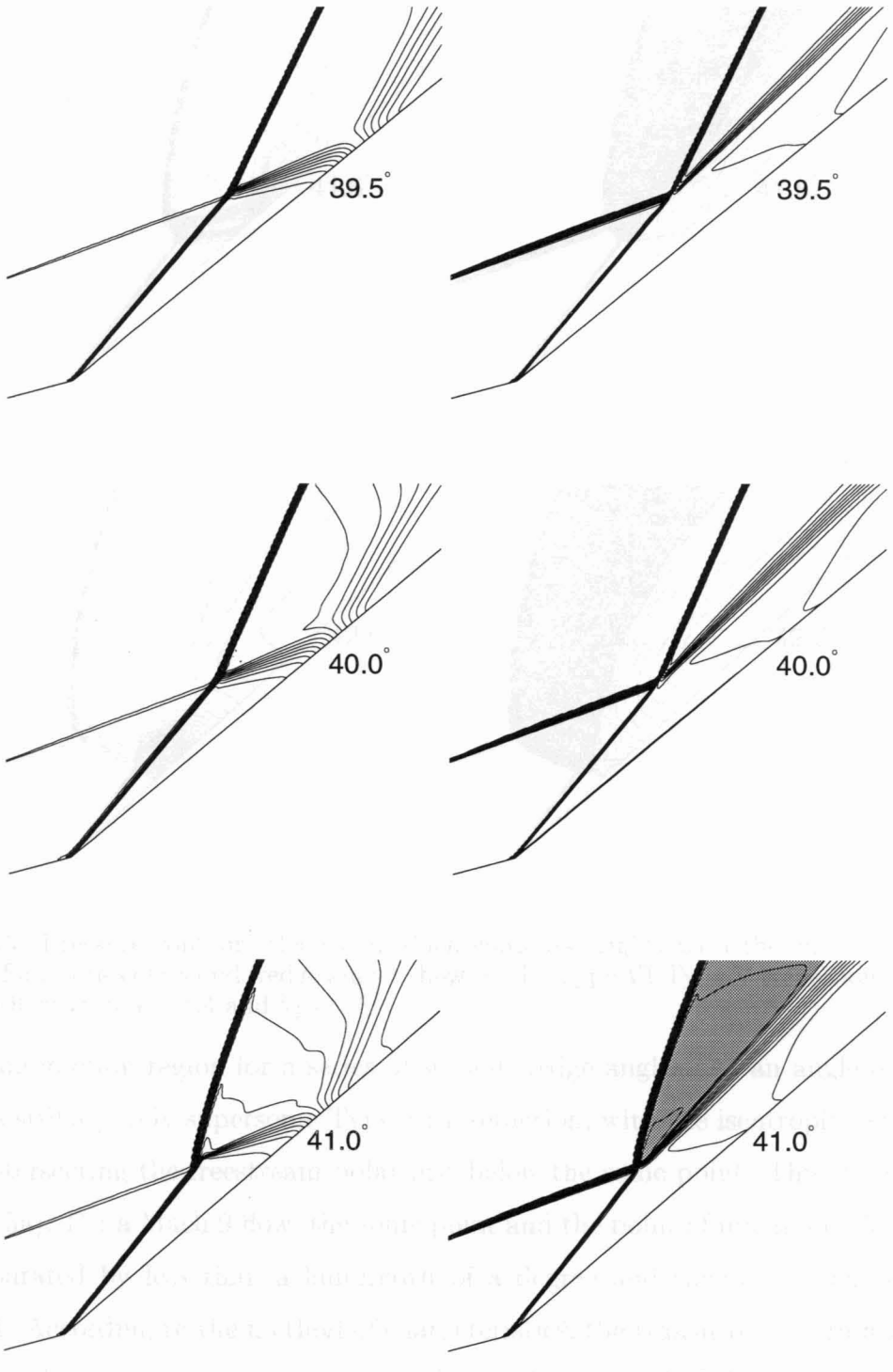
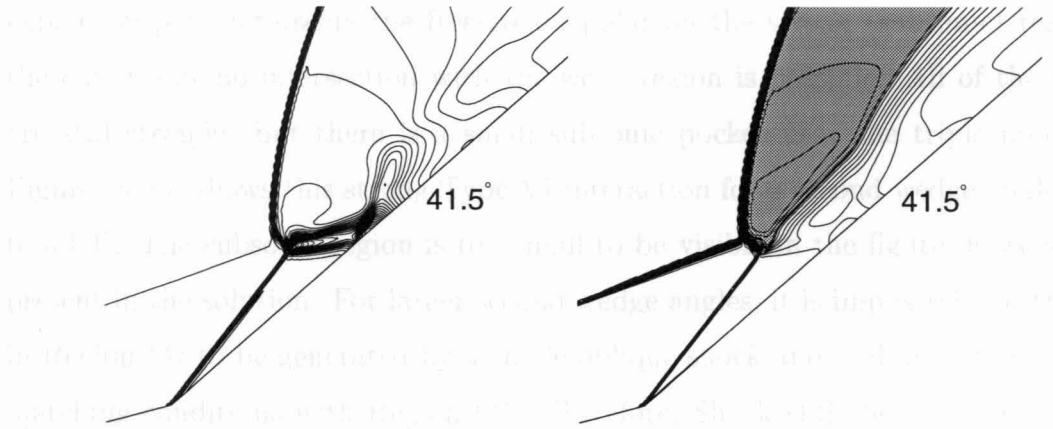
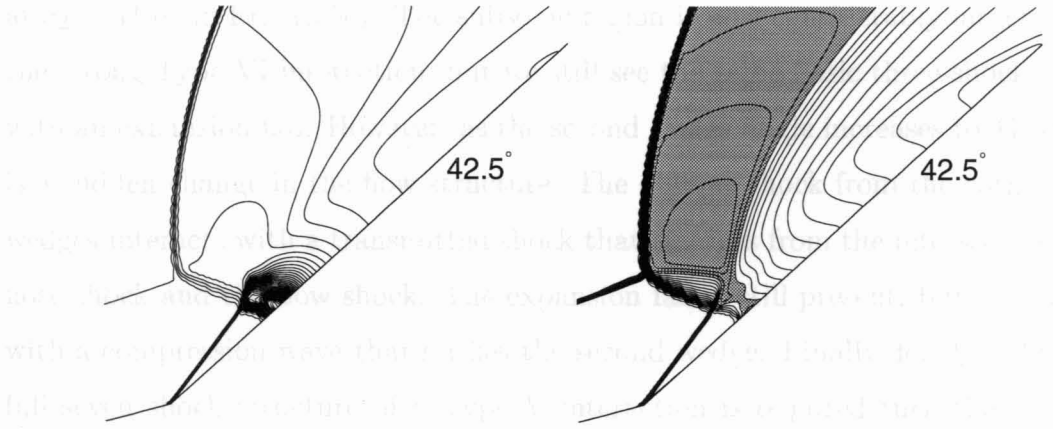


Fig. 8.4 Shock polar diagram for a Type VI shock interaction with $\theta_1 = 15°$, $\theta_2 = 35°$, $M = 9$, and $\gamma = 1.4$.


39.5°
39.5°
a)
40.0°
40.0°
b)
41.0°
41.0°
c)



d)



e)

Fig. 8.5 Pressure contours (left) and Mach contours (right) with the subsonic region shaded for a series of second wedge angles showing the Type VI-Type V transition process. Mach 9 flow with $\gamma = 1.4$ and $\theta_1 = 15°$.

in the interaction region for a series of second wedge angles. At an angle of 39.5°, there is still a purely supersonic Type VI interaction, with the isentropic expansion path intersecting the freestream polar just below the sonic point. This is shown in Fig. (8.5a). For a Mach 9 flow, the sonic point and the point of maximum deflection are separated by less than a hundredth of a degree and thus are essentially co-located. According to the method of characteristics, the transition criterion defined above predicts transition at a second wedge angle of $\theta_2 = 39.76°$. Above this angle a purely supersonic Type VI interaction is no longer possible. However, for second wedge angles up to 40.0° strong Type VI interactions exist, where the isentropic

expansion path intersects the freestream polar on the strong (subsonic) region of the curve, but no intersection with the weak region is possible. All of the shocks are still straight, but there is a small subsonic pocket near the triple point (P). Figure (8.5b) shows this strong Type VI interaction for a second wedge angle equal to 40.0°. The subsonic region is too small to be visible in the figure, however it is present in the solution. For larger second wedge angles, it is impossible for the flow in Region (4) to be generated by a single oblique shock and still meet the required matching conditions with Region (3). Therefore, Shock (PS) becomes curved and there is an intermediate interaction that we call a supercritical Type VI. This is seen at $\theta_2 = 41.0°$ in Fig. (8.5c). The subsonic region is now considerably larger than in the strong Type VI interaction, but we still see the same basic three-shock pattern with an expansion fan. However, as the second wedge angle increases to 41.5° there is a sudden change in the flow structure. The oblique shock from the corner of the wedges interacts with a transmitted shock that emerges from the intersection of the nose shock and the bow shock. The expansion fan is still present, but it is merged with a compression wave that strikes the second wedge. Finally, for $\theta_2 = 42.5°$ the full seven shock structure of a Type V interaction is required turn the flow; this Type V interaction is discussed in detail in the next section. It is interesting to note that the maximum surface pressure jumps from 110 times freestream for the Type VI interaction to 730 times freestream for the Type V interaction within a range of second wedge angles that is less than 3°.

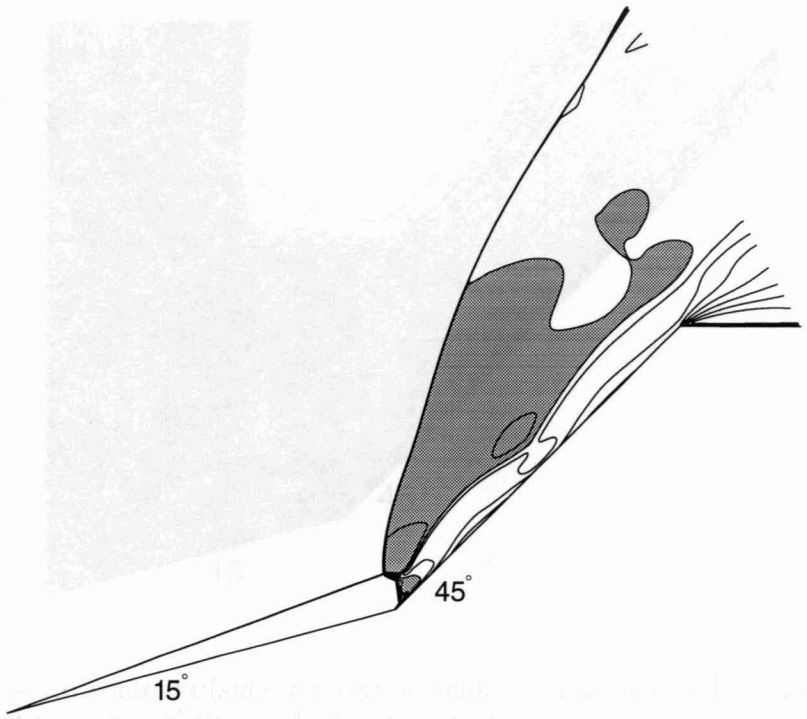
8.4.2 Type V Interactions

The computed Mach contours for a Type V interaction with $\theta_2 = 45°$ are shown in Fig. (8.6). The flow structure is best explained by considering the schematic drawing shown in Fig.(8.7). In the following discussion we relate the features shown in the schematic to the solution shown in Fig. (8.6). Because Shock (PS) is curved and the flow in Region (3) is non-uniform, Shock (PT) must also be curved. The curvature of this shock gives rise to a region of weak flow gradients originating

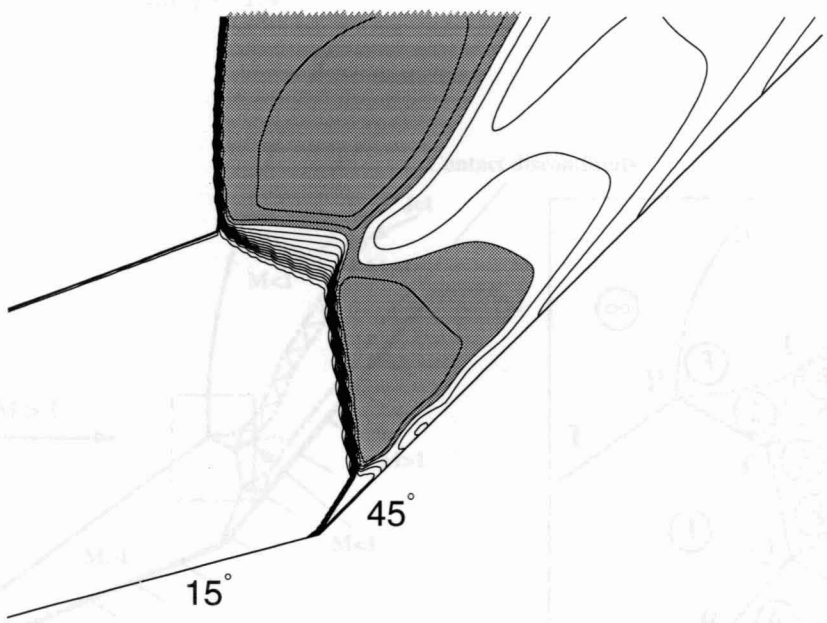
behind the shock, bounded by the two contact discontinuities (C_1 and C_2) emitted from (P) and (T). This forms what appears in numerical solutions and experiments as a thick contact surface, which is termed a jet by Edney. However, in Fig. (8.6c), a plot of the entropy in the shock interaction region, it is possible to distinguish (C_1) and (C_2) from the regions of weak gradients (Regions 2 and 4) which they enclose. The streamlines which pass through the triple points show the location of the contact discontinuities.

The flow in Region (4) must match pressure and angle with the flow which passes through Shock (TQ), necessitating Shock (TU) which separates Regions (2) and (4). This shock can also be seen in Fig. (8.6). This shock was foreseen by Edney and is shown on his shock polar, but was in fact never indicated in his schematic diagram. The inboard flow remains supersonic as it is turned by Shock (QB) at the corner of the wedges and passes through Shock (QR). In this case, Shock (QR) undergoes a regular reflection at the wedge surface, although it can also undergo a Mach reflection for some conditions, as seen in Fig.(8.5e).

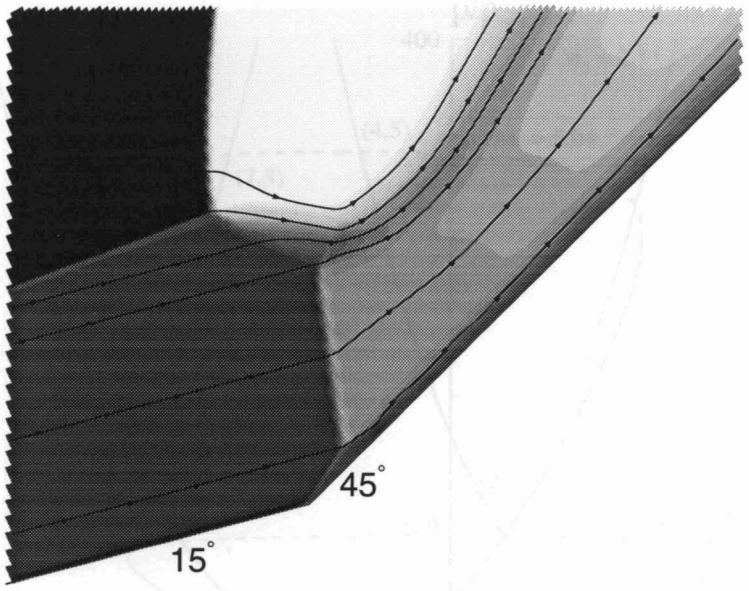
Figure (8.8) shows the shock polar for the Type V interaction shown in Fig. (8.6), with numbered regions corresponding to Fig. (8.7). Points (1) and (6) are located by the requirement that the flow angles match the first and second wedge angles, respectively. Points (2) and (3) must be co-located on the intersection of the freestream and Region (1) polars. Points (4) and (5) must be co-located on the intersection of the Region (1) and Region (2) polars. Finally, Points (7) and (8) must be co-located on the intersection of the Region (1) and Region (6) polars. The predictive capability of the shock polar can be judged by comparing the analytical pressure ratios with the computed surface pressure, shown in Fig. (8.9). The first peak in surface pressure, at 150.5 times freestream, is due to the oblique shock at the corner and should be equal to the pressure ratio in Region (6). Point (6) on the shock polar diagram is at a value of 149.2, and the thus the computation agrees well with theory. The largest peak in computed surface pressure, at a pressure



a)



b)



c)

Fig. 8.6 a) Mach contours (subsonic region shaded) for a Type V interaction, b) zoom of the interaction region showing Mach contours, and c) zoom of the interaction region showing streamlines with flooded entropy contours. The wedge angles are $\theta_1 = 15°$ and $\theta_2 = 45°$ with $M = 9$ and $\gamma = 1.4$.

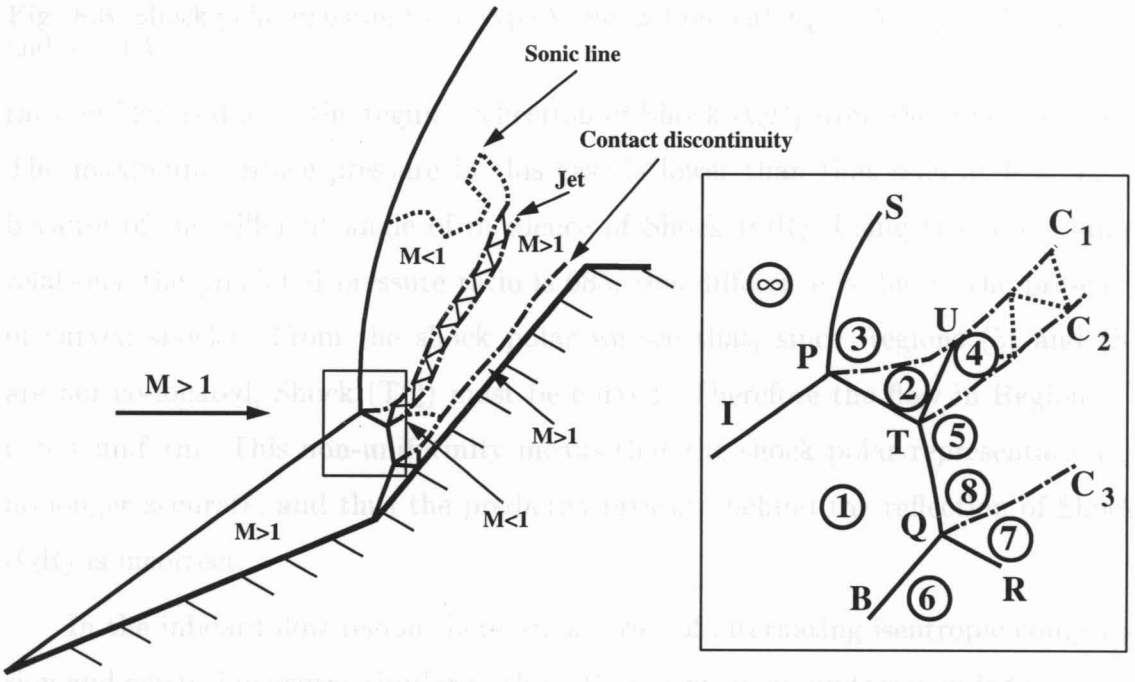


Fig. 8.7 Schematic diagram of a Type V shock interaction with a zoom of the interaction region.

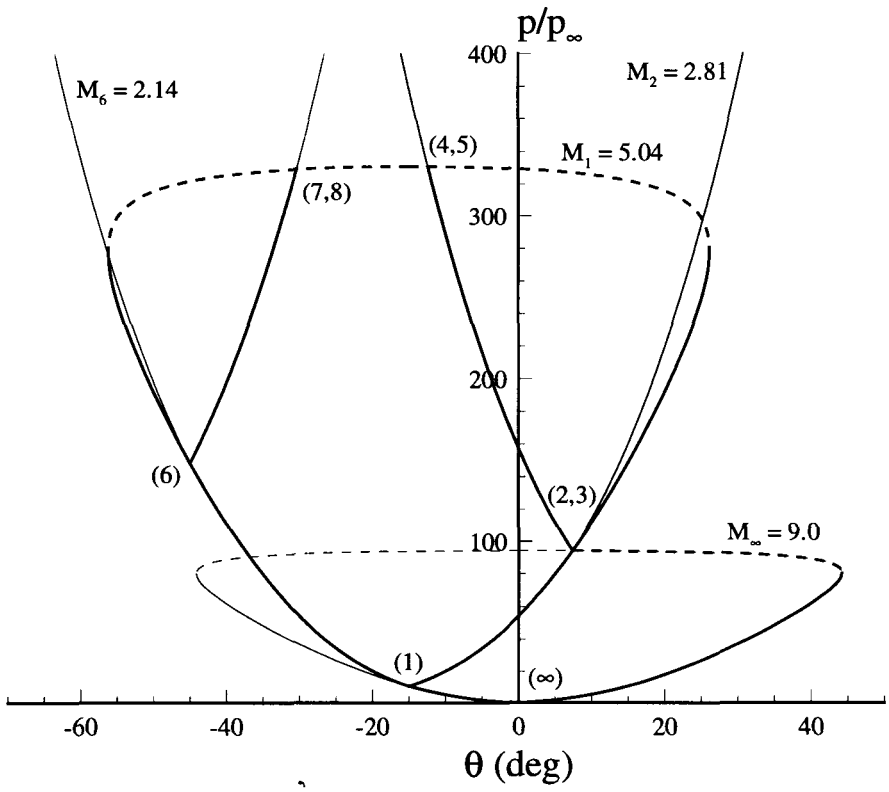


Fig. 8.8 Shock polar diagram for a Type V interaction with $\theta_1 = 15°$, $\theta_2 = 45°$, $M = 9$, and $\gamma = 1.4$.

ratio of 520, is due to the regular reflection of Shock (QR) from the wedge surface. The maximum surface pressure in this case is lower than that seen at $\theta = 42.5°$ because of the different angle of incidence of Shock (QR). Using the shock jump relations, the predicted pressure ratio is 685; this difference is due to the presence of curved shocks. From the shock polar we see that, since Regions (5) and (8) are not co-located, Shock (TQ) must be curved. Therefore the flow in Region (7) is not uniform. This non-uniformity means that the shock polar representation is no longer accurate, and thus the predicted pressure behind the reflection of Shock (QR) is incorrect.

In the inboard flow region there are a series of alternating isentropic compression and expansion waves, similar to those that occur in an underexpanded jet. The appearance of an underexpanded jet impinging on the body is generally associated

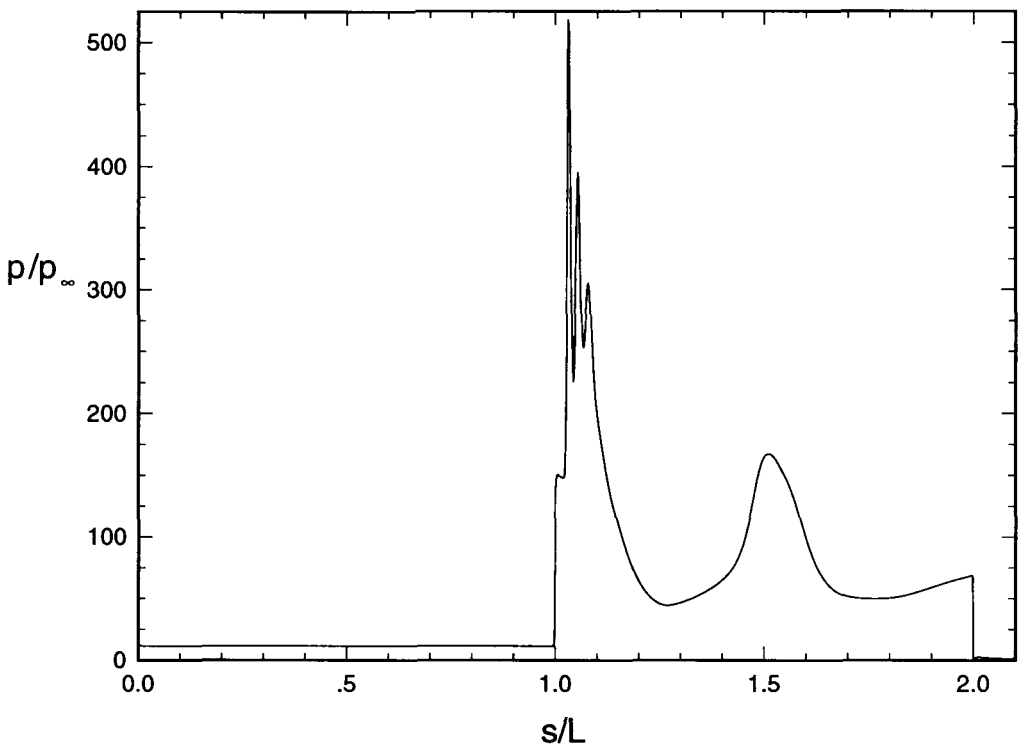


Fig. 8.9 Surface pressure for a Type V interaction with $\theta_1 = 15°$, $\theta_2 = 45°$, $M = 9$, and $\gamma = 1.4$.

with a Type IV interaction and is explained in detail in the next section. However, these results show that the jet is actually associated with the fact that the inboard flow downstream of the interaction is underexpanded with respect to the outboard flow in the adjacent subsonic region. In this case two underexpanded jets are formed. The first jet forms behind the reflection of Shock (QR) from the wedge surface, and is responsible for the two sharp pressure peaks after the maximum in Fig. (8.9). The close spacing of these peaks demonstrates the need for highly resolved grids to simulate this interaction. The second jet forms after the fluid in Regions (5) and (8) re-accelerates to supersonic speed and is then underexpanded with respect to the outboard flow. This jet absorbs the much smaller one described above, and produces the final pressure maximum seen in Fig. (8.9). Mach waves from this underexpanded jet are partially transmitted as acoustic waves through the contact surface into the subsonic region. These acoustic waves produce the inflection points in the bow

shock and the unusual shape of the sonic line that can be seen in Fig. (8.7a).

As the second wedge angle increases, it eventually becomes impossible to match the imposed pressure condition with an attached oblique shock on the second wedge. At this point, Shock (QB) disappears, and Shock (TQ) impinges directly on the first wedge. This marks the transition to a Type IV shock interaction. The second wedge angle at which this transition occurs cannot be computed analytically due to the curved shocks and subsonic flow regions. Numerical computations indicate that the flow transitions to a Type IV interaction at a second wedge angle that is much smaller than the relative maximum flow deflection angle for Region (1), due to the imposed pressure condition which must be met by the inboard flow. For these conditions the largest second wedge angle which produces a Type V interaction is 45.25°, while the relative maximum flow deflection angle is 56.18°. Again, this is a different transition criterion than observed by Bertin and Hinkle (1975), who reported transition for second wedge angles within 2° of the maximum deflection angle for the flow adjacent to the first wedge. Their result is not surprising as the gap between the wedges in their experiment will alter the Type V-Type IV transition process.

8.4.3 Type IV Interactions

Figure (8.10) is a plot of pressure contours for a Type IV interaction with $\theta_2 = 50°$. The corresponding surface pressure is shown in Fig. (8.11). Large amplitude variations in the steady-state pressure are clearly visible along the surface of the second wedge. The interaction is so close to the corner that the compression due to the normal shock and the compression due to the corner result in a nearly constant pressure in the corner region. The peak pressure occurs at the corner and is about 360 times the freestream value, which is lower than the peak pressure for the Type V interaction discussed earlier. After the pressure peak at the corner, alternating pressure minima and maxima are visible along the second wedge surface.

Figure (8.12) shows the Mach contours and the streamlines with flooded entropy contours in the region of the triple point. A sinuous contact discontinuity originates from the triple point, as visualized from the value of the entropy and the streamlines. The reason for the shape of the contact discontinuity is explained as follows. Referring to the schematic diagram (Fig. 8.13), the fluid passing through the bow shock (PS) just above the triple point is deflected towards the body, compressing the supersonic fluid in Regions (2) and (4). This fluid forms a supersonic jet which is underexpanded with respect to the adjacent subsonic fluid in Regions (3) and (5). This jet strikes the second wedge and is trapped against it, undergoing a series of alternate compressions and expansions which produce the waviness of the contact discontinuity and the pressure variations on the wedge surface. This is analogous to the classic problem of an underexpanded jet exhausting into stagnant air. However, there are fundamental differences which produce the unique characteristics of these double-wedge flows. The jet is trapped against the wedge surface for much of its extent, and the imposed pressure condition at the edge of the jet is not constant, resulting in curved Mach waves in the jet and a series of smooth, rounded variations. Additionally, the flow in Regions (3) and (5) surrounding the jet is not stagnant, and thus non-uniform transmission of acoustic waves into this subsonic flow is required to satisfy the flow tangency condition along the contact surfaces. These results are similar to those of Lamont and Hunt (1980) who studied underexpanded jets impinging on inclined plates, and Hains and Keyes (1972) who looked at a Type IV shock interaction between an externally generated shock and the oblique shock from a single wedge. They also observed steady variations in the surface pressure.

The shock polar for a Type IV interaction with $\theta_1 = 15°$ and $\theta_2 = 50°$ is shown in Fig. (8.14). This shock polar is much simpler than the Type V polar. Effectively, Regions (6), (7), and (8) from the Type V interaction disappear and the contact surface (C_3) is replaced by the surface of the double-wedge. Region (5) is bounded from

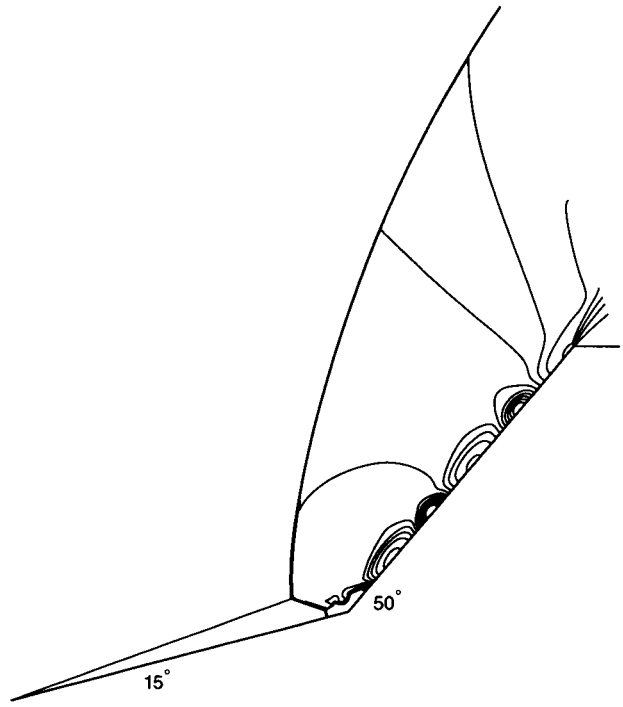


Fig. 8.10 Pressure contours for a Type IV interaction with $\theta_1 = 15°$, $\theta_2 = 50°$, $M = 9$, and $\gamma = 1.4$.

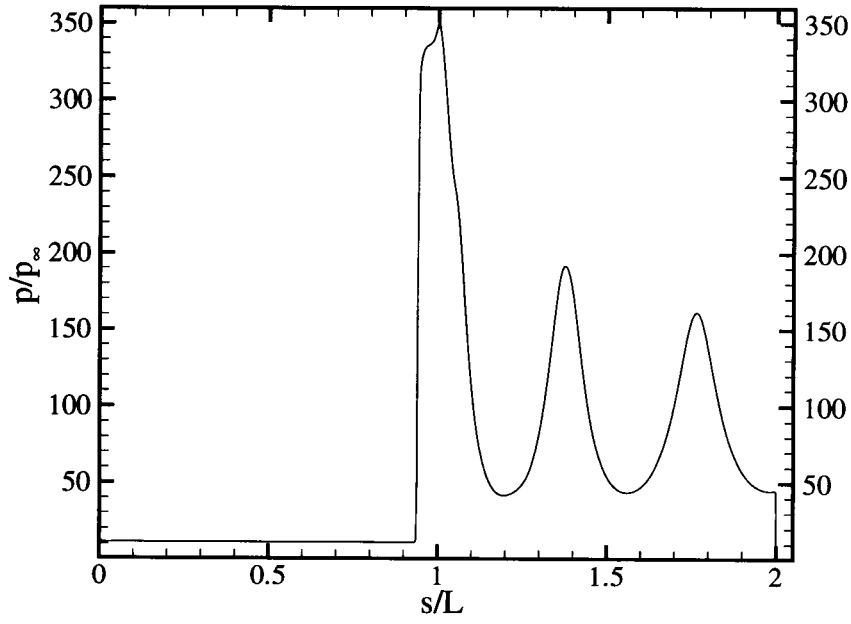
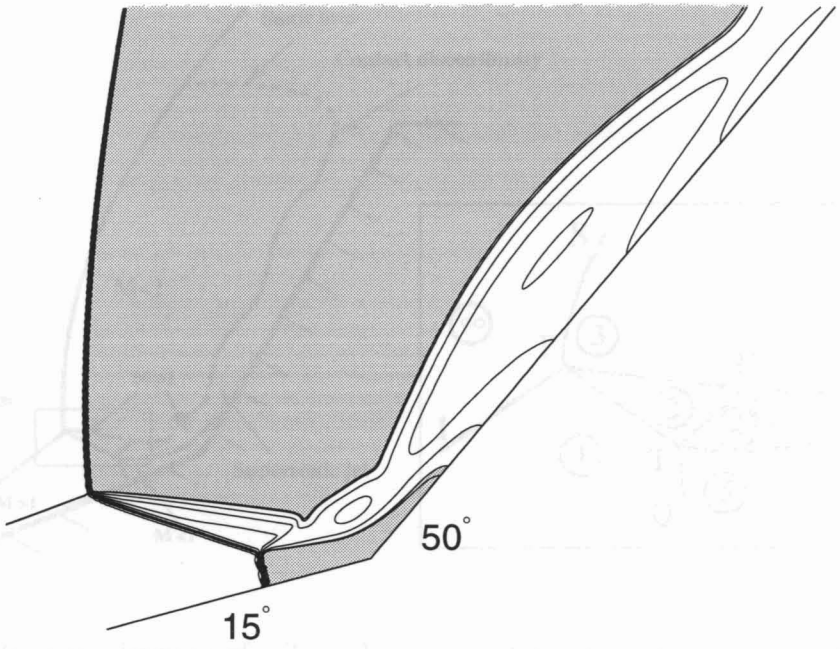
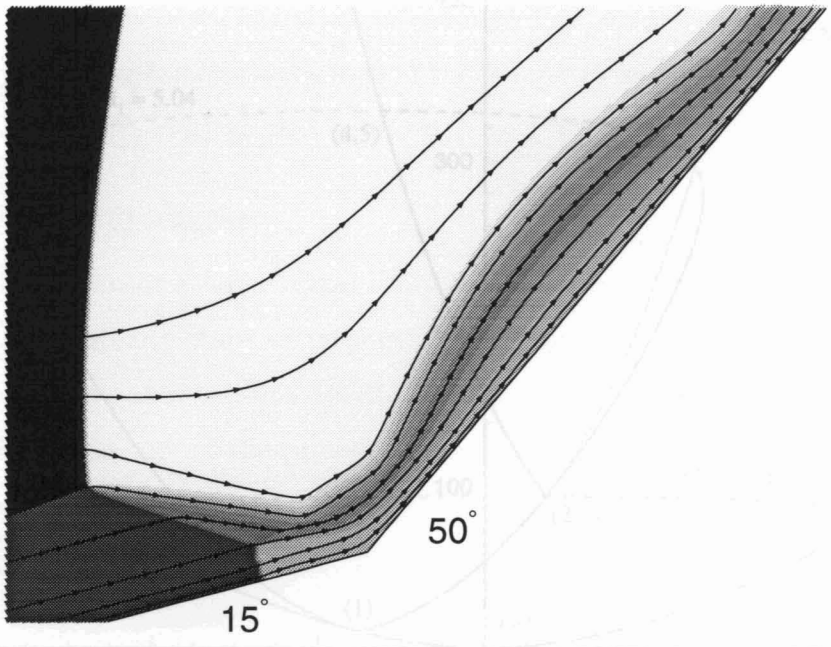


Fig. 8.11 Surface pressure for a Type IV interaction with $\theta_1 = 15°$, $\theta_2 = 50°$, $M = 9$, and $\gamma = 1.4$.



a)



b)

Fig. 8.12 Zoom of the interaction region for a Type IV interaction showing a) Mach contours (subsonic region shaded), and b) streamlines with flooded entropy contours. The wedge angles are $\theta_1 = 15°$ and $\theta_2 = 50°$ with $M = 9$, and $\gamma = 1.4$.

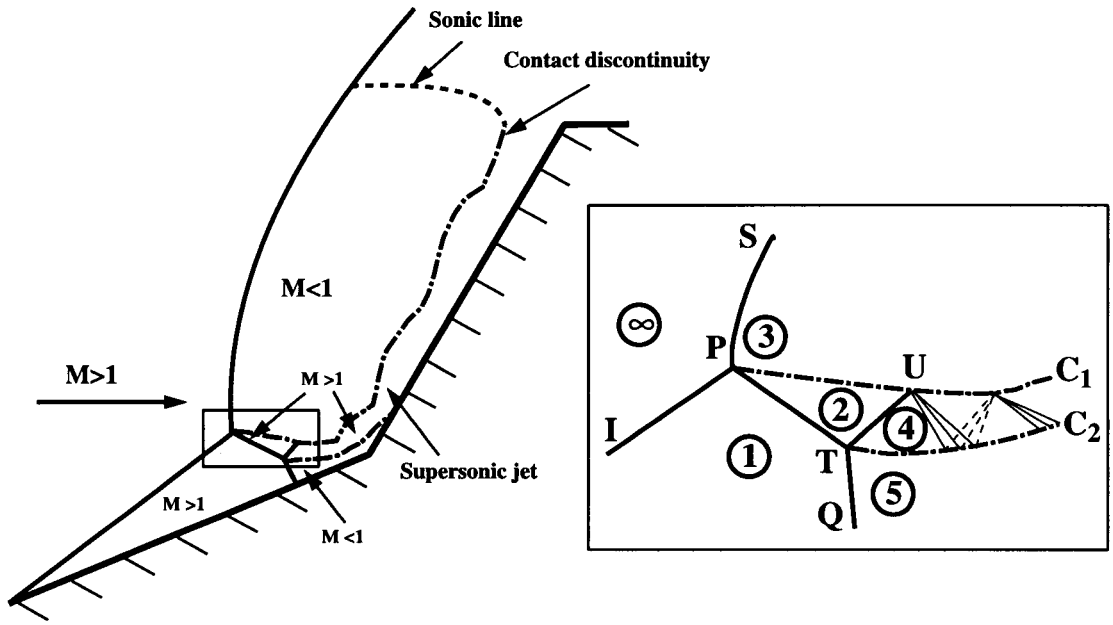


Fig. 8.13 Schematic diagram of a Type IV shock interaction with a zoom of the interaction region.

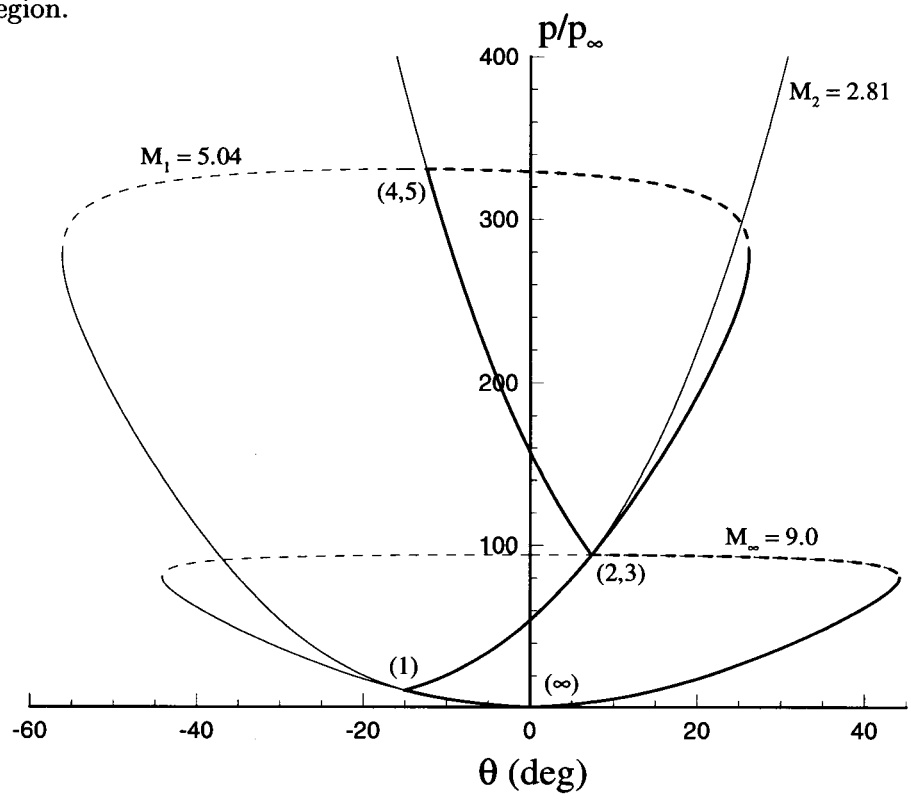


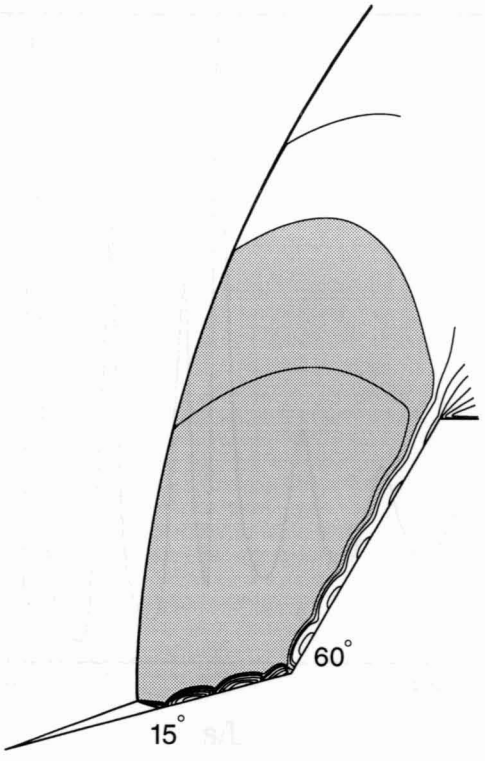
Fig. 8.14 Shock polar diagram for a Type IV interaction with $\theta_1 = 15°$, $\theta_2 = 50°$, $M = 9$, and $\gamma = 1.4$.

below by the first wedge surface, and therefore Shock (TQ) must be curved to match the flow tangency conditions at the contact surface (C_2) and on the wedge surface. These shock curvature effects have a major impact on the flowfield, as discussed in the previous section, and cannot be predicted by the shock polar diagram.

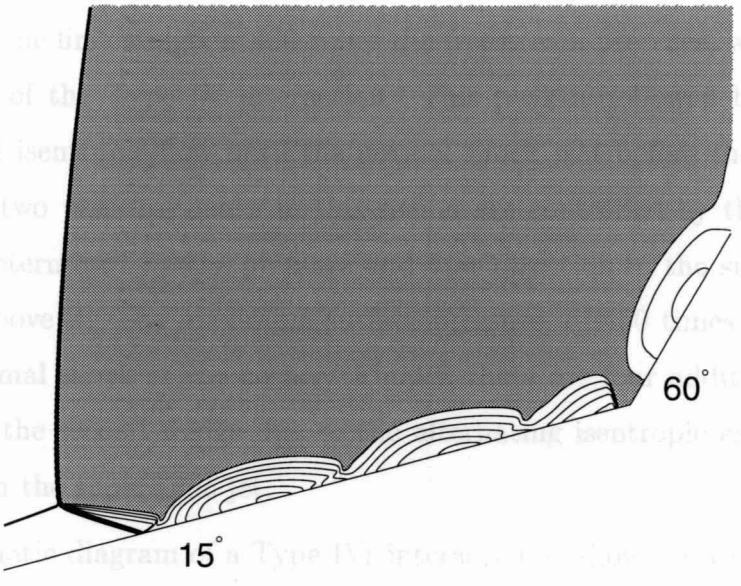
As the second wedge angle increases further, the triple point moves towards the nose of the first wedge and the flow just above the triple point in Region (3) is deflected less towards the body as the oblique nose shock impinges nearer to the point of zero curvature on the bow shock. The Mach stem (TQ) grows shorter, until a point is reached at which a regular reflection rather than a Mach reflection occurs. Region (5) in the Type IV interaction then disappears and the flow in Region (4) is determined by the flow tangency condition on the first wedge. This is the transition criterion between a Type IV interaction and a new type of shock interaction that does not fit into Edney's classification scheme. In fact, this type of interaction cannot occur for the types of blunt body flows Edney investigated, as it is only possible for specific orientations of the geometry surface with respect to the interaction point. We refer to this interaction as a Type IVr.

8.4.4 Type IVr Interactions

The Mach contours for a Type IVr interaction with $\theta_2 = 60°$ are shown in Fig. (8.15). The steady-state variations in Mach number and surface pressure extend from the interaction point to the end of the second wedge, and are more pronounced than in any of the previous cases. As can be seen in Fig. (8.15b), the inboard flow is entirely supersonic after the interaction region. However, a normal shock occurs just before the corner because the angle that the flow must turn is greater the maximum deflection angle. The flow downstream of the corner quickly re-accelerates to supersonic speed. The interaction region is similar to that shown in Fig. (8.12) for the Type IV interaction, but the fluid passing through the oblique shocks is more compressed. This can be seen in Fig. (8.16), which shows the surface pressure for this case. The first peak, due to the regular reflection of the transmitted



a)



b)

Fig. 8.15 a) Mach contours (subsonic region shaded) for a Type IVr interaction, b) zoom of the interaction region. The wedge angles are $\theta_1 = 15°$ and $\theta_2 = 45°$ with $M = 9$ and $\gamma = 1.4$.

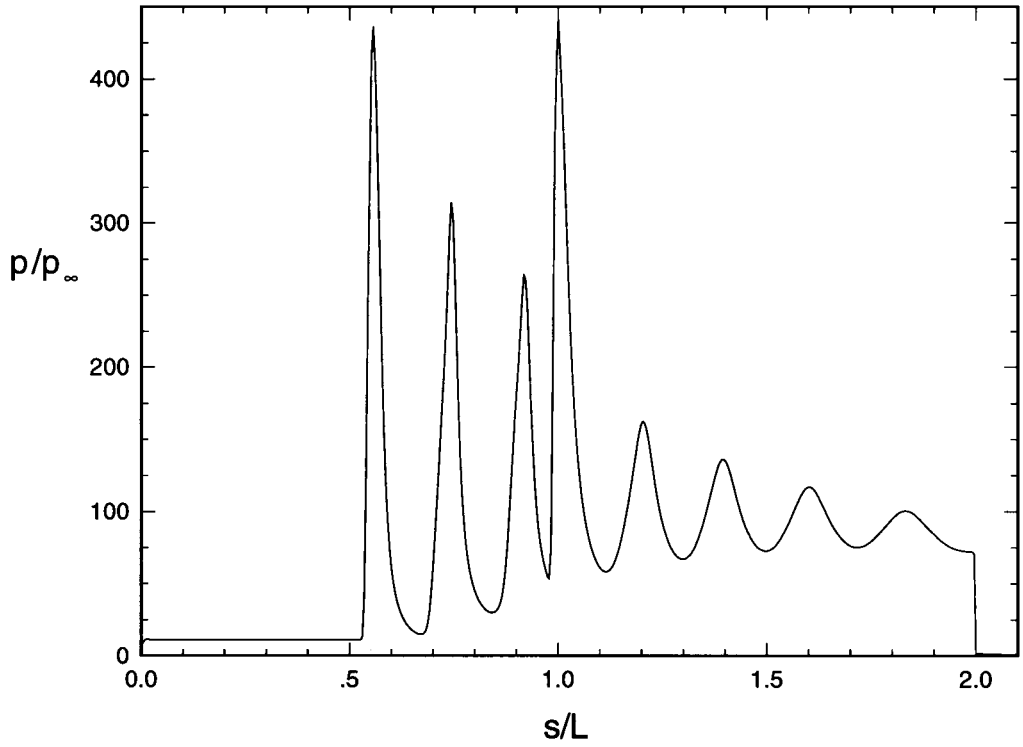


Fig. 8.16 Surface pressure for a Type IVr interaction with $\theta_1 = 15°$, $\theta_2 = 60°$, $M = 9$, and $\gamma = 1.4$.

shock (PT) off the first wedge, is 430 times the freestream pressure, which is higher than the peak of the Type IV interaction. This peak is followed by a region of underexpanded isentropic flow until the normal shock just before the corner. The heights of the two pressure peaks in this region are controlled by the area of the jet, which is determined by the pressure and flow direction in the subsonic region immediately above it. The maximum surface pressure, of 450 times freestream, is due to the normal shock at the corner. Finally, there are four additional pressure maxima along the second wedge due to the alternating isentropic expansions and compressions in the supersonic jet.

The schematic diagram of a Type IVr interaction is shown in Fig. (8.17). The numbered regions correspond to the points on the shock polar diagram (Fig. 8.18) with $\theta_2 = 60°$. The only difference between this shock polar and the Type IV polar is the location of Point (4). In the Type IV interaction, the location of

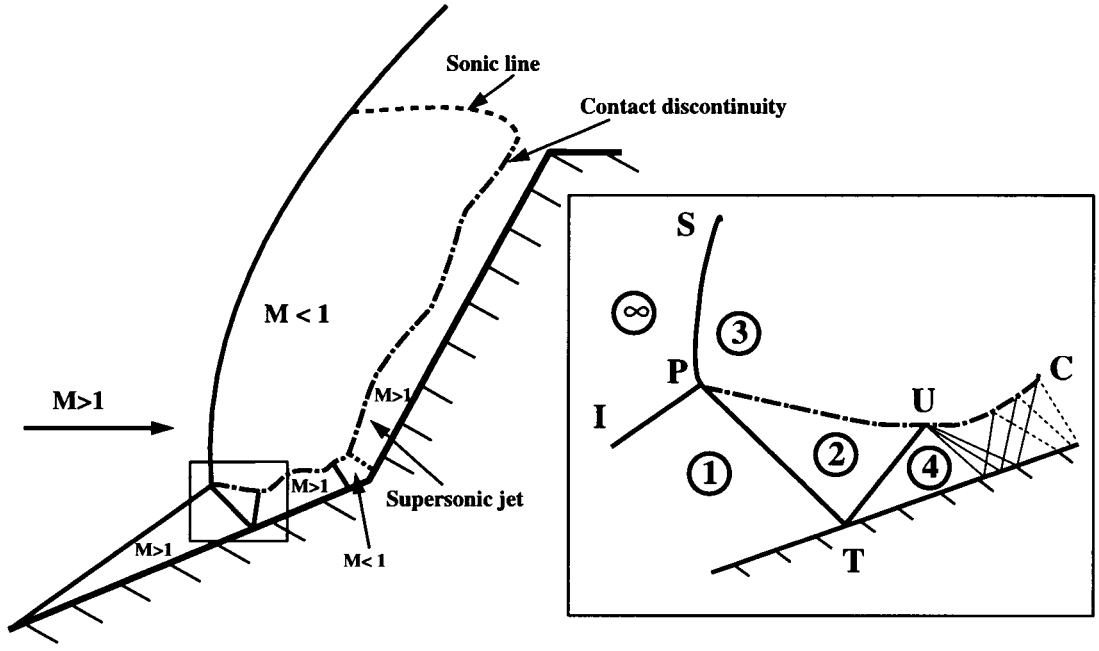


Fig. 8.17 Schematic diagram of a Type IVr shock interaction with a zoom of the interaction region.

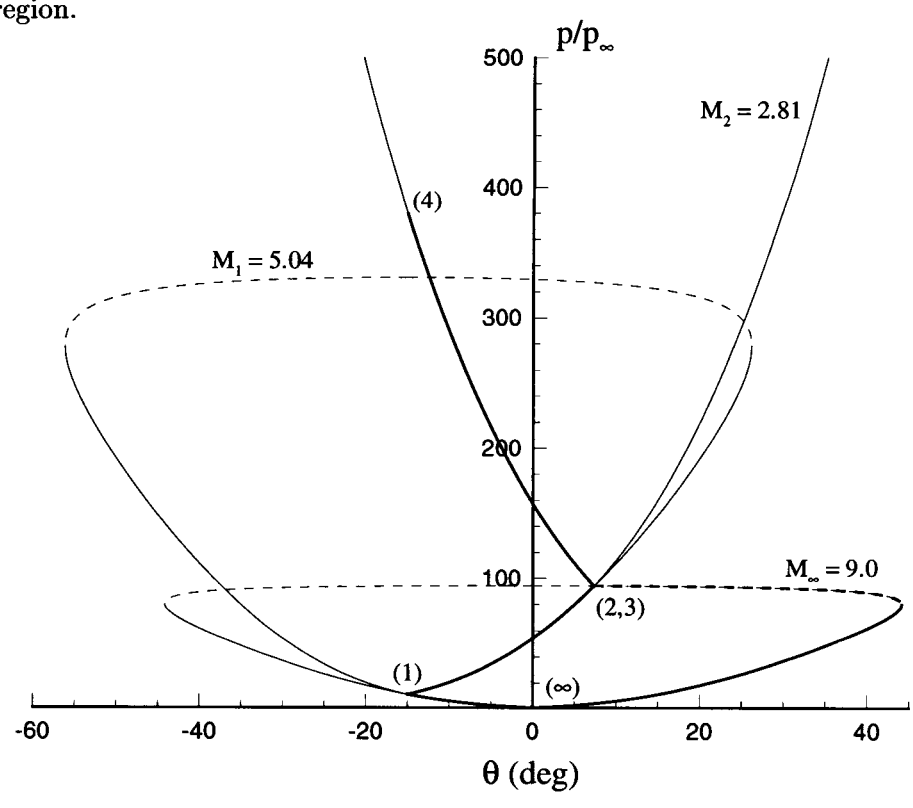


Fig. 8.18 Shock polar diagram for a Type IVr interaction with $\theta_1 = 15°$, $\theta_2 = 60°$, $M = 9$, and $\gamma = 1.4$.

Point (4) is determined by the fact that it must be co-located with Point (5). In the Type IVr interaction, Point (4) must be located on the Region (2) polar at the point $\theta = -15°$. In both cases, Points (2) and (3) are determined from the freestream Mach number and the first wedge angle; there is no effect of the second wedge angle on either the Type IV or the Type IVr shock polar diagram. Because the Type IV-Type IVr transition depends on the curvature of the bow shock near Point (P), which depends on the second wedge angle, it is impossible to predict the Type IV-Type IVr transition using the shock polar representation. Once again, due to curved shock effects, the shock polar predicts a pressure ratio in Region (4) of 375, which is less than the computed value of 430.

As the second wedge angle increases and the impingement point moves closer to the nose of the first wedge, the oblique shock from the nose impinges closer to the point of zero curvature on the bow shock, resulting in less downward streamline deflection after the triple point. This produces a lower pressure at the edge of the supersonic region adjacent to the body and reduces the strength of the reflected shock at the surface of the first wedge. This in turn reduces the first maximum in surface pressure and the amount of underexpansion in the supersonic jet. At the same time, Points (P) and (U) move closer to the wedge surface, decreasing the jet area and producing shorter peak-to-peak distances for the pressure maxima because the waves have less distance to travel between reflections. This can be seen in Fig. (8.19), which plots the surface pressure for a series of Type IVr interactions with increasing second wedge angle. The limiting case occurs when Point (P) reaches the nose of the wedge, and the bow shock is at the point of detaching. For this case there is no downward streamline deflection, and the post bow shock pressure on the surface is the normal shock jump value for the freestream Mach number, which is 94 times freestream pressure for this case. This flow can be thought of as perfectly expanded. The surface pressure is nearly constant until the sharp peak at the corner. The pressure then decreases to approximately the normal shock jump

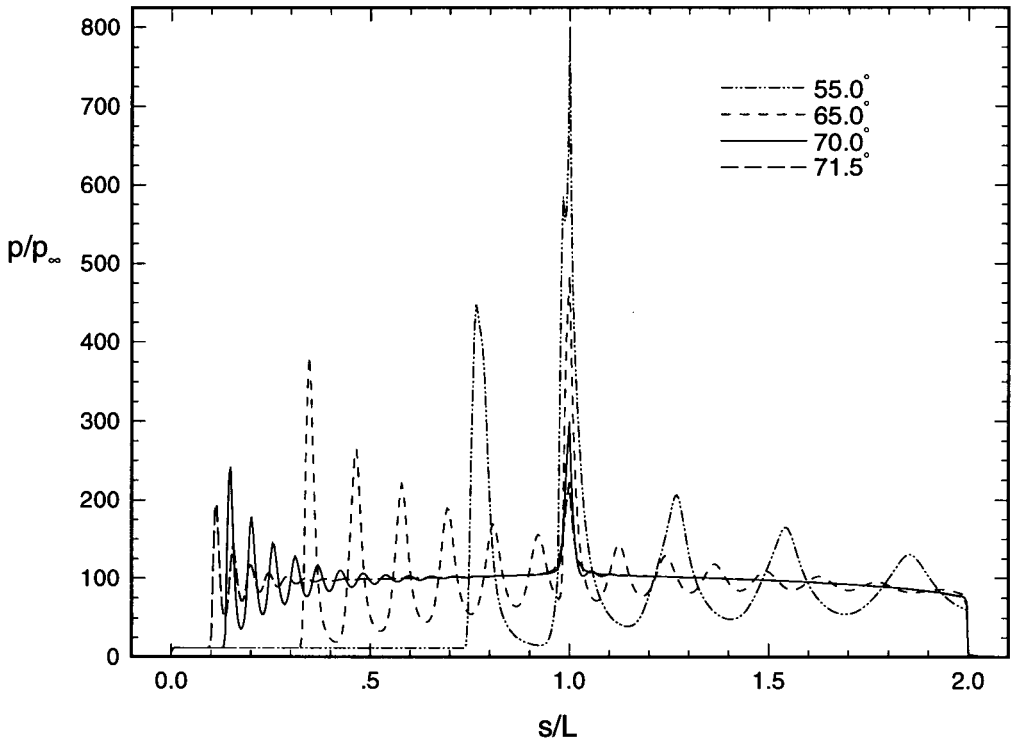


Fig. 8.19 Surface pressure for a series of Type IVr interactions with $\theta_1 = 15°$, $M = 9$, and $\gamma = 1.4$.

value by the end of the second wedge.

8.4.5 Transition Criteria

The transition criteria discussed above can be summarized on a parametric diagram of Mach number versus second wedge angle, shown in Fig. (8.20). The Type VI-Type V transition line is determined analytically from the method of characteristics, and predicts transition at significantly lower second wedge angles than the maximum deflection angle. The reason for this can be seen by considering Fig. (8.3). The expansion fan between Regions (2) and (3) turns the flow counterclockwise so that the angle of the contact discontinuity at the triple point is larger than the second wedge angle. Therefore, the flow in Region (4) is turned by an angle greater than the second wedge angle. Thus, transition occurs when the flow angle in Region (4) reaches the maximum deflection angle. The actual transition process

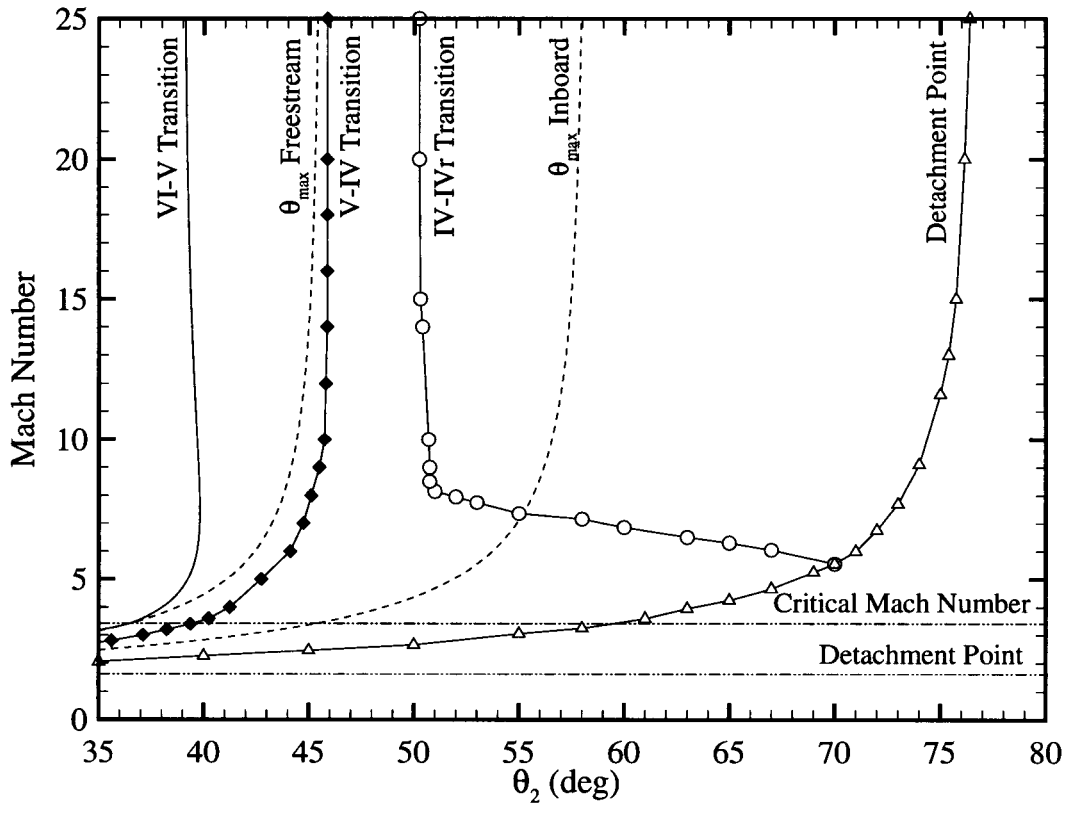


Fig. 8.20 Parametric diagram of Mach number vs. second wedge angle showing the regimes of the different interactions for high Mach number flows. $\theta_1 = 15°$, $\gamma = 1.4$, $L_1/L_2 = 1$.

occurs over a range of second wedge angles, and the computations agree with this analytical criterion in the sense that purely supersonic Type VI interactions cease to exist at the predicted second wedge angle.

The Type V-Type IV transition line cannot be determined analytically due to the subsonic flow and curved shocks present, and must be computed. Each symbol on this curve in Fig. (8.20) represents a numerically determined point. It can be seen that the flow transitions to a Type IV interaction quickly after the maximum deflection angle for the freestream, and much before the maximum deflection angle with respect to the inboard flow. The Type IV-Type IVr transition line has a strong Mach number dependence. For high Mach number flow (above Mach 8) transition occurs at about 50.5°. As the Mach number decreases below this point,

the transition to a Type IVr interaction is rapidly delayed, and in fact does not occur at all below Mach 5.5, as it is no longer possible to have a regular reflection at the wedge surface. This transition line also must be determined numerically. The curved line marked "detachment point" signifies when the triple point reaches the nose of the first wedge and the bow shock detaches, resulting in a single bow shock rather than a shock interaction. The lower horizontal line, at Mach 1.61, is the point at which the nose shock will always be detached on the 15° first wedge. The other horizontal line, at Mach 3.43, marks the distinction between the high and low Mach number regimes. Above this line the inboard flow downstream of the interaction region is always underexpanded, as described above. Below this line it is possible to have either overexpanded or underexpanded flows, as described in the next section. This is the basis for separating the discussion into high and low Mach number regimes. The value of this critical Mach number can be calculated analytically and depends only on γ and θ_1.

8.5 Low Mach Number Interactions

In this section we present results for inviscid double-wedge flows at $M = 2.8$ with varying second wedge angles. These results are characteristic of shock interactions that occur below the critical Mach number. The effect of varying the Mach number is discussed at the end of the section. As these low Mach number interactions are in many respects similar to the high Mach number cases, only the differences between them are discussed in detail.

8.5.1 Type VI Interactions

Figure (8.21) shows Mach contours for a Type VI interaction with $\theta_2 = 30°$. As compared to the high Mach number case, this and all low Mach number Type VI interactions are only slightly underexpanded. The expansion fan is very weak and turns the flow through an angle of only 0.085°, corresponding to a pressure ratio of approximately 0.995. Therefore, the expansion fan cannot be seen in Fig. (8.21).

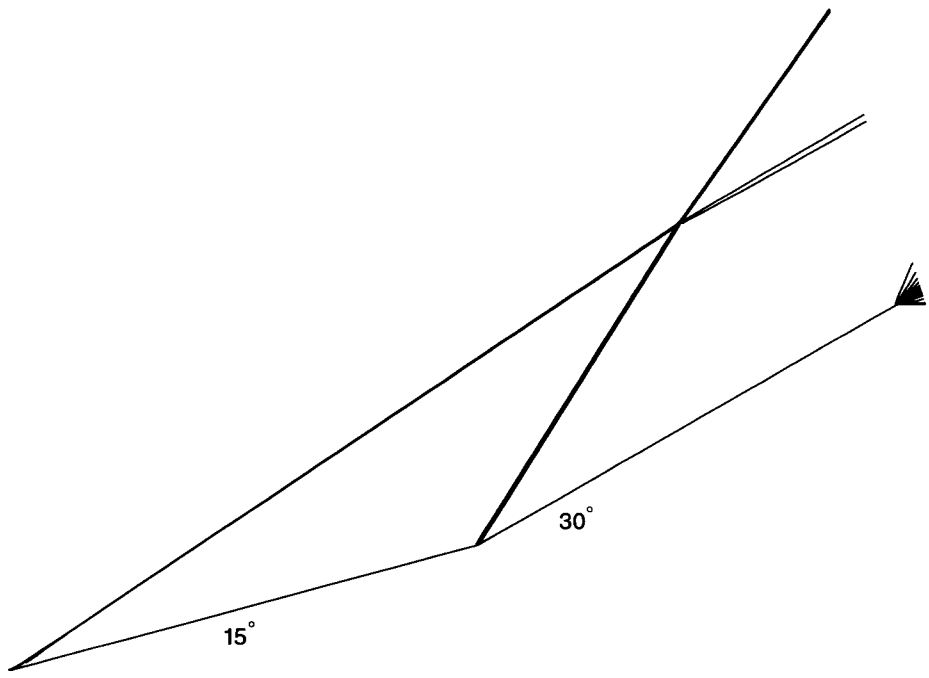


Fig. 8.21 Mach contours for a Type VI interaction with $\theta_1 = 15°$, $\theta_2 = 30°$, $M = 2.8$, and $\gamma = 1.4$.

As the second wedge angle increases, the flow becomes less underexpanded, eventually reaching a point where it is perfectly expanded and there are no waves downstream of the shock interaction point. A further increase in the second wedge angle results in an overexpanded flow where the pressure must increase after the interaction region to match the outboard flow conditions. Referring to Fig. (8.3), the shock polar for a Type VI interaction, Point (2) moves below Points (3) and (4), and a fourth shock wave replaces the expansion fan. This is now a Type I interaction, which in this context can be thought of as an overexpanded Type VI interaction.

8.5.2 Type I Interactions

The four shock Type I interaction is common when two shocks of different families intersect, but only rarely occurs after the intersection of two shocks from the same family. A schematic drawing of a Type I interaction is shown in Fig. (8.22).

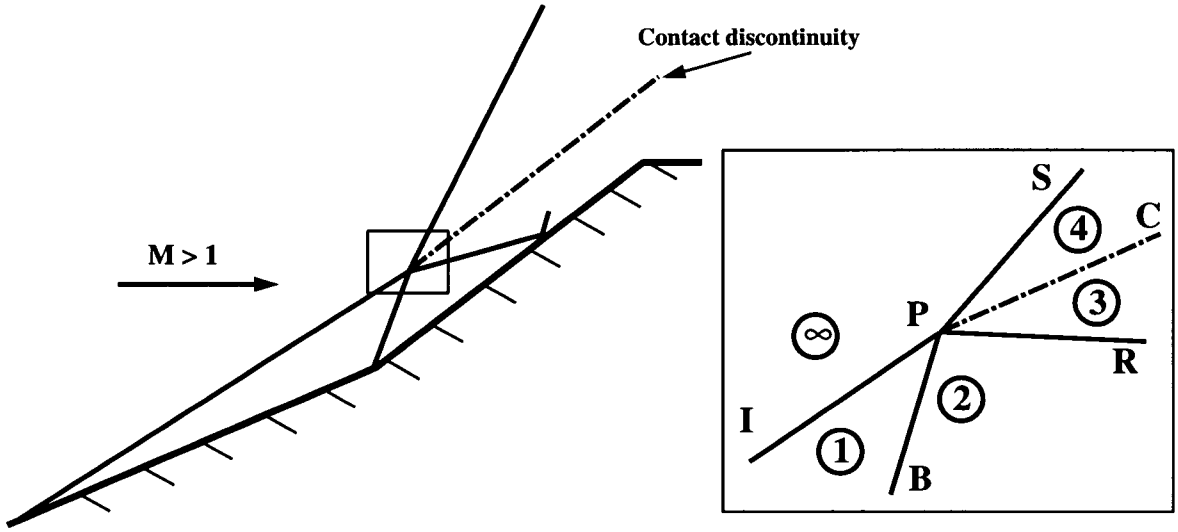


Fig. 8.22 Schematic diagram of a Type I shock interaction with a zoom of the interaction region.

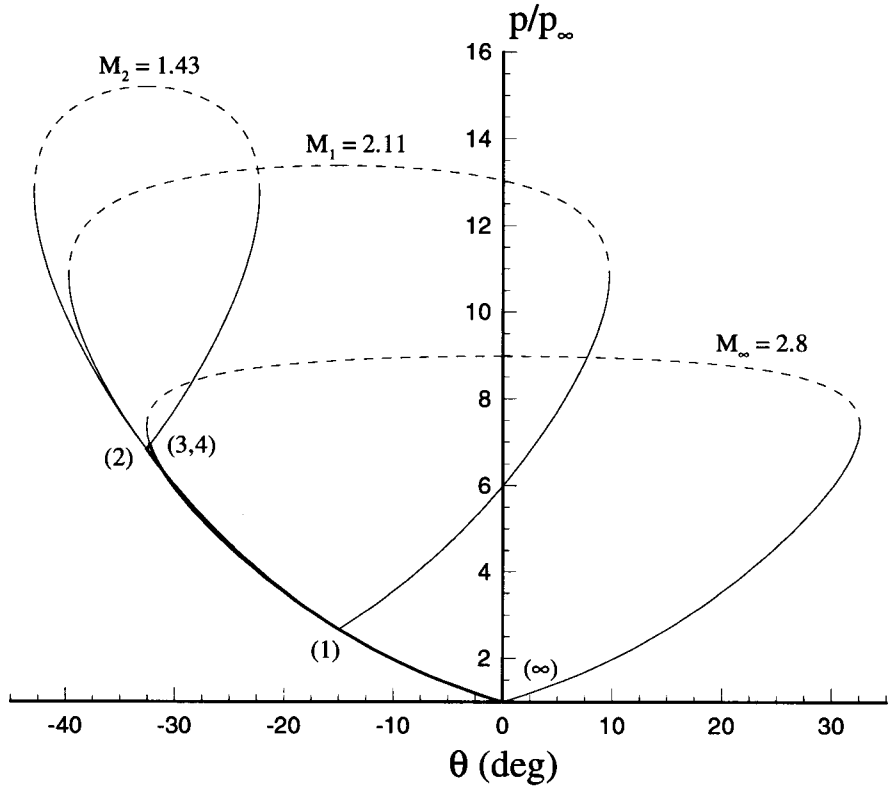


Fig. 8.23 Shock polar diagram for a Type I interaction with $\theta_1 = 15°$, $\theta_2 = 32.5°$, $M = 2.8$, and $\gamma = 1.4$.

The only difference between this Type I interaction and the Type VI interaction shown schematically in Fig. (8.2) is that an oblique shock (PR) has replaced the expansion fan. For double-wedge flows all Type I interactions are only weakly overexpanded. This can be seen in Fig. (8.23), which shows the shock polar for a Type I interaction with $M = 2.8$ and $\theta_2 = 32.5°$. Point (2) is at $\theta = -32.5°$ on the Region (1) polar, while Points (3) and (4) are on the intersection of the freestream polar with the Region (2) polar. The pressure in Region (3) is only 1.014 times greater than the pressure in Region (2). The perfectly expanded case occurs where the freestream polar and the Region (1) polar intersect. This occurs at $\theta_2 = 31.08°$ for Mach 2.8 flow. In the perfectly expanded case Points (2), (3), and (4) are all co-located. In the low Mach number regime, Type I interactions transition to Type V interactions. The Type I-Type V transition process is similar to the Type VI-Type V transition process that occurs at high Mach numbers.

8.5.3 Type V Interactions

While by definition the low Mach number Type V interaction has the same shock polar as the high Mach number case, the flowfield downstream of the interaction point is significantly different. This can be seen in Fig. (8.24), which shows the Mach contours for a Type V interaction with $\theta_2 = 35°$. As compared to the high Mach number case, Shock (TU) is much smaller and can not be distinguished in Fig. (8.24). However a small supersonic region exists, as evidenced by the shock polar for this interaction. Since Shock (TU) is vanishingly small for this case, the two contact surfaces (C_1) and (C_2) that bound the jet in the high Mach number flow merge into the jet-like feature that Edney identified. Neither this jet nor the contact surface (C_3) impinges on the body. Additionally, the flow is subsonic after the normal shock that strikes the second wedge. This flow is analogous to an overexpanded nozzle, where a normal shock forms inside the nozzle in order to match the downstream pressure condition. The pressure decreases smoothly along the second wedge after the normal shock, with the flow re-accelerating to supersonic speed.

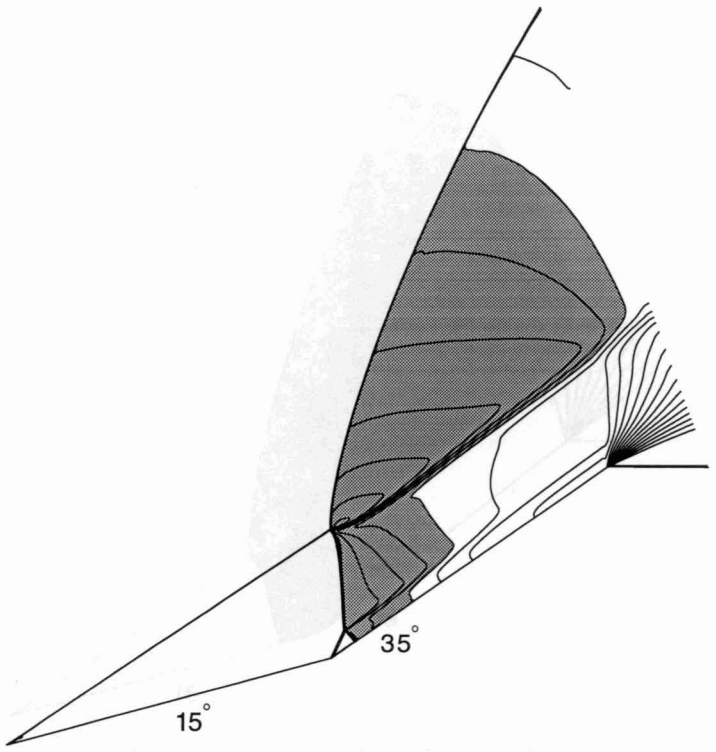


Fig. 8.24 Mach contours (subsonic region shaded) for a Type V interaction with $\theta_1 = 15°$, $\theta_2 = 35°$, $M = 2.8$, and $\gamma = 1.4$.

The Type V-Type IV transition criterion is the same as in the high Mach number regime.

8.5.4 Type IV Interactions

The Mach contours for a Type IV interaction with $\theta_2 = 38°$ are shown in Fig. (8.25). As compared to the high Mach number case shown previously in Fig. (8.10), Shock (PT) nearly disappears, and the underexpanded jet that is such a prominent feature of the typical Type IV shock interaction becomes a thick shear layer which does not impinge on the body. The computed flowfield behind Shocks (PS) and (TQ) is subsonic, including the shear layer that separates them. However, if we examine the shock polar for this case we see that the flow in Region (2) must still be supersonic ($M_2 = 1.16$) until it passes through Shock (TU), which is present in the calculation but is too small to be distinguished in Fig. (8.25). The subsonic

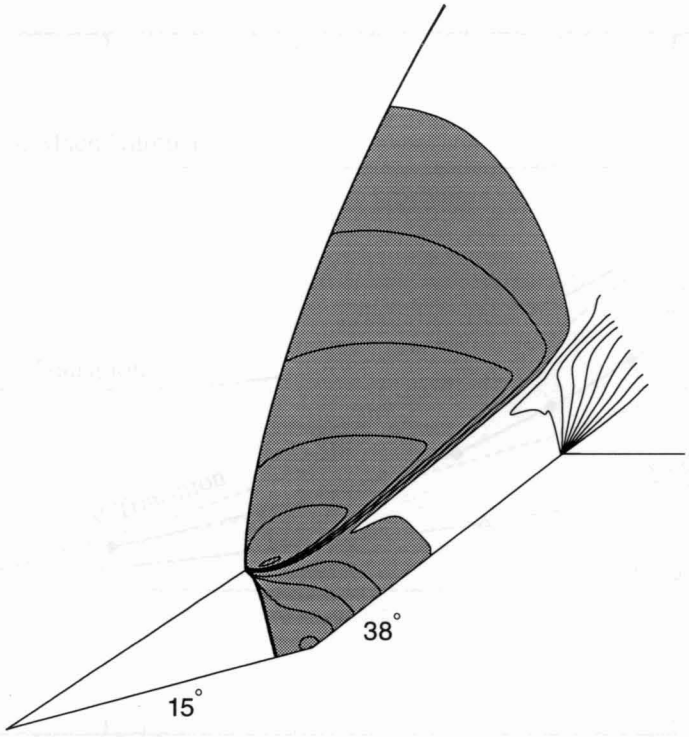


Fig. 8.25 Mach contours (subsonic region shaded) for a Type IV interaction with $\theta_1 = 15°$, $\theta_2 = 38°$, $M = 2.8$, and $\gamma = 1.4$.

flow downstream of the interaction smoothly accelerates to supersonic speed as it travels down the second wedge. A Type IVr interaction is impossible at this Mach number because a regular reflection cannot occur at the surface of the first wedge, as seen in Fig. (8.20). Therefore, the Type IV interaction is seen with increasing second wedge angle until the bow shock detaches from the nose.

8.5.5 Transition Criteria

Figure (8.26) shows the low Mach number regime of the M-θ_2 diagram. For small second wedge angles and high Mach numbers, a Type VI interaction occurs. In this region of the parameter space, the inboard flow downstream of the interaction is underexpanded. The Type VI-Type I transition line represents the locus of flow states for which the flow is perfectly expanded. Below this line, the flow is overexpanded and Type I interactions occur.

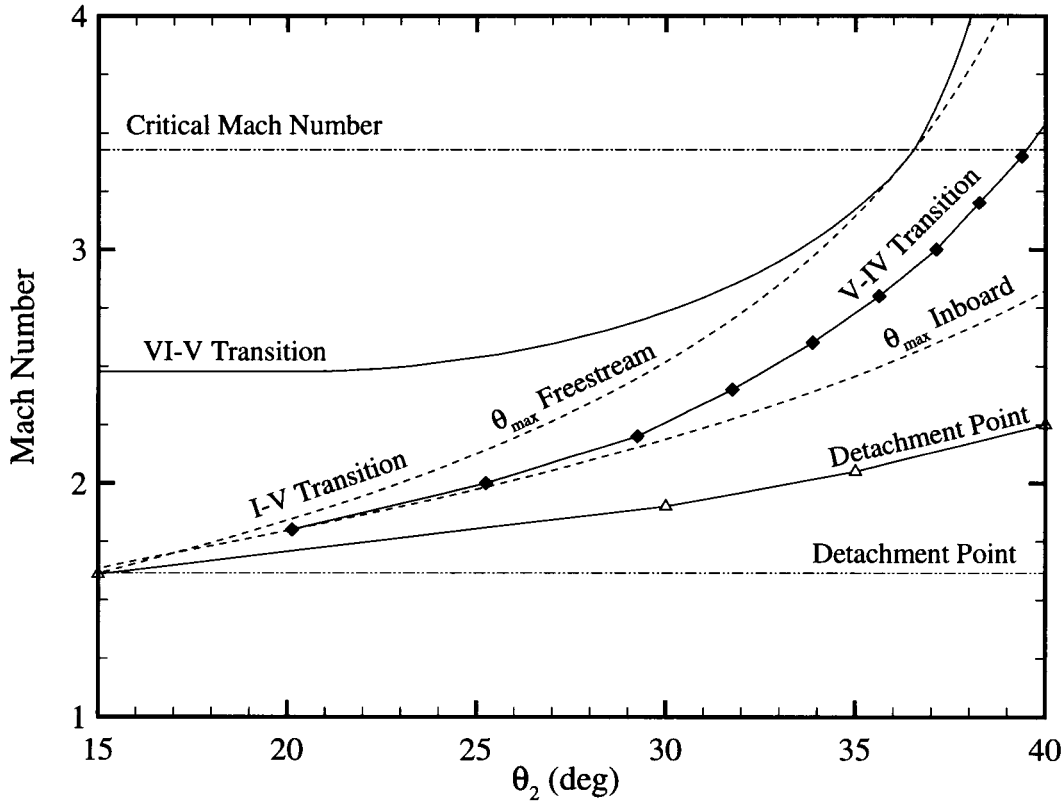


Fig. 8.26 Parametric diagram of Mach number vs. second wedge angle showing the regimes of the different interactions for low Mach number flows. $\theta_1 = 15°$, $\gamma = 1.4$, and $L_1/L_2 = 1$.

The Type I-Type V transition occurs at a second wedge angle equal to the maximum deflection angle for the freestream, because for second wedge angles greater than this maximum deflection angle, a single oblique shock cannot turn the freestream flow. While this condition still applies in the high Mach number regime, the fact that the flow downstream of the interaction point is underexpanded results in a more restrictive criterion for Type VI-Type V transition.

Unlike the Type VI-Type I transition line and the Type I-Type V transition line, the Type V-Type IV transition line must be calculated numerically. At low second wedge angles, this line approaches the maximum deflection angle for the inboard flow along the first wedge. The curved line labeled "detachment point" represents the point at which the bow shock detaches from the nose of the wedge,

while the lowest horizontal line represents the point at which the nose shock is detached.

8.6 Summary

We have performed detailed simulations of steady inviscid shock interactions on double-wedge geometries. Four of the interactions that occur (Type VI, Type V, Type IV, and Type I) fit into Edney's classification scheme. One new interaction that we term a Type IVr also occurs. This new shock interaction is similar to a Type IV interaction, except that the shock which impinges on the wedge surface undergoes a regular reflection rather than a Mach reflection as in the Type IV interaction. The Type IVr interaction occurs because of the geometrical constraints of the double-wedge and cannot occur for the type of blunt body flows investigated by Edney.

Our calculations show that there is a critical Mach number for these flows, above which the inboard flow along the wedge surface is always underexpanded with respect to the outboard flow. For the Type V, Type IV, and Type IVr interactions this can result in an underexpanded supersonic jet along the surface. The interaction of this jet with the adjacent subsonic region results in large amplitude steady pressure variations on the wedge surface. Below the critical Mach number, either underexpanded Type VI interactions or overexpanded Type I, Type V, or Type IV interactions occur. In the overexpanded interactions, the surface pressure decreases smoothly from a maximum at the corner of the wedges.

The transition criteria between the various interactions have also been identified. Below the critical Mach number, a Type VI-Type I transition occurs when the inboard and outboard flow downstream of the interaction region are perfectly expanded with respect to each other. A Type I-Type V transition occurs when the second wedge angle reaches the maximum deflection angle for the freestream flow. Above the critical Mach number, a Type VI-Type V transition occurs when

the outboard flow angle reaches the maximum deflection angle for the freestream. These transition criteria can be determined analytically using the method of characteristics, and the numerical results are in excellent agreement. The Type I-Type V and the Type VI-Type V transitions do not occur suddenly, but occur over a finite range of second wedge angles as the seven-shock Type V configuration develops. The Type V-Type IV and Type IV-Type IVr transition criteria cannot be predicted analytically and have been calculated numerically.

Chapter 9

Viscous Shock Interactions

9.1 Introduction

The previous chapter presented the results of a numerical study of inviscid shock interactions on double-wedge geometries. This study was performed to gain a better understanding of the underlying gas dynamics of these interactions, and to determine the transition criteria between the various types of interactions. However, the results presented in Chapter 8 are meant as a purely theoretical exercise, since in any real flow the viscous effects will alter the basic structure of the interaction.

Laminar viscous double-wedges have been examined numerically by Olejniczak *et al.* (1996a). These flows were shown to be considerably more complicated than the inviscid cases, primarily due to the separation zone which forms at the corner of the wedges, and the resulting separation shock which interacts with the oblique shock from the nose of the first wedge. In many cases the viscous effects alter the basic nature of the interaction, making it difficult to classify these flows using Edney's scheme. However, flow patterns similar to Edney's Type VI and Type V shock interactions were observed for certain cases. These flows were steady only for a limited range of Reynolds numbers and second wedge angles. Many of the unsteady flows exhibited an oscillatory behavior similar to the spike-tipped blunt body flows investigated by Maull (1960) and Wood (1962). Flow patterns similar to the Type IV shock interaction, with the accompanying pressure variations, were not observed for these laminar flows.

There is very little experimental work currently available in the area of shock interactions on double-wedge or double-cone geometries. Bertin and Hinkle (1975)

performed an experimental investigation of the Type V and Type VI interactions on double-wedges in which they prevented separation by bleeding off the boundary layer at the corner of the wedges, but it is unclear how the bleeding procedure and the three-dimensional end effects impact their results. In addition, Holden and Moselle (1970) and Holden (1978) have examined laminar hypersonic flow over two-dimensional compression corners. These flows exhibit similar features to the double-wedge flows of interest, including the separation region at the corner and the associated separation shock, but lack the strong shock-shock interaction that is the most interesting feature of the double-wedge flowfield.

In this chapter, we combine a series of experiments with a numerical analysis of viscous double-cone shock interactions. The models were chosen to be axisymmetric double-cone geometries, rather than double-wedges, to eliminate the possibility of three-dimensional end effects. A series of numerical calculations were undertaken to examine the very large parameter space of these flows in order to identify a small number of experimental test cases. In order to simplify this parametric study we have chosen to look only at cases in which the lengths of the two cones are equal. From this numerical study two baseline model geometries were selected, which are expected to produce a Type VI shock interaction and a Type V shock interaction. These models are run in the Princeton University Mach 8 blowdown wind tunnel, and the results are compared to laminar CFD calculations. Finally, further research is proposed which could provide experimental and computational evidence of a perfect gas Type IV shock interaction with surface pressure variations similar to those seen in reacting or inviscid flows.

9.2 Experimental Setup

The experiments were conducted in the Princeton University Gas Dynamics Laboratory Mach 8 blowdown facility, described in detail in Baumgartner *et al.* (1995). This wind tunnel has a 0.229 m (9 in) diameter test section with 4-way optical access. Run times of up to two minutes are possible. This facility is ideal

for our purposes, since it allows us to perform experiments well into the high Mach number regime without the complications of real gas effects. This permits us to directly examine the underlying fluid mechanics of these flows. Two nominal run conditions for the experiments were selected: a low Reynolds number (Re) condition, with a stagnation pressure of $p_o = 3.45 \pm 0.03$ MPa (500 ± 5 psi), and a high Re condition, with a stagnation pressure of 6.89 ± 0.07 MPa (1000 ± 10 psi). Both test conditions had a stagnation temperature of $T_o = 750 \pm 10$ K. Freestream conditions were then obtained by assuming a perfect isentropic expansion to Mach 8 flow from these known reservoir (stagnation) values, and are summarized in Table (9.1) below. Actual run conditions were typically within 10% of the nominal values and varied by less than 5% over the course of each run.

	Low Re	High Re
T_o	750 K	750 K
p_o	3.45 MPa	6.89 MPa
T_∞	54 K	54 K
p_∞	353 Pa	706 Pa
U_∞	1180 m/s	1180 m/s
Re_∞	7.4×10^6 m^{-1}	14.8×10^6 m^{-1}

Table 9.1. Nominal stagnation and operating conditions of the Princeton Mach 8 wind tunnel.

The two baseline models have identical first cone half angles of 25°. The second cone half angle is 35° for the first model and 50° for the second. Both single piece brass models were machined with a 0.051 m (2 in) base diameter. This diameter was selected because it was small enough to allow good tunnel starting characteristics, but large enough to allow detailed examination of the flow structure. The $25°-35°$ double-cone has equal face lengths of $L = 0.026$ m (1.00 in) and a 0.038 m (1.50 in) long cylindrical afterbody following the 35° face. The $25°-50°$ double-cone has face lengths of $L = 0.021$ m (0.84 in) and an afterbody length of 0.019 m (0.75 in). The models were mounted to a sting in the test section using bolts screwed into the aft end of the model.

The flow was examined by means of a video schlieren system, similar to the Toepler schlieren described by Merzkirch (1987). The light source was a white strobe followed by a narrow slit. Instead of lenses, large spherical mirrors ($f = 72$ in) were used. The strobe pulse length was approximately 2 microseconds, and the strobe speed was adjusted to match the 30 frames per second frame rate of the CCD camera. The slit and the knife edge were oriented 45° from horizontal in order for the system to be more sensitive to the density gradients expected in this direction. The CCD video output was recorded by a commercial grade VCR and subsequently digitized using an Imaging Technology Series 151 Image Processor. The digitized images were enhanced using image processing software to reduce noise and highlight the flow structure. The models were not instrumented and thus no surface data are available.

9.3 Numerical Method

The axisymmetric Navier-Stokes equations are solved using the modified form of Steger-Warming flux vector splitting discussed in Chapter 3. This method was validated for these calculations by matching the computed surface pressure and heat transfer with experimental data from compression corner flows. Freestream conditions were obtained by assuming a perfect isentropic expansion to Mach 8 flow from known stagnation values. A no-slip velocity condition was applied at the body surface, and an isothermal wall temperature of 600 K was assumed for all cases. Based on the work we have done on inviscid shock interactions, a grid size of 512×512 points was chosen. The solutions were advanced in time using the full matrix DP-LUR method described in Chapter 5. Between 10,000 and 25,000 iterations were typically required to reach a steady-state solution for these flows. Solutions were obtained on a 64 processor Thinking Machines CM-5, where between two and five hours of CPU time were required.

All of the calculations in this chapter assume laminar flow. This is probably a good assumption for the boundary layer on the first cone, since the transition

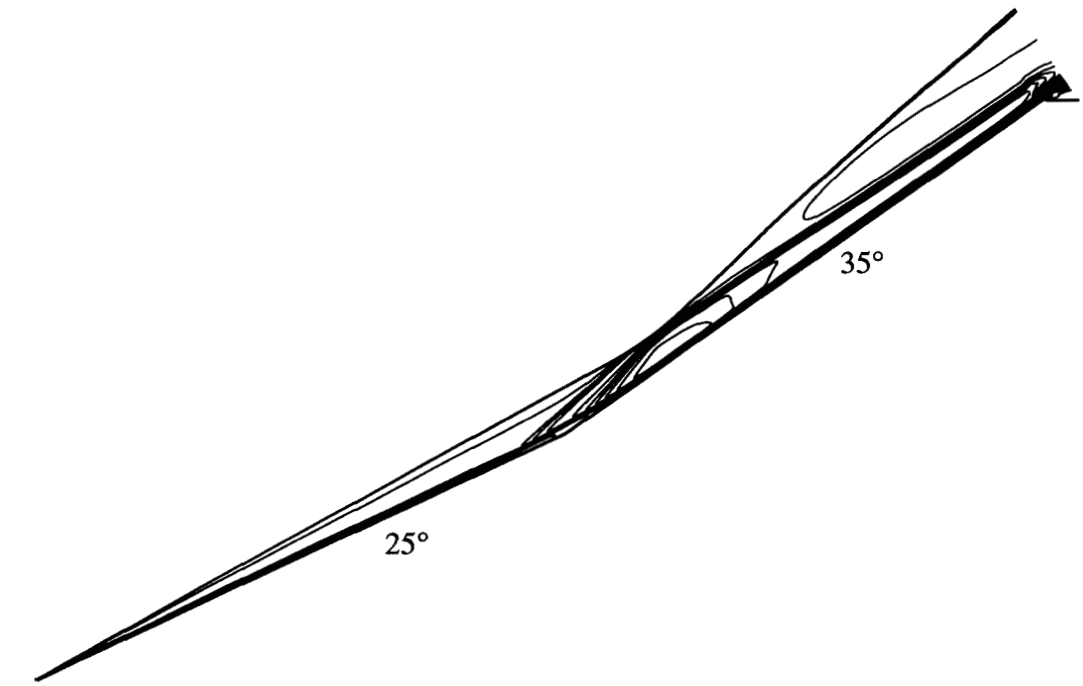
Reynolds number for a sharp cone in a wind tunnel [Anderson (1989)] at Mach 8 ($Re_{xt} \approx 3.5 \times 10^6$) is at least an order of magnitude larger than the Reynolds number on the first cone at all test conditions. However, at these Reynolds numbers it is possible that the boundary layer will become turbulent after the interaction with the separation and re-attachment shocks. Due to the complex nature of this flow we feel that the choice of an appropriate turbulence model is a difficult one. The presence of strong adverse pressure gradients and separated flow at the corner of the cones make the use of a zero equation or standard k-ϵ model inappropriate. In addition, the presence of free shear layers emitted from the shock intersection points complicates the choice of the turbulence model, since many models which give adequate results for purely wall bounded flows can show poor performance in the simulation of free shear layers [Malone (1996)]. Work is underway to choose an appropriate turbulence model for this flow and implement it in parallel. Further discussion of the possible role of turbulence in these flows will be presented below on a case-by-case basis.

9.4 Results

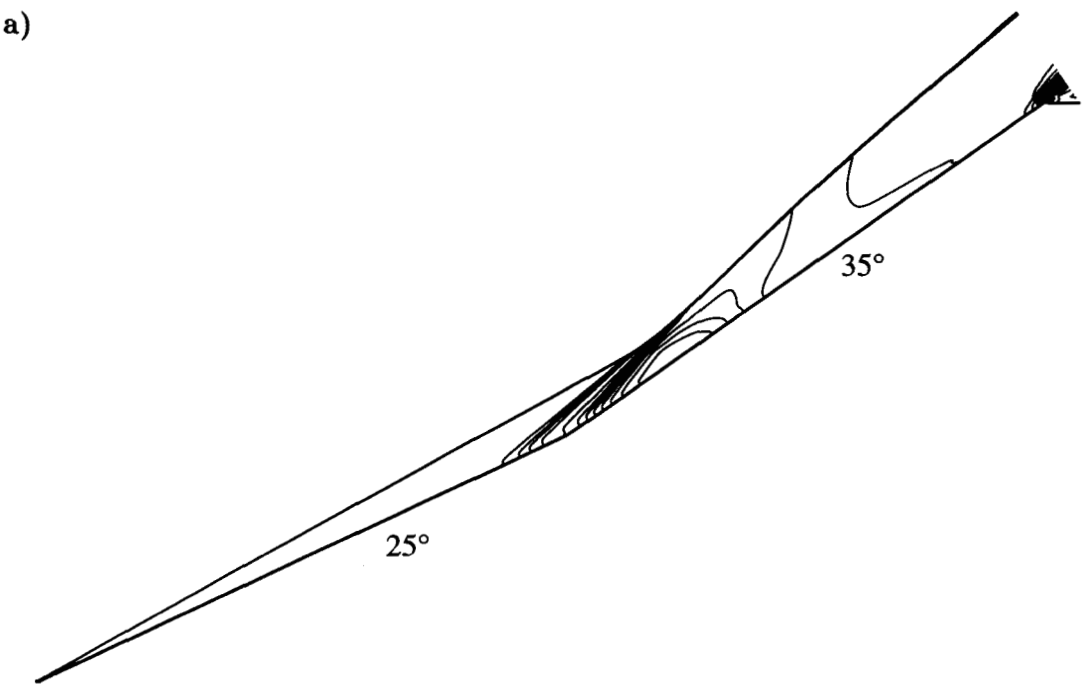
9.4.1 Type VI Interaction

The first test case chosen corresponds to a Type VI shock interaction as classified by Edney. The cone half angles are 25° and 35°. A series of runs were made at both the low and high Reynolds number operating conditions, with nominal freestream conditions given in Table (9.1) above. The Reynolds numbers based on freestream conditions and the model base diameter (0.051 m) are $Re = 3.75 \times 10^5$ for the low Reynolds number operating condition, and $Re = 7.50 \times 10^5$ for the high Reynolds number condition. The base diameter was chosen as the relevant length scale based on the work of Maull (1960).

Computed density and pressure contours for the low Reynolds number case are shown in Fig. (9.1). The features of this flow can be best described with the



a)



b)

Fig. 9.1 Computed contours of a) density and b) pressure for a Type VI shock interaction. Axisymmetric double-cone geometry with cone half angles $\theta_1 = 25°$ and $\theta_2 = 35°$. Laminar air at Mach 8, $T = 54$ K, and $Re = 3.75 \times 10^5$; 512×512 computational grid.

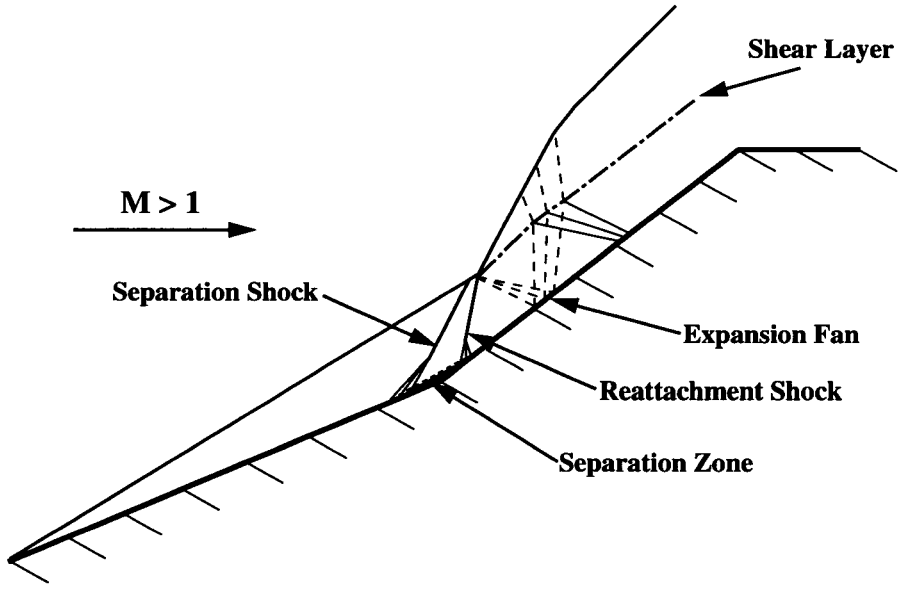


Fig. 9.2 Schematic diagram of a viscous Type VI shock interaction on a double-cone geometry.

help of a schematic drawing of a viscous Type VI interaction, shown in Fig. (9.2). All of the features shown in Fig. (9.2) can be seen in the computed pressure and density contours for this case. The oblique shock from the nose intersects the oblique shock from the corner of the cones. An expansion fan is emitted from the intersection point, which strikes and reflects off the body surface. The reflected expansion waves are then deflected as they pass through the shear layer emanating from the intersection point, and eventually intersect with and weaken the oblique shock. The waves are also partially reflected from the shear layer and slightly re-compress the flow along the surface of the second cone. There is a small separated flow region at the corner of the cones caused by the adverse pressure gradient. The separation and re-attachment shocks can be seen in Fig. (9.1).

Figure (9.3a) shows an experimental schlieren photograph of the low Reynolds number Type VI interaction, and Fig. (9.3b) shows a zoom of the interaction region for the same case. Although it is difficult to see the fine detail of the flow due to the small model size, the major structures of the interaction can be resolved, and are qualitatively similar to the computational results. The size of the separation

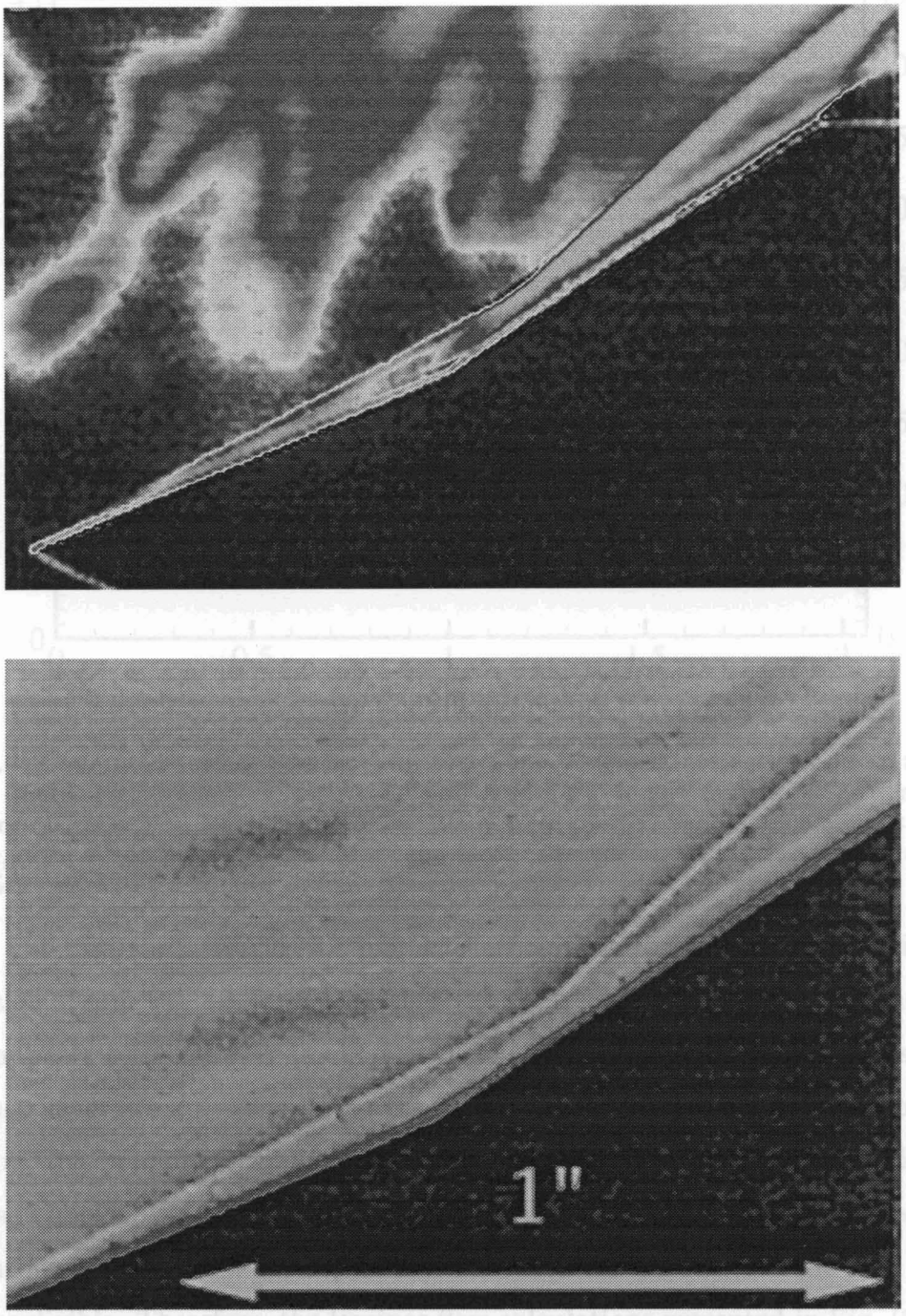


Fig. 9.3 Experimental schlieren image of a Type VI shock interaction on an axisymmetric double-cone geometry with cone half angles $\theta_1 = 25°$ and $\theta_2 = 35°$. Air at Mach 8, $T = 54$ K, and $Re = 3.75 \times 10^5$. a) Entire flowfield and b) zoom of the interaction region.

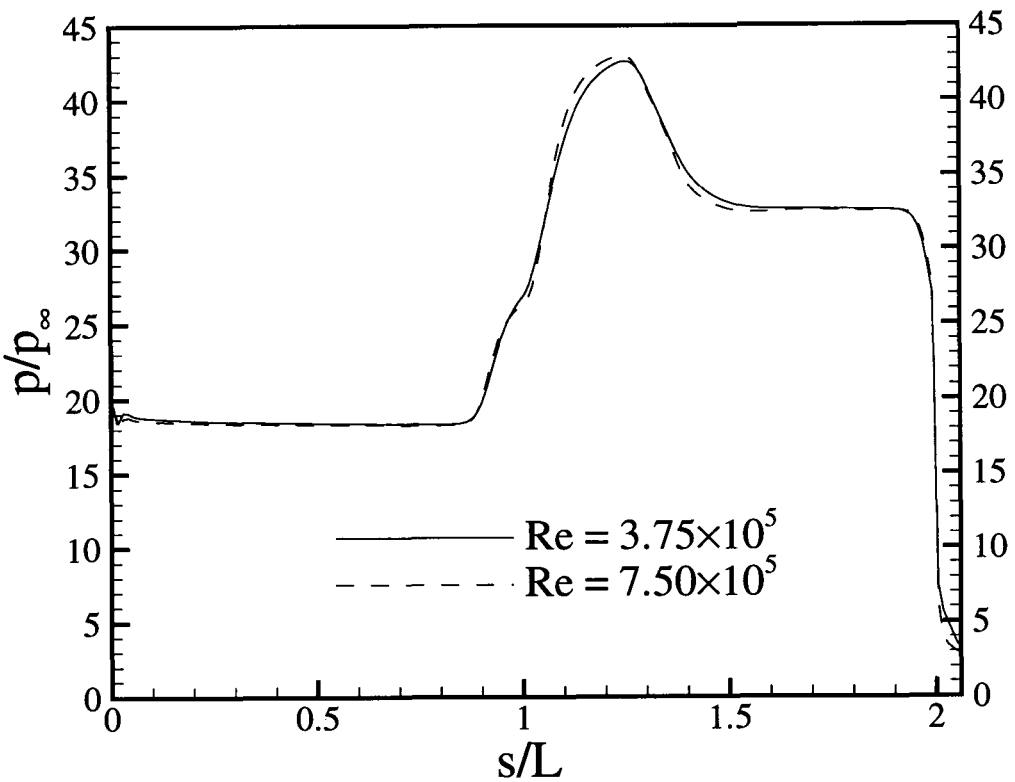


Fig. 9.4 Computed surface pressures for two Type VI shock interactions on an axisymmetric double-cone geometry with cone half angles $\theta_1 = 25°$ and $\theta_2 = 35°$, showing influence of freestream Reynolds number. Values are plotted vs. distance along the cone surface, normalized by the length of a cone face. Laminar air at Mach 8 and $T = 54$ K. 512 x 512 computational grid used.

region and the shock angles all match the computation to within experimental resolution. From Fig. (9.3b) we see that the shear layer emanating from the triple point appears to be entirely laminar, as is expected since the flow gradients across the shear layer are quite small and the flow on each side is entirely supersonic. In addition, the boundary layer on the first cone is also expected to be laminar, as discussed above. For this case the separation and re-attachment shocks are weak, and turbulence effects in the separation zone and on the second cone face appear to be negligible. Therefore, the laminar flow calculations presented in Fig. (9.1) are an accurate representation of this flow.

Figure (9.4) shows the computed normalized surface pressure for both the low

($Re = 3.75 \times 10^5$) and high ($Re = 7.50 \times 10^5$) Reynolds number cases. The first rise in surface pressure corresponds to the separation point, and is followed by a broad peak at about 43 times freestream corresponding to the compression waves which coalesce into the re-attachment shock. The pressure then decreases as the expansion fan strikes the body and is reflected. As seen in Fig. (9.4), the flowfield is nearly unaffected by this small (factor of two) variation in Reynolds number. This is also observed experimentally, where there is no detectable difference between schlieren images of the low and high Reynolds number runs.

9.4.2 Type V Interaction

As the second cone half angle is increased, a point will be reached where a Type VI interaction is no longer possible and a Type V interaction occurs. At this point the freestream flow can no longer be turned by an oblique shock on the second cone, and a curved bow shock appears. The second test case, with cone half angles of 25° and 50°, falls into this category. As before, nominal freestream conditions for this case are given in Table (9.1) above. Figure (9.5) shows the computed density contours for the low Reynolds number condition. Again, the features of this flow can be best described with the help of a schematic diagram of a typical viscous Type V shock interaction, shown in Fig. (9.6). The separation zone for this case is much larger than the Type VI shock interaction, and extends halfway down the first cone. A shock is formed at the separation point, which interacts with and steepens the oblique shock from the nose of the first cone. This oblique shock intersects the bow shock from the second cone, and third shock is transmitted from the intersection point toward the body. The re-attachment shock then intersects this transmitted shock. Finally, another oblique shock is transmitted from this intersection point, strikes the body surface, and undergoes a regular reflection. A thick shear layer is emitted from the intersection point on the bow shock, which separates the subsonic flow from the supersonic inboard flow.

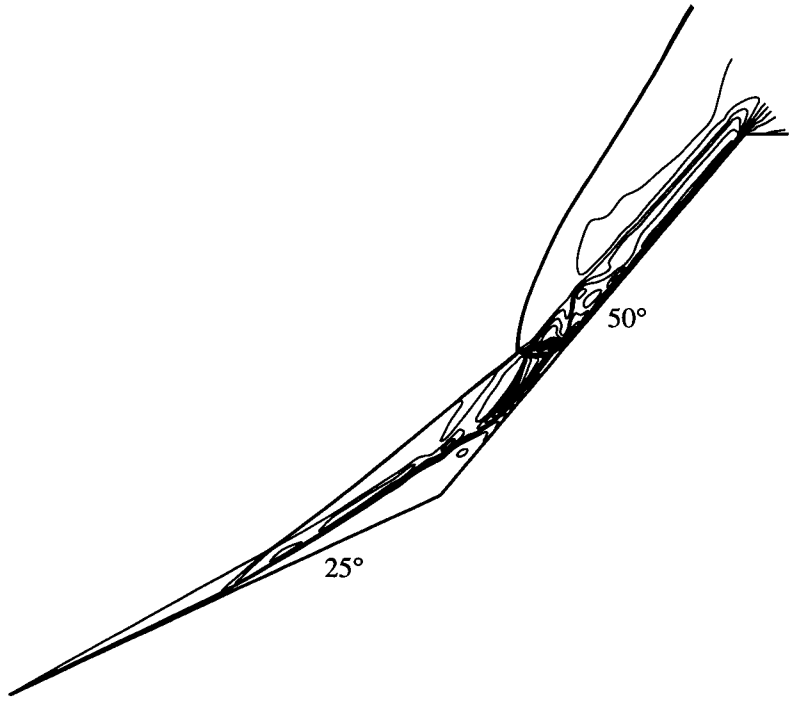


Fig. 9.5 Computed density contours for a Type V shock interaction. Axisymmetric double-cone geometry with cone half angles $\theta_1 = 25°$ and $\theta_2 = 50°$. Laminar air at Mach 8, $T = 54$ K, and $Re = 3.75 \times 10^5$. 512×512 computational grid used.

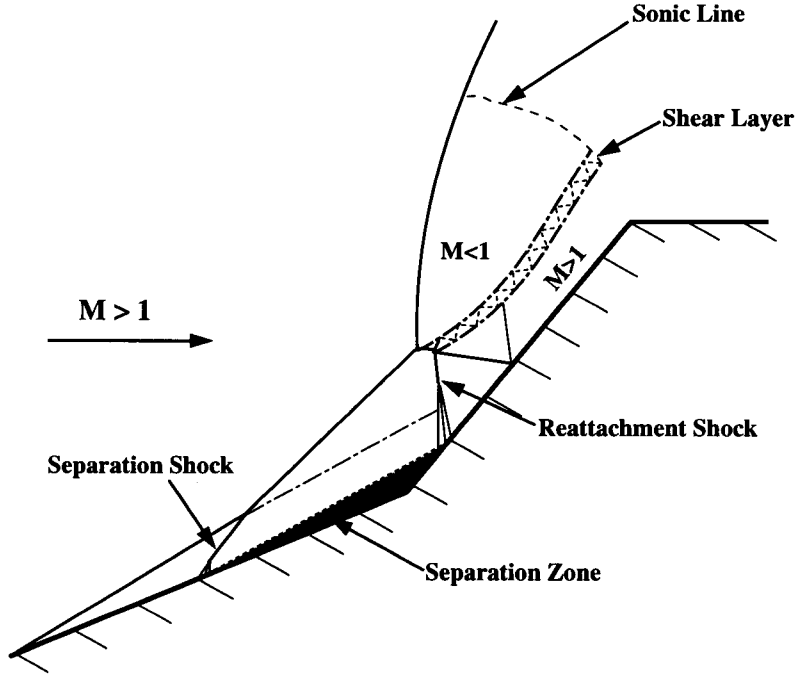


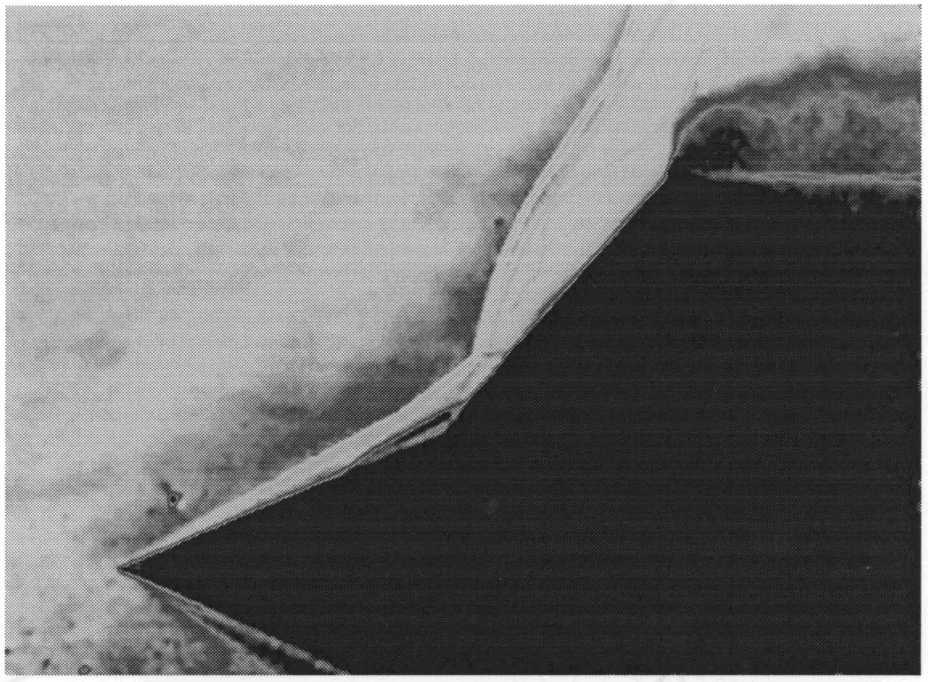
Fig. 9.6 Schematic diagram of a viscous Type V shock interaction on a double-cone geometry.

Early experimental results from this case were puzzling. Figure (9.7) shows two typical experimental schlieren photographs of the low Reynolds number Type V shock interaction taken at different times during the same run. While the interaction looks qualitatively similar to the CFD solution, it is apparent that this flow is unsteady, in contrast to the computational result. This can be seen by comparing Fig. (9.7a) to Fig. (9.7b). We see that in Fig. (9.7b), the separation zone at the corner has become much smaller, and the shock interaction thus occurs closer to the corner of the cones. In addition, the shape of the bow shock and even the angle of the oblique shock from the nose appear to be changing. The bow shock also appears to steepen near the top of the image. This unsteadiness can be clearly observed in videos taken of these runs, and is not simply a transient phenomenon. Runs of up to one minute in duration have been made, and all show this same basic unsteady flow pattern. Similar results were obtained for the high Reynolds number operating condition [Wright *et al.* (1997b)].

After careful analysis of the experimental data carried out at Princeton University, the unsteadiness was determined to be caused by a turbulent interaction between the bow shock and the boundary layer on the wall of the test section. This problem was overcome by reducing the size of the model, so that the bow shock impinged on the test section wall further downstream. The model diameter was reduced to 0.038 m (1.50 in) and another set of experiments were run. The Reynolds numbers based on the new model diameter are $Re = 2.80 \times 10^5$ for the low Reynolds number operating condition, and $Re = 5.60 \times 10^5$ for the high Reynolds number condition.

Figure (9.8) shows the computed density and pressure contours for the low Reynolds number condition. The flow is qualitatively similar to the computational results for the larger model, shown in Figure (9.5). Note the inflection point in the bow shock near the middle. This shock shape indicates that pressure waves are being transmitted from the supersonic fluid near the body surface into the subsonic

a)



b)

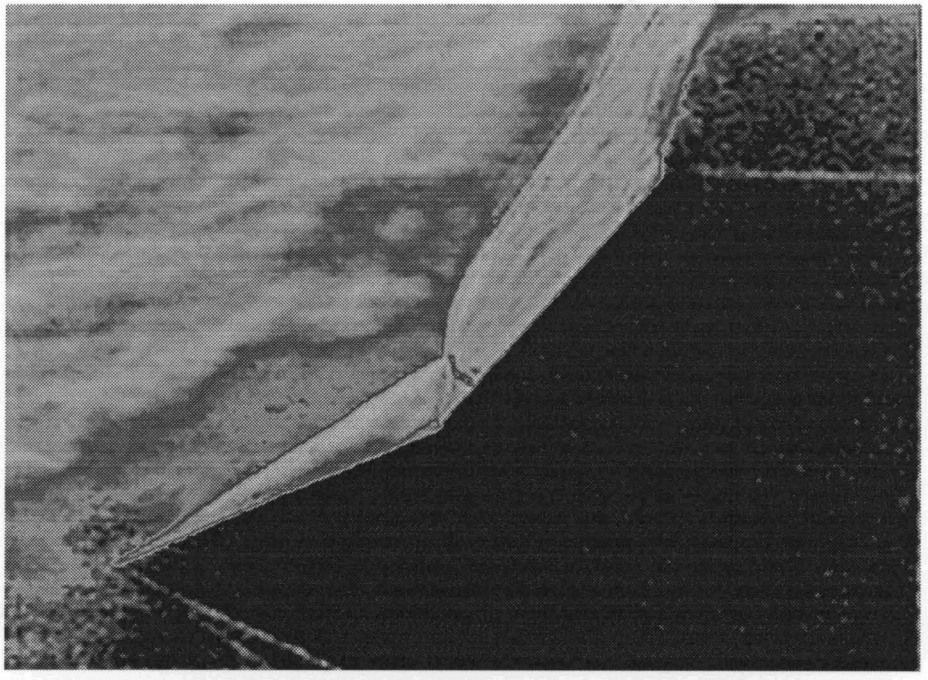
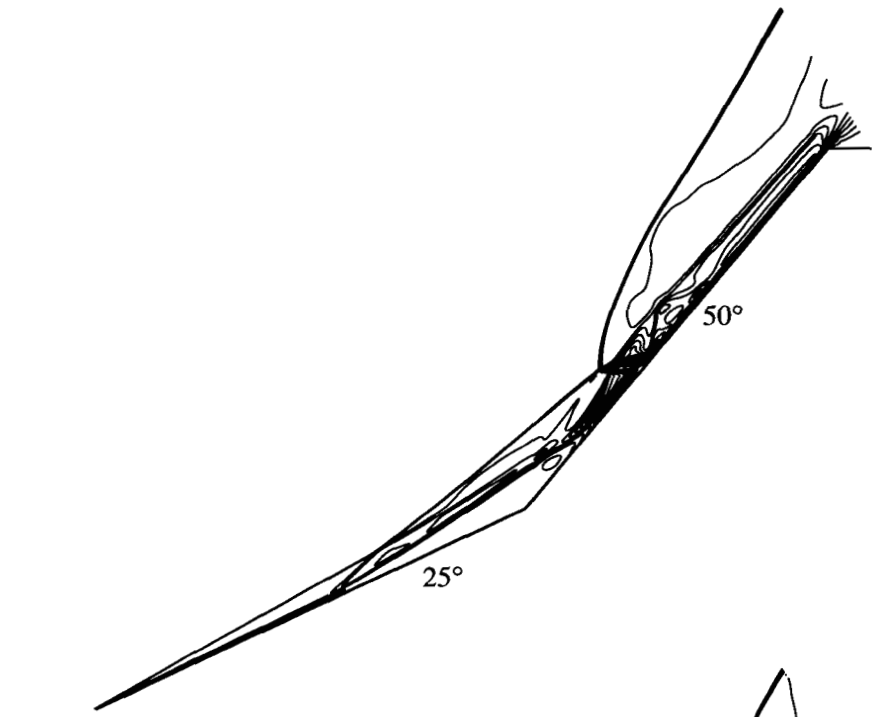
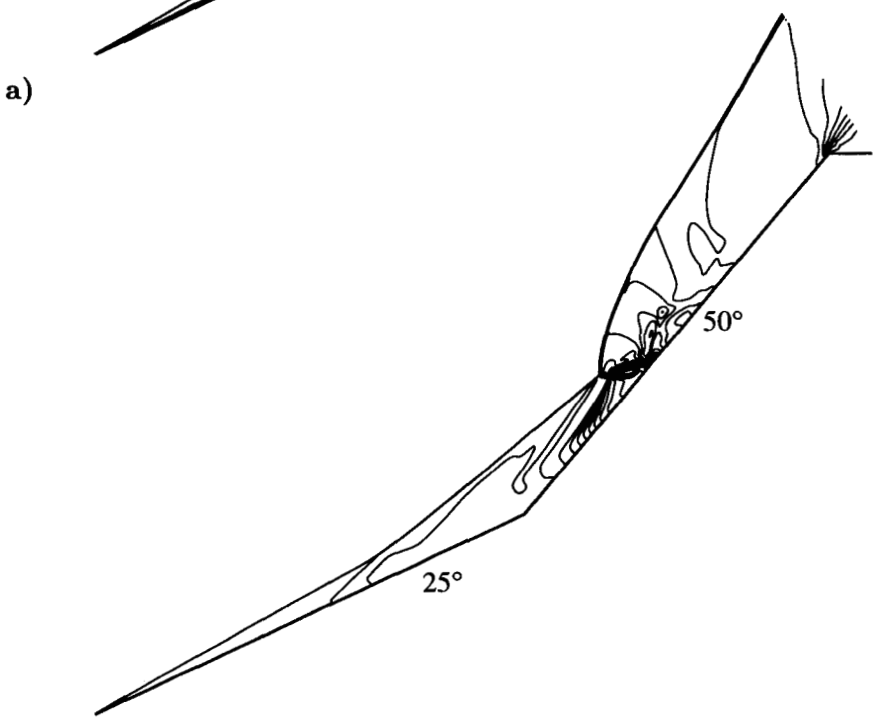


Fig. 9.7 Experimental schlieren images of a Type V shock interaction on an axisymmetric double-cone geometry with cone half angles $\theta_1 = 25°$ and $\theta_2 = 50°$. Air at Mach 8, $T = 54K$, and $Re = 3.75 \times 10^5$. Images taken at different times during the same experimental run.



a)



b)

Fig. 9.8 Computed contours of a) density and b) pressure for a Type V shock interaction. Axisymmetric double-cone geometry with cone half angles $\theta_1 = 25°$ and $\theta_2 = 50°$. Laminar air at Mach 8, $T = 54$ K, and $Re = 2.80 \times 10^5$; 512×512 computational grid.

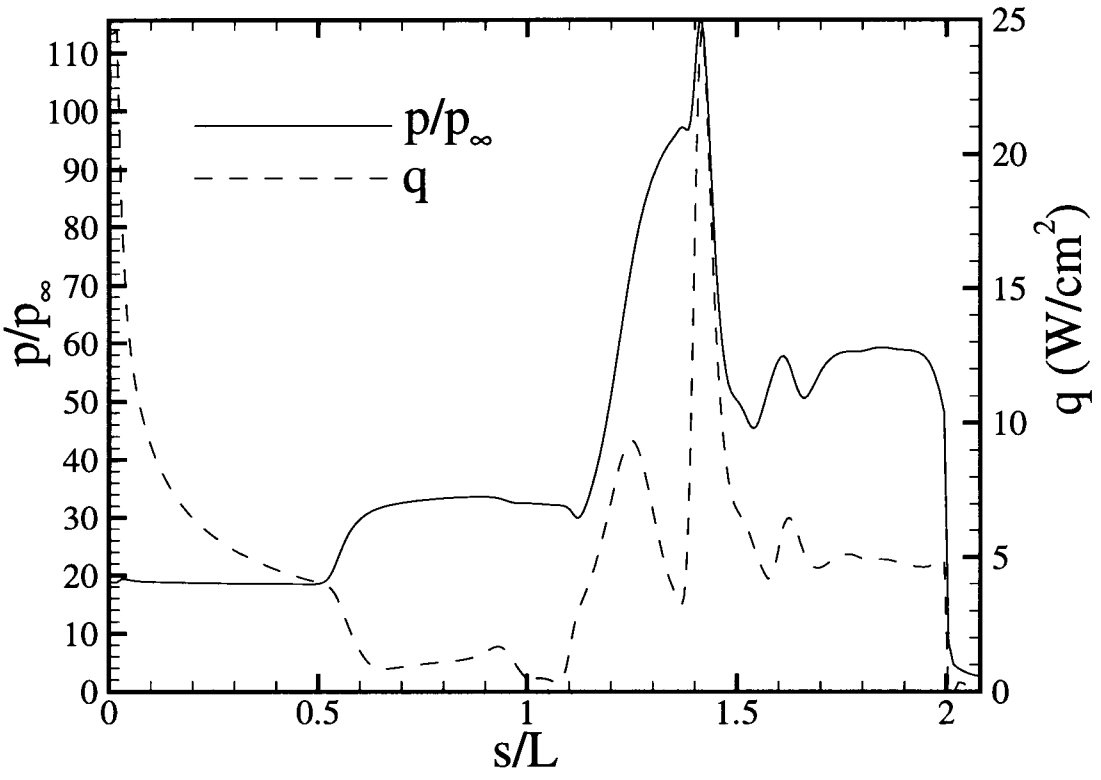


Fig. 9.9 Computed surface pressure and heat transfer rate for a Type V shock interaction on an axisymmetric double-cone geometry with cone half angles $\theta_1 = 25°$ and $\theta_2 = 50°$. Values are plotted vs. distance along the cone surface, normalized by the length of a cone face. Laminar air at Mach 8 and $T = 54$ K; 512 × 512 computational grid.

outboard flow. These compression waves strike the bow shock and steepen it slightly, creating the inflection point. This phenomena was described in Chapter 8 for an inviscid Type V interaction, and was shown to be caused by an underexpanded jet trapped between the body surface and the subsonic outboard flow. However, in this case the shear layer does not show the characteristic sinuous shape that would also indicate the presence of an underexpanded jet.

The computed normalized surface pressure and heat transfer for this case are presented in Fig. (9.9). The separation point is clearly visible as the first pressure rise at about the midpoint of the first cone ($s/L = 0.53$), and the matching dip in heat transfer. The second broad pressure peak, at 95 times the freestream pressure, marks the location of the re-attachment shock. The highest pressure (115 times

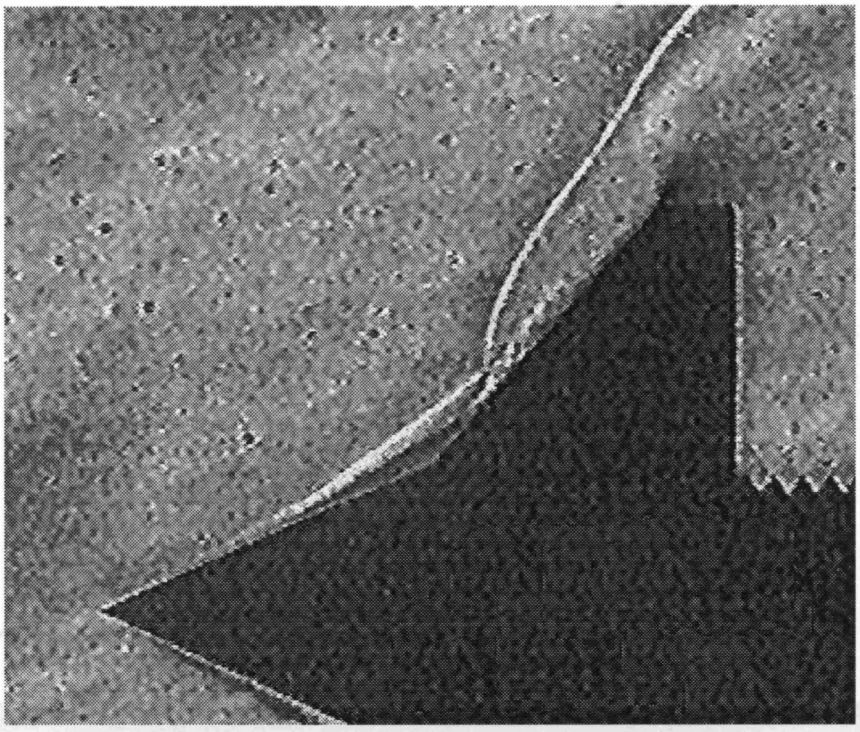
freestream) and heat transfer (25 W/cm^2) occur after the transmitted shock impinges on the second cone face and undergoes a regular reflection (see Fig. 9.6). One further small peak in both surface pressure and heat transfer is visible on the second cone face, due to a weak reflection of Mach waves trapped between the shear layer and the wedge surface. This flow thus exhibits characteristics similar to the inviscid Type V and Type IV shock interactions discussed earlier, however in this case the underexpanded jet is not fully formed and only a single small pressure peak is observed.

Figure (9.10a) shows an experimental schlieren photograph of the low Reynolds number Type V interaction, and Fig. (9.10b) shows a zoom of the interaction region. We see that the flow is now steady, and the bow shock does not steepen near the top of the image. The agreement with the computation is excellent for this case. Table (9.2) summarizes the locations of the separation and shock impingement points for both the CFD and experiment. Locations are are given versus distance along the cone surface, normalized by the length of a cone face. We see that the predicted location of the separation point is within 6% of the experiment, and the predicted location of the shock impingement is within 3%.

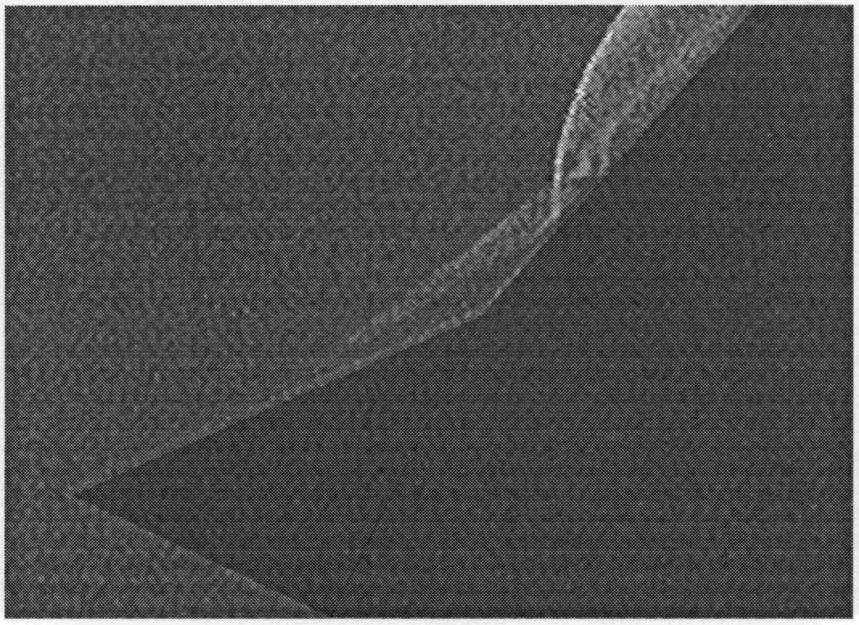
	Experiment	CFD
Separation	0.53	0.56
Impingement	1.36	1.40

Table 9.2. Experimentally and computationally determined locations of the separation and shock impingement points for a Type V shock interaction. Axisymmetric double-cone geometry with cone half angles $\theta_1 = 25°$ and $\theta_2 = 50°$. Laminar air at Mach 8, $T = 54$ K, and $Re = 2.80 \times 10^5$. Locations given in terms of distance along the cone surface, normalized by the length of a cone face.

Figure (9.11a) shows the computed density contours for the high Reynolds number condition, and Fig. (9.11b) shows an experimental schlieren photograph of the same case. Although the high Reynolds number flow is qualitatively similar to the low Reynolds number case, there are some significant differences, even for this factor of two difference in Re. The predicted and experimental separation and shock im-

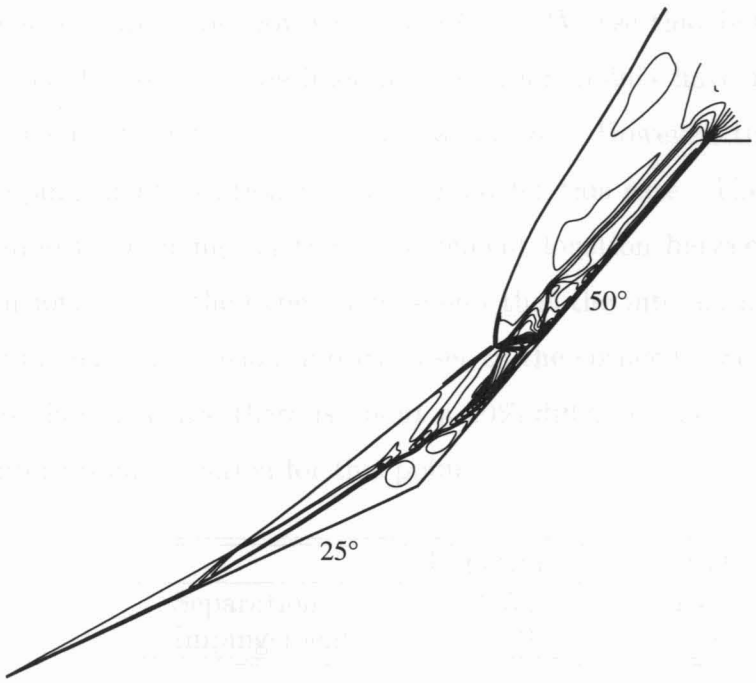


a)

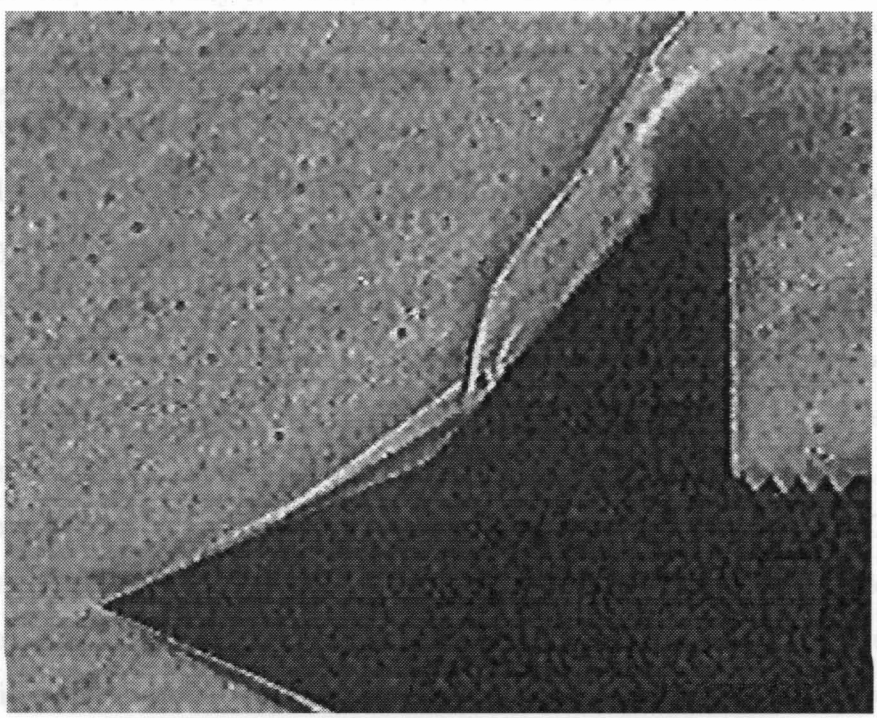


b)

Fig. 9.10 Experimental schlieren image of a Type V shock interaction on an axisymmetric double-cone geometry with cone half angles $\theta_1 = 25°$ and $\theta_2 = 50°$. Air at Mach 8, $T = 54K$, and $Re = 2.80 \times 10^5$. a) Entire flowfield and b) zoom of the interaction region.



a)



b)

Fig. 9.11 a) Computed density contours and b) experimental schlieren image, of a Type V shock interaction on an axisymmetric double-cone geometry with cone half angles $\theta_1 = 25°$ and $\theta_2 = 50°$. Air at Mach 8, $T = 54K$, and $Re = 5.60 \times 10^5$.

pingement locations are shown in Table (9.3). We see that both the computationally predicted and the experimental separation points have moved closer to the nose. Agreement for this point is still within 6%. However, the agreement for the shock impingement location is not as good for this case. The CFD solution predicted almost no change in the impingement location between the low and high Re conditions, while the experiment shows that the interaction, and therefore the shock impingement location, moves closer to the corner between the wedges as Re increases. For this case there is about a 10% difference between the experimental and computational location for this point.

	Experiment	CFD
Separation	0.52	0.49
Impingement	1.29	1.41

Table 9.3. Experimentally and computationally determined locations of the separation and shock impingement points for a Type V shock interaction. Axisymmetric double-cone geometry with cone half angles $\theta_1 = 25°$ and $\theta_2 = 50°$. Laminar air at Mach 8, $T = 54$ K, and $Re = 5.60 \times 10^5$. Locations are given in terms of distance along the cone surface, normalized by the length of a cone face.

While work is still underway to determine the cause for this discrepancy, one possibility is the effect of turbulence. It is expected that this flow will be more strongly affected by turbulence than the Type VI interaction, and therefore these laminar results may not accurately reflect the true flow conditions. As before, the boundary layer on the first cone should be entirely laminar, but in this case the separation and re-attachment shocks are much stronger, and may force transition. The boundary layer on the second cone would then be fully turbulent. The effect of turbulence on the free shear layers can be assessed following the work of Birch and Keyes (1972) who experimentally measured transition Reynolds number on compressible shear layers produced by shock interactions. Birch and Keyes predict transition at a local Reynolds number (Re_{lt}) of about 3×10^4 to 6×10^4, where Re_{lt} is calculated based on local flow quantities on the supersonic side of the shear layer, and l is the total length of the shear layer at the transition point. Based on

this criterion, we expect the shear layer emitted from the intersection point on the bow shock to be laminar for most of its length, but the shear layer bounding the separated flow at the corner of the cones will transition almost immediately, and should become fully turbulent for the high Reynolds number case.

Therefore, one possible explanation for the results shown here is that the low Reynolds number case may remain fairly laminar until after re-attachment, and then possibly become turbulent only on the second cone face. However, the high Reynolds number case may be transitioning in the separation region. This would then alter the structure of the interaction, and could result in the differences seen between the experiment and the laminar CFD. Further experiments are planned with an instrumented model which will help to determine the amount of turbulence in the flow, and a turbulence model is being prepared which will be implemented in the CFD code.

Once this problem has been resolved there are other interesting features of these double-cone shock interactions that can be explored. At still larger second cone angles, inviscid theory [Olejniczak *et al.* (1996a)] predicts that the Type V shock interaction will be replaced by a Type IV interaction with the characteristic underexpanded jet trapped against the body surface. This leads to a pattern of steady large amplitude variations in the flow quantities along the surface of the cones. These variations can have a large effect on the heat transfer rate and the surface pressure of the double-cone geometry. However, in all cases tested to date, numerical calculations predict that the laminar viscous perfect gas flow becomes unsteady before this jet is fully formed, due to the presence of the large separated flow region at the corner of the cones. As the second cone half angle increases, more fluid is reversed into the separation zone, causing it to grow. As the re-attachment point moves up the second cone, the shape of the bow shock changes, and the downward deflection of the streamlines necessary to cause the underexpansion is delayed. At the same time the separation point moves down the first cone. Eventually the

separation point reaches the nose of the first cone and the oblique shock detaches. The complex shock interaction pattern is then replaced by a single bow shock, and the mechanism for reversing fluid into the separation region disappears. The bow shock then collapses onto the body, the oblique shocks reform, and the cycle begins again. This is essentially the unsteady flow pattern reported by Maull (1960) in his experimental study of spike-tipped body flow.

Type IV shock interactions have been observed both experimentally and computationally in reacting nitrogen flow over double-wedges and double-cones [Olejniczak *et al.* (1996b)]. This is because the large amount of endothermic dissociation occurring in these flows removes thermal energy from the fluid and consequently reduces the size of the separation zone. This delays the onset of the Maull oscillations. In addition, the re-attachment point occurs closer to the corner of the wedges, which allows the downward streamline deflection necessary for the formation of an underexpanded jet to occur. This same type of result may be possible if the flow is turbulent. The presence of a turbulent boundary layer on the first cone would delay the onset of separation, and the energy transferred into the turbulent motion of the gas would decrease the size of the separated flow region. This could lead to the formation of a steady Type IV shock interaction. In addition, it is possible that some of the unsteady flows may pass through a Type IV configuration during their oscillatory behavior, and therefore transient pressure variations may be observed.

Chapter 10

Summary and Conclusions

10.1 Summary

This dissertation presents a family of new implicit methods for solving the Euler and Navier-Stokes equations on massively parallel computers. This work was motivated by the increasing need for methods capable of producing large-scale CFD calculations in a time frame that makes them useful for preliminary vehicle design. It was also desired to solve flows with complex chemical and thermal nonequilibrium processes, which results in a very large equation set. Most previous efforts in this area used domain decomposition to implement a serial or vector algorithm on a parallel machine without altering the algorithm itself. This approach does not work well in many cases, due to inherent data-dependencies in the serial algorithm. In this research a different approach is taken. Successful serial algorithms are examined, and modifications are made that eliminate the data-dependencies. In this way the modified algorithms can be efficiently implemented in parallel without an explicit domain decomposition.

The first serial method examined is the Lower-Upper Symmetric Gauss-Seidel (LU-SGS) method of Yoon and Jameson. This method is attractive for the solution of large-scale problems, because of simplifications made to the implicit operator which significantly reduce the memory requirements and computational cost. This method does not parallelize effectively, due to the data-dependencies that are inherent in the Gauss-Seidel sweeping procedure. However, in Chapter 4 an approach is presented that replaces the Gauss-Seidel sweeps with a series of point relaxation steps. The resulting diagonal Data-Parallel Lower-Upper Relaxation (DP-LUR) al-

gorithm is almost perfectly data-parallel, and is shown to be effective for the solution of several test cases.

However, the diagonal DP-LUR method suffers from a degradation of the convergence rate when high Reynolds number or stiff chemically reacting flows are simulated. The reasons for this are explored in Chapter 4, and the degradation is found to be partially caused by the diagonalizing approximation made by Yoon and Jameson, which leads to an over-stabilization on high cell aspect ratio grids. Therefore, in Chapter 5 this approximation is relaxed, and the full matrix form of the DP-LUR method is developed. This method is more computationally intensive and requires more memory than the diagonal version since it requires the storage and inversion of a Jacobian matrix at each grid point. However, the full matrix method is shown in Chapter 5 to alleviate the convergence problems with high Reynolds number and chemically reacting flows. The full matrix method is also shown to converge in less CPU time than the diagonal method on the CM-5 for all of the test cases in Chapter 5.

While the full matrix DP-LUR method improves the convergence degradation problems of the diagonal method, there is still an effect of the cell aspect ratio on the convergence rate. In Chapter 6 this problem is shown to be caused by the amount of coupling in the implicit equation. In order to increase this coupling, the implicit body-normal terms are moved back to the left-hand side of the equation, and a line relaxation method is proposed. This new method is similar to the Gauss-Seidel Line Relaxation (GSLR) method of MacCormack; however, the Gauss-Seidel sweeps are again replaced with a series of relaxation steps in a manner similar to that used in Chapter 4. The resulting Data-Parallel Line Relaxation (DPLR) method now requires the storage of five Jacobian matrices for 2-D flows, and seven for 3-D flows. Therefore the memory requirements are higher than those for both DP-LUR methods. However, the DPLR method converges much faster than either of the DP-LUR methods on all cases tested. In addition, the convergence rate of

the DPLR method is essentially independent of the cell aspect ratio.

In Chapter 7 all three of the new methods are shown to have a high parallel efficiency on the Cray T3E, where nearly perfect speedup is achieved. On the CM-5 the two DP-LUR methods perform very well, achieving about 30% of the peak theoretical performance of the machine in some cases. However, the DPLR method is shown to be inefficient on the CM-5 due to the vector hardware on the processors.

Chapters 8 and 9 discuss the application of the methods to a series of shock interaction problems. Inviscid shock interactions on double-wedge geometries are examined in Chapter 8. The interactions observed correspond to four of the interactions classified by Edney. However, the geometrical constraints imposed by the double-wedge create unique features, and produce a new interaction that does not fit into Edney's classification scheme. At high Mach numbers these interactions are underexpanded, which is shown in many cases to lead to the formation of a supersonic jet trapped against the wedge surface. This jet produces large amplitude steady-state variations in the flow quantities along the wedge. These underexpanded jets do not occur for low Mach number flows. In Chapter 9, a series of experiments are performed at Princeton University to examine shock interactions on double-cone geometries. The results are then compared to laminar CFD calculations. The Edney Type VI and Type V shock interactions are observed, and the CFD is in good agreement with the experimental data.

10.2 Conclusions

Three new parallel methods have been developed for the solution of large scale compressible flows. All three methods have good convergence properties for many problems, and all have a high parallel efficiency when implemented in either the data-parallel or message passing styles. All of the methods were formulated so that they are inherently data-parallel. Therefore, they can be implemented on a data-parallel computer directly without an explicit domain decomposition. This also

makes the methods readily portable to many parallel computers, since the rigid communication patterns of a data-parallel code make it easy to modify it for use on a message-passing or shared memory machine.

The DPLR method is by far the most cost effective of the three, typically converging about 7 times faster than the full matrix DP-LUR method and 10 times faster than the diagonal method on a Cray T3E. However, this does not mean the two DP-LUR methods should be discarded. The diagonal DP-LUR method is entirely matrix free, so it is the most memory efficient approach. Therefore, this method will remain useful for large-scale (memory limited) problems. The full matrix DP-LUR method also requires less memory than DPLR, and is more cost effective than the diagonal method for most high Mach number flows. In addition, the full matrix method is a point relaxation algorithm, which means that there is no preferred direction during the solution procedure. On the other hand, the DPLR method is a line relaxation approach, which assumes a strong body-normal coupling of the physical problem. This is a good assumption for many external flows, but may be a poor assumption in separated flows such as wakes, and in internal flows. For these cases the pointwise treatment of the full matrix method may be a better approach. From this we see that each of the new methods will have applications to which it is best suited. The high parallel efficiency and good convergence properties of the new methods make them attractive for the large-scale simulation of compressible flows.

These new methods were applied to a series of shock interaction problems, Inviscid shock interactions were investigated on double-wedge geometries, where it was discovered that geometrical constraints can influence the basic structure of the interaction, and can lead to new shock interactions that do not fit into Edney's classification scheme. One such interaction was observed for high Mach number flows, the Type IVr. This interaction is similar to the Type IV, however the underexpanded jet in this case does not impinge on the body surface, but

travels along it leading to a series of large amplitude steady-state variations in the flow quantities. The formation of this jet is usually associated only with the Type IV interaction, but this work showed that the jet is actually caused by an underexpansion of the flow along the wedge surface with respect to the outboard flow. Therefore the underexpanded jet can also be observed for other interactions. In addition, it was discovered that there is a critical Mach number that governs the formation of the underexpanded jet. For cases below this critical Mach number the flow along the body surface actually becomes overexpanded, and the jet never forms. This critical Mach number can be determined analytically from compressible gasdynamics.

Finally, a series of experiments were conducted on double-cone shock interactions, and the results were compared to laminar CFD calculations. The CFD solutions compared very well for the Type VI interaction, which was expected to be relatively unaffected by turbulence. Comparison was also good for the Type V interaction at the low Reynolds number operating condition. However, when the Reynolds number was increased by a factor of two, agreement between the CFD and experiment was not as good. The good agreement at the low Reynolds number operating condition suggests that the CFD is doing a good job reproducing the laminar structure of these flows. Therefore, one explanation for the worse agreement at the high Reynolds number condition is the effect of turbulence on the flowfield. It is hoped that further research in this area will resolve this problem.

Appendix A

Constants Used in the Air Model

Species	θ_{vs}
N_2	3395
O_2	2239
NO	2817

Table A.1 Characteristic harmonic oscillator vibrational temperatures (K).

Species	h_s°
N_2	0.0
O_2	0.0
NO	2.996123×10^6
N	3.362161×10^7
O	1.543119×10^7

Table A.2 Heats of formation (J/kg).

Species	A_s	B_s	C_s
N_2	0.0268142	0.3177838	-11.3155513
O_2	0.0449290	-0.0826158	-9.2019475
NO	0.0436378	-0.0335511	-9.5767430
N	0.0115572	0.6031679	-12.4327495
O	0.0203144	0.4294404	-11.6031403

Table A.3 Viscosity Coefficients for Blottner Model.

Reaction	C_{fm} (m^3/kg s)		η_m	θ_m (K)
	m = diatomic	m = monatomic		
1	3.70×10^{18}	1.11×10^{19}	-1.60	113200
2	2.75×10^{16}	8.25×10^{16}	-1.00	59500
3	2.30×10^{14}	4.60×10^{14}	-0.50	75500
4	3.18×10^{10}	3.18×10^{10}	0.10	37700
5	2.16×10^{5}	2.16×10^{5}	1.29	19220

Table A.4 Arrhenius coefficients for forward reaction rates.

Reaction	C_m	A_{1m}	A_{2m}	A_{3m}	A_{4m}	A_{5m}
1	1000	3.898	-12.611	0.683	-0.118	0.006
2	1000	1.335	-4.127	-0.616	0.093	-0.005
3	1000	1.549	-7.784	0.228	-0.043	0.002
4	0	2.349	-4.828	0.455	-0.075	0.004
5	0	0.215	-3.652	0.843	-0.136	0.007

Table A.5 Constants for computing K_{eq}.

Appendix B

Perfect Gas Flux Jacobians

In this appendix we present the flux Jacobians, and the matrices that are used to construct the split flux Jacobians for a 2-D perfect gas flow. The results presented here are easily extended to a three-dimensional flow [Gnoffo (1990)], or a chemically reacting flow [Candler (1988)]. The Jacobians are ordered in the same way as the flux vector presented in Eq. (3.4).

The unsplit Jacobian A is given by

$$A = \frac{\partial F}{\partial U} = \begin{pmatrix} 0 & s_x & s_y & 0 \\ -uu' + \frac{\partial p}{\partial \rho} s_x & u' + us_x + \frac{\partial p}{\partial \rho u} s_x & us_y + \frac{\partial p}{\partial \rho v} s_x & \frac{\partial p}{\partial E} s_x \\ -vu' + \frac{\partial p}{\partial \rho} s_y & vs_x + \frac{\partial p}{\partial \rho u} s_y & u' + vs_y + \frac{\partial p}{\partial \rho v} s_y & \frac{\partial p}{\partial E} s_y \\ u'(\frac{\partial p}{\partial \rho} - \frac{E+p}{\rho}) & u' \frac{\partial p}{\partial \rho u} + \frac{E+p}{\rho} s_x & u' \frac{\partial p}{\partial \rho v} + \frac{E+p}{\rho} s_y & u'(1 + \frac{\partial p}{\partial E}) \end{pmatrix}, \tag{A.1}$$

where

$$\begin{aligned} \frac{\partial p}{\partial \rho} &= \frac{1}{2}\gamma(u^2 + v^2) \\ \frac{\partial p}{\partial \rho u} &= (1-\gamma)u \\ \frac{\partial p}{\partial \rho v} &= (1-\gamma)v \\ \frac{\partial p}{\partial E} &= (\gamma - 1). \end{aligned} \tag{A.2}$$

where $\gamma = \frac{c_p}{c_v}$.

The split Jacobians are composed from their components, given by Eq. (3.10). The diagonal matrix of eigenvalues is

$$\Lambda = \mathrm{diag}(u', u' + a, u', u' - a). \tag{A.3}$$

The left and right eigenvector matrices are

$$R = \begin{pmatrix} 1 & \rho & 0 & \rho \\ 0 & as_x & -s_y & -as_x \\ 0 & as_y & s_x & -as_y \\ 0 & \rho a^2 & 0 & \rho a^2 \end{pmatrix}, \tag{A.4}$$

$$R^{-1} = \begin{pmatrix} 1 & 0 & 0 & -1/a^2 \\ 0 & s_x/2a & s_y/2a & 1/2\rho a^2 \\ 0 & -s_y & s_x & 0 \\ 0 & -s_x/2a & -s_y/2a & 1/2\rho a^2 \end{pmatrix}. \tag{A.5}$$

The transformation matrices between primitive and conserved variables are given by

$$S = \frac{\partial V}{\partial U} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -u/\rho & 1/\rho & 0 & 0 \\ -v/\rho & 0 & 1/\rho & 0 \\ \frac{\partial p}{\partial \rho} & \frac{\partial p}{\partial \rho u} & \frac{\partial p}{\partial \rho v} & \frac{\partial p}{\partial \rho} \end{pmatrix}, \tag{A.6}$$

$$S^{-1} = \frac{\partial U}{\partial V} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ u & \rho & 0 & 0 \\ v & 0 & \rho & 0 \\ \frac{1}{2}(u^2+v^2) & \rho u & \rho v & \frac{1}{\gamma-1} \end{pmatrix}. \tag{A.7}$$

The viscous Jacobians are given by Tysinger and Caughey (1991), or Gnoffo (1990). For a perfect gas flow we can write

$$L = -\frac{\mu S}{\rho J} \begin{pmatrix} 0 & 0 & 0 & 0 \\ u + \frac{1}{3}u's_x & -(1+\frac{1}{3}s_x^2) & -\frac{1}{3}s_x s_y & 0 \\ v + \frac{1}{3}u's_y & -\frac{1}{3}s_x s_y & -(1+\frac{1}{3}s_y^2) & 0 \\ l_{41} & -u - \frac{1}{3}u's_x + \frac{\gamma}{Pr}u & -v - \frac{1}{3}u's_y + \frac{\gamma}{Pr}v & -\frac{\gamma}{Pr} \end{pmatrix}, \tag{A.8}$$

where

$$l_{41} = u^2 + v^2 + \frac{1}{3}u'^2 - \frac{\gamma}{Pr}(E/\rho - u^2 - v^2).$$

References

Amodio, P. and Mazzia, F., (1995) "A Parallel Gauss-Seidel Method for Block-Tridiagonal Linear Systems," *SIAM Journal of Scientific Computing,* Vol. 16, No. 6, pp. 1451-1461.

Anderson, J. D., (1989) *Hypersonic and High Temperature Gas Dynamics,* McGraw Hill, New York.

Bailey, D., Barton, J., Lasinski, T., and Simon, H., ed., (1993) "The NAS Parallel Benchmarks," NASA Technical Memorandum 103863.

Bailey, D., Barszcz, E., Dagum, L., and Simon, H., (1994) "NAS Parallel Benchmark Results 3-94," NASA Technical Report RNR-94-006.

Barszcz, E., Fatoohi, R., Venkatakrishnan, V., and Weeratunga, S., (1993) "Solution of Regular, Sparse Triangular Linear Systems on Vector and Distributed-Memory Multiprocessors," NASA Technical Report RNR-93-007.

Baumgartner, M. L., Smits, A. J., Nau, T., and Rowley, C. W., (1995) "A New Hypersonic Boundary Layer Facility," AIAA Paper No. 95-0787.

Bertin, J. J., Graumann, B. W., and Goodrich, W. D., (1973) "Aerothermodynamic Aspects of Shock-Interference Patterns for Shuttle Configurations During Entry," AIAA Paper No. 73-0238.

Bertin, J. J. and Hinkle, J. C., (1975) "Experimental Investigation of Supersonic Flow Past Double Wedge Configurations," *AIAA Journal,* Vol. 13, No. 7, pp. 897-901.

Bibb, K. L., Peraire, J., and Riley, C. J., (1997) "Hypersonic Flow Computations on Unstructured Meshes," AIAA Paper No. 97-0625.

Birch, S. F. and Keyes, J. W., (1972) "Transition in Compressible Free Shear Layers," *Journal of Spacecraft and Rockets,* Vol. 9, No. 8, pp. 623-624.

Blottner, F. G., Johnson, M., and Ellis, M., (1971) "Chemically Reacting Viscous Flow Program for Multi-Component Gas Mixtures," Sandia Laboratories Report No. SC-RR-70-754, Albuquerque, New Mexico.

Bose, D. and Candler, G. V., (1996) "Thermal Rate Constants of the $N_2 + O \rightarrow NO + N$ Reaction Rate Using *Ab Initio* $^3A''$ and $^3A'$ Potential Energy Surfaces," *Journal of Chemical Physics,* Vol. 104, No. 8, pp. 2825-2833.

Boyd, I. D., Chen, G., and Candler, G. V., (1995) "Predicting Failure of the Continuum Fluid Equations in Transitional Hypersonic Flows," *Physics of Fluids,* Vol. 7, No. 1, pp. 210-219.

Buelow, P. E., Venkateswaran, S., and Merkle, C. L., (1994) "Effect of Grid Aspect Ratio on Convergence," *AIAA Journal,* Vol. 32, No. 12, pp. 2401-2408.

Candler, G. V., (1988) "The Computation of Weakly Ionized Hypersonic Flows in Thermochemical Nonequilibrium," PhD. Thesis, Stanford University.

Candler, G. V. and Olynick, D. R., (1992) "Hypersonic Flow Simulations Using a Diagonal Implicit Method," *Proceedings of the 10th International Conference on Computing Methods in Applied Science and Engineering*, ed. R. Glowinski, Nova Science Publishers, New York, pp. 29-48.

Candler, G. V., Wright, M. J., and McDonald, J.D., (1994) "Data-Parallel Lower-Upper Relaxation Method for Reacting Flows," *AIAA Journal,* Vol. 32, No. 12, pp. 2380-2386.

Clarke, J. F. and McChesney, M., (1975) *Dynamics of Relaxing Gases,* Butterworths, London.

Eberhardt, S. and Imlay, S. T., (1990) "A Diagonal Implicit Scheme for Computing Flows with Finite Rate Chemistry," AIAA Paper No. 90-1577.

Edney, B., (1968) "Anomalous Heat Transfer and Pressure Distributions on Blunt Bodies at Hypersonic Speeds in the Presence of an Impinging Shock," Report 115, Flygtekniska Försöksanstalten (The Aeronautical Research Institute of Sweden), Stockholm.

Edwards, J., (1994) "A Diagonal Implicit/Multigrid Algorithm for Computing Hypersonic, Chemically Reacting Viscous Flows," AIAA Paper No. 94-0762.

Evans, D. J., (1984) "Parallel SOR Iterative Methods," *Parallel Computing*, Vol. 1, No. 1, pp. 3-18.

Gnoffo, P. A., (1990) "An Upwind-Biased, Point Implicit Relaxation Algorithm for Viscous, Compressible Perfect-Gas Flows," NASA TP 2953.

Gupta, R. N., Yos, J. M., Thompson, R. A., and Lee, K., (1990) "A Review of Reaction Rates and Thermodynamic and Transport Properties for an 11-Species Air Model for Chemical and Thermal Nonequilibrium Calculations to 30,000K," NASA RP 2953.

Hains, F. D. and Keyes, J. W., (1972) "Shock Interference Heating in Hypersonic Flows," *AIAA Journal,* Vol. 10, No. 11, pp. 1441-1447.

Hassan, B., Candler, G. V., and Olynick, D. R., (1993) "The Effect of Thermo-Chemical Nonequilibrium on the Aerodynamics of Aerobraking Vehicles," *Journal of Spacecraft and Rockets,* Vol. 30, No. 6, pp. 647-655.

Hirsch, C., (1991) *Numerical Computation of Internal and External Flows,* Wiley, New York.

Holden, M. S. and Moselle, J. R., (1970) "Theoretical and Experimental Studies of the Shock Wave-Boundary Layer Interaction on Compression Surfaces in Hypersonic Flow," Technical Report ARL 70-0002, Aerospace Research Laboratories, WPAFB, OH.

Holden, M. S., (1978) "A Study of Flow Separation in Regions of Shock Wave-Boundary Layer Interaction in Hypersonic Flow," AIAA Paper No. 78-1169.

Hunt, J. L. and Creel, T. R., (1971) "Shock-Interference Heating and Density-Ratio Effects, Part II – Hypersonic Density-Ratio Effects," NASA Space Shuttle Technology Conference, Volume I – Aerothermodynamics, Configurations, and Flight Mechanics, TMX-2272.

Johnson, H., Candler, G. V., and Hudson, M., (1997) "Numerical Study of Hypersonic Boundary Layer Transition on a Blunt Body," AIAA Paper No. 97-0554.

Kandula, M. and Buning, P. G., (1994) "Implementation of LU-SGS Algorithm and Roe Upwinding Scheme in OVERFLOW Thin-Layer Navier-Stokes Code," AIAA Paper No. 94-2357.

Klopfer, G. H. and Yee, H. C., (1988) "Viscous Hypersonic Shock-on-Shock Interaction on Blunt Cowl Lips," AIAA Paper No. 88-0233.

Lamont, P. J. and Hunt, B. L., (1980) "The Impingement of Underexpanded, Axisymmetric Jets on Perpendicular and Inclined Plates," *Journal of Fluid Mechanics,* Vol. 100, No. 3, pp. 471-511.

Lee, J. H., (1986) "Basic Governing Equations for the Flight Regime of Aeroassisted Orbital Transfer Vehicles," *Thermal Design of Aeroassisted Orbital Transfer Vehicles,* ed. H. F. Nelson, Progress in Aeronautics and Astronautics, Vol. 96.

Levin, D. A., Collins, R. J., Candler, G. V., Wright, M. J., and Erdman, P.W., (1996) "Examination of OH Ultraviolet Radiation From Shock Heated Air," *Journal of Thermophysics and Heat Transfer,* Vol. 10, No. 2, pp. 200-208.

Liou, M. S. and Van Leer, B., (1988) "Choice of Implicit and Explicit Operators for the Upwind Differencing Method," AIAA Paper No. 88-0624.

MacCormack, R. W., (1985) "Current Status of the Numerical Solutions of the Navier-Stokes Equations," AIAA Paper No. 85-0032.

MacCormack, R. W. and Candler, G. V., (1989) "The Solution of the Navier-Stokes Equations Using Gauss-Seidel Line Relaxation," *Computers and Fluids,* Vol. 17, No. 1, pp. 135-150.

MacCormack, R.W., (1990) "Solution of the Navier-Stokes Equations in Three Dimensions," AIAA Paper No. 90-1520.

Macheret, S. O. and Rich, J. W., (1993) "Nonequilibrium Dissociation Rates Behind Strong Shock Waves: Classical Model," *Chemical Physics,* Vol. 174, pp. 25-43.

Malone, M. B., (1996) "Turbulence Model Evaluation for Free Shear Layer Dominated Flows," AIAA Paper No. 96-2038.

Marrone, P. V. and Treanor, C. E., (1963) "Chemical Relaxation with Preferential Dissociation from Excited Vibrational Levels," *Physics of Fluids,* Vol. 6, No. 9, pp. 1215-1221.

Maull, D. J., (1960) "Hypersonic Flow over Axially Symmetric Spiked Bodies," *Journal of Fluid Mechanics,* Vol. 8, No. 4, pp. 584-592.

Merzkirch, W., (1987) *Flow Visualization,* Harcourt Brace Jovanovich, pp. 115-218.

Millikan, R. C. and White, D. R., (1963) "Systematics of Vibrational Relaxation," *Journal of Chemical Physics,* Vol. 39, pp. 3209-3213.

Obayashi, S., (1988) "Numerical Solutions of Underexpanded Plumes Using Upwind Algorithms," AIAA Paper No. 88-4360.

Obayashi, S. and Guruswamy, G. P., (1994) "Convergence Acceleration of an Aeroelastic Navier-Stokes Solver," AIAA Paper No. 94-2268.

Olejniczak, J. and Candler, G. V., (1995) "Study of Experiments Sensitive to Vibration-Dissociation Coupling Models," Proceedings of the 20th International Symposium on Shock Waves.

Olejniczak, J., Wright, M. J., and Candler, G. V., (1996a) "Numerical Study of Shock Interactions on Double-Wedge Geometries," AIAA Paper No. 96-0041.

Olejniczak, J., Candler, G. V., Wright, M. J., Hornung, H. G., and Leyva, I. A, (1996b) "High Enthalpy Double-Wedge Experiments," AIAA Paper No. 96-2238.

Olejniczak, J., (1997) "Computational and Experimental Study of Nonequilibrium Chemistry in Hypersonic Flows," PhD. Thesis, University of Minnesota.

Park, C., (1985) "On Convergence of Computation of Chemically Reacting Flows," AIAA Paper No. 85-0247.

Park, C., (1986) "Assessment of Two-Temperature Kinetic Model for Dissociating and Weakly Ionizing Nitrogen," AIAA Paper No. 86-1347.

Park, C., (1988) "Two Temperature Interpretation of Dissociation Rate Data for N_2 and O_2," AIAA Paper No. 88-0458.

Saini, S. and Bailey, D. H., (1996) "NAS Parallel Benchmark (Version 1.0) Results 11-96," NAS Technical Report NAS-96-018.

Simon, H. D., ed., (1992) *Parallel Computational Fluid Dynamics Implementations and Results,* MIT Press, Cambridge, MA.

Srinivasan, G. R. and Baeder, J. D., (1993) "TURNS: A Free-Wake Euler/Navier-Stokes Numerical Method for Helicopter Rotors," *AIAA Journal,* Vol. 31, No. 5, pp. 959-962.

Svehla, R. A., (1962) "Estimated Viscosities and Thermal Conductivities of Gases at High Temperatures," NASA TR R-132.

Steger, J. L. and Warming, R. F., (1981) "Flux Vector Splitting of the Inviscid Gasdynamic Equations with Application to Finite Difference Methods," *Journal of Computational Physics,* Vol. 40, pp. 263-293.

Swanson, R. and Turkel, E., (1985) "A Multistage Time-Stepping Scheme for the Navier-Stokes Equations," AIAA Paper No. 85-0035.

Taylor, S. and Wang, J. C., (1995) "Launch Vehicle Simulations Using a Concurrent Implicit Navier-Stokes Solver," AIAA Paper No. 95-0223.

Tysinger, T. and Caughey, D., (1991) "Implicit Multigrid Algorithm for the Navier-Stokes Equations," AIAA Paper No. 91-0242.

Vatsa, V. and Wedan, B., (1990) "Development of a Multigrid Code for 3-D Navier-Stokes Equations and its Application to a Grid Refinement Study," *Computers & Fluids,* Vol. 18, No. 4, pp. 391-403.

Vincenti, W. G. and Kruger, C. H., (1986) *Introduction to Physical Gas Dynamics,* Krieger Pub., Malabar, Florida.

Wang, J. C. and Widhopf, G. F., (1990) "An Efficient Finite Volume TVD Scheme for Steady State Solutions of the 3-D Compressible Euler/Navier-Stokes Equations," AIAA Paper No. 90-1523.

White, F. M., (1991) *Viscous Fluid Flow,* McGraw-Hill, New York.

Wilke, C. R., (1950) "A Viscosity Equation for Gas Mixtures," *Journal of Chemical Physics,* Vol. 18, pp. 517-519.

Wissink, A. M., Lyrintzis, A. S., and Strawn, R. C., (1996) "Parallelization of a Three-Dimensional Flow Solver for Euler Rotorcraft Aerodynamics Predictions," *AIAA Journal,* Vol. 34, No. 11, pp. 2276-2283.

Wong, C. C., Blottner, F. G., and Payne, J. L., (1995) "A Domain Decomposition Study of Massively Parallel Computing in Compressible Gas Dynamics," AIAA Paper 95-0572.

Wood, C. J., (1962) "Hypersonic Flow Past Spiked Cones," *Journal of Fluid Mechanics,* Vol. 12, pp. 614-624.

Wright, M. J., Candler, G. V., and Prampolini, M., (1996) "Data-Parallel Lower Upper Relaxation Method for the Navier-Stokes Equations," *AIAA Journal,* Vol. 34, No. 7, pp. 1371-1377.

Wright, M. J., Olejniczak, J., Candler, G. V., Magruder, T. D., and Smits, A. J., (1997a) "Numerical and Experimental Investigation of Double-Cone Shock Interactions," AIAA Paper No. 97-0063.

Wright, M. J., Candler, G. V., and Bose, D., (1997b) "A Data-Parallel Line Relaxation Method for the Navier-Stokes Equations," AIAA Paper No. 97-2046.

Yee, H. C., (1989) "A Class of High-Resolution Explicit and Implicit Shock Capturing Methods," NASA TM 101088.

Yoon, S. and Jameson, A., (1987) "An LU-SSOR Scheme for the Euler and Navier-Stokes Equations," AIAA Paper No. 87-0600.

Yoon, S. and Jameson, A., (1988) "A Lower-Upper Symmetric Gauss-Seidel Method for the Euler and Navier-Stokes Equations," *AIAA Journal,* Vol. 26, No. 9, pp. 1025-1026.

Yoon, S. and Kwak, D., (1994) "Multigrid Convergence of an Implicit Symmetric Relaxation Scheme," *AIAA Journal,* Vol. 32, No. 5, pp. 950-955.